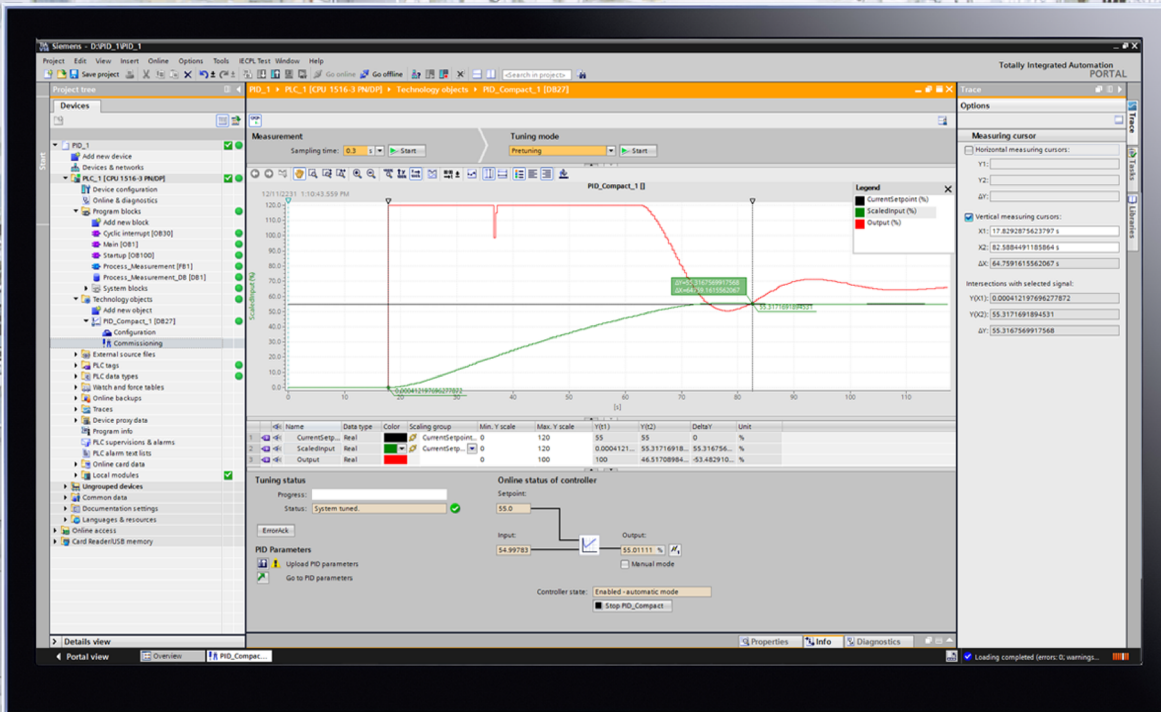


# SIEMENS



# SIMATIC

## S7-1200, S7-1500

### PID control

#### Function manual

Edition

09/2016

siemens.com

# SIEMENS

## SIMATIC

### S7-1200, S7-1500 PID control

#### Function Manual

#### Preface

---

Documentation guide

1

Principles for control

2

Configuring a software  
controller

3

Using PID\_Compact

4

Using PID\_3Step

5

Using PID\_Temp

6

Using PID basic functions

7

Instructions

8

Service & Support

A

## Legal information

### Warning notice system

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

#### **DANGER**

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

#### **WARNING**

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

#### **CAUTION**

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

#### **NOTICE**

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

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

### Qualified Personnel

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

### Proper use of Siemens products

Note the following:

#### **WARNING**

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

### Trademarks

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

### Disclaimer of Liability

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

# Preface

## Purpose of the documentation

This documentation will support you in configuring and programming control tasks with the S7-1200 and S7-1500 automation systems.

## Basic knowledge required

The following knowledge is required in order to understand the documentation:

- General knowledge of automation technology
- Knowledge of the industrial automation system SIMATIC
- Experience of working with STEP 7 (TIA Portal)

## Validity of the documentation

This documentation applies to the use of SW controllers on the CPUs of automation systems S7-1200 and S7-1500 together with STEP 7 (TIA Portal). Additional SW controllers that are not covered in this documentation are available for the use of S7-300 and S7-400 with STEP 7 (TIA Portal). Section Overview of software controller (Page 38) gives a complete overview of all SW controllers in STEP 7 (TIA Portal) and their possible applications.

## Conventions

Please observe notes marked as follows:

---

### Note

The notes contain important information on the product described in the documentation, on the handling of the product or on part of the documentation to which particular attention should be paid.

---

## Additional assistance

- Information on the offers of our Technical Support are available in the appendix Service & Support (Page 520).
- The range of technical documentation for the individual SIMATIC products and automation systems is available on the Internet (<http://www.siemens.com/simatic-tech-doku-portal>).
- The online catalog and the ordering system are available on the Internet (<http://mall.automation.siemens.com>).

# Table of contents

	<b>Preface .....</b>	<b>4</b>
<b>1</b>	<b>Documentation guide .....</b>	<b>12</b>
<b>2</b>	<b>Principles for control .....</b>	<b>16</b>
2.1	Controlled system and actuators .....	16
2.2	Controlled systems .....	18
2.3	Characteristic values of the control section .....	20
2.4	Pulse controller .....	23
2.5	Response to setpoint changes and disturbances .....	27
2.6	Control Response at Different Feedback Structures .....	29
2.7	Selection of the controller structure for specified controlled systems.....	36
2.8	PID parameter settings .....	37
<b>3</b>	<b>Configuring a software controller .....</b>	<b>38</b>
3.1	Overview of software controller .....	38
3.2	Steps for the configuration of a software controller .....	40
3.3	Add technology objects.....	40
3.4	Comparing values .....	42
3.4.1	Comparison display and boundary conditions.....	42
3.4.2	Comparing values .....	43
3.5	Configure technology objects .....	45
3.6	Call instruction in the user program.....	46
3.7	Parameter view .....	47
3.7.1	Introduction to the parameter view .....	47
3.7.2	Structure of the parameter view.....	49
3.7.2.1	Toolbar.....	49
3.7.2.2	Navigation .....	50
3.7.2.3	Parameter table .....	51
3.7.3	Opening the parameter view.....	53
3.7.4	Default setting of the parameter view .....	54
3.7.5	Working with the parameter view.....	57
3.7.5.1	Overview .....	57
3.7.5.2	Filtering the parameter table.....	58
3.7.5.3	Sorting the parameter table .....	59
3.7.5.4	Transferring parameter data to other editors.....	59
3.7.5.5	Indicating errors .....	60
3.7.5.6	Editing start values in the project.....	61
3.7.5.7	Status of configuration (offline) .....	63
3.7.5.8	Monitoring values online in the parameter view .....	64

3.7.5.9	Create snapshot of monitor values .....	65
3.7.5.10	Modifying values .....	66
3.7.5.11	Comparing values .....	68
3.7.5.12	Applying values from the online program as start values .....	70
3.7.5.13	Initializing setpoints in the online program.....	71
3.8	Downloading technology objects to device .....	72
3.9	Commissioning software controller .....	74
3.10	Save optimized PID parameter in the project .....	74
3.11	Display instance DB of a technology object.....	75
<b>4</b>	<b>Using PID_Compact .....</b>	<b>76</b>
4.1	PID_Compact V2.....	76
4.1.1	Configuring PID_Compact V2 .....	76
4.1.1.1	Basic settings V2.....	76
4.1.1.2	Process value settings V2.....	80
4.1.1.3	Advanced settings V2 .....	81
4.1.2	Commissioning PID_Compact V2.....	89
4.1.2.1	Pretuning V2 .....	89
4.1.2.2	Fine tuning V2.....	91
4.1.2.3	"Manual" mode V1 .....	93
4.1.3	Override control with PID_Compact V2 .....	94
4.1.4	Simulating PID_Compact V2 with PLCSIM.....	98
4.2	PID_Compact V1.....	99
4.2.1	Configuring PID_Compact V1 .....	99
4.2.1.1	Basic settings V1.....	99
4.2.1.2	Process value settings V1.....	103
4.2.1.3	Advanced settings V1 .....	104
4.2.2	Commissioning PID_Compact V1.....	112
4.2.2.1	Commissioning V1 .....	112
4.2.2.2	Pretuning V1 .....	113
4.2.2.3	Fine tuning V1.....	115
4.2.2.4	"Manual" mode V1 .....	117
4.2.3	Simulating PID_Compact V1 with PLCSIM.....	118
4.3	Technology object PID_Compact.....	119
<b>5</b>	<b>Using PID_3Step .....</b>	<b>120</b>
5.1	Technology object PID_3Step.....	120
5.2	PID_3Step V2.....	121
5.2.1	Configuring PID_3Step V2 .....	121
5.2.1.1	Basic settings V2.....	121
5.2.1.2	Process value settings V2.....	126
5.2.1.3	Final controlling element settings V2 .....	127
5.2.1.4	Advanced settings V2 .....	131
5.2.2	Commissioning PID_3Step V2.....	135
5.2.2.1	Pretuning V2 .....	135
5.2.2.2	Fine tuning V2.....	137
5.2.2.3	Commissioning with manual PID parameters V2 .....	139
5.2.2.4	Measuring the motor transition time V2 .....	140
5.2.3	Simulating PID_3Step V2 with PLCSIM.....	143

5.3	PID_3Step V1 .....	144
5.3.1	Configuring PID_3Step V1 .....	144
5.3.1.1	Basic settings V1 .....	144
5.3.1.2	Process value settings V1 .....	149
5.3.1.3	V1 final controlling element setting .....	150
5.3.1.4	Advanced settings V1 .....	153
5.3.2	Commissioning PID_3Step V1 .....	157
5.3.2.1	Commissioning V1 .....	157
5.3.2.2	Pretuning V1 .....	158
5.3.2.3	Fine tuning V1 .....	159
5.3.2.4	Commissioning with manual PID parameters V1 .....	160
5.3.2.5	Measuring the motor transition time V1 .....	161
5.3.3	Simulating PID_3Step V1 with PLCSIM .....	163
<b>6</b>	<b>Using PID_Temp .....</b>	<b>164</b>
6.1	Technology object PID_Temp .....	164
6.2	Configuring PID_Temp .....	165
6.2.1	Basic settings .....	165
6.2.1.1	Introduction .....	165
6.2.1.2	Controller type .....	166
6.2.1.3	Setpoint .....	166
6.2.1.4	Process value .....	167
6.2.1.5	Heating and cooling output value .....	167
6.2.1.6	Cascade .....	169
6.2.2	Process value settings .....	170
6.2.2.1	Process value limits .....	170
6.2.2.2	Process value scaling .....	170
6.2.3	Output settings .....	171
6.2.3.1	Basic settings of output .....	171
6.2.3.2	Output value limits and scaling .....	174
6.2.4	Advanced settings .....	177
6.2.4.1	Process value monitoring .....	177
6.2.4.2	PWM limits .....	178
6.2.4.3	PID parameters .....	181
6.3	Commissioning PID_Temp .....	188
6.3.1	Commissioning .....	188
6.3.2	Pretuning .....	189
6.3.3	Fine tuning .....	192
6.3.4	"Manual" mode .....	196
6.3.5	Substitute setpoint .....	197
6.3.6	Cascade commissioning .....	197
6.4	Cascade control with PID_Temp .....	198
6.4.1	Introduction .....	198
6.4.2	Program creation .....	200
6.4.3	Configuration .....	202
6.4.4	Commissioning .....	204
6.4.5	Substitute setpoint .....	205
6.4.6	Operating modes and fault response .....	205

6.5	Multi-zone controlling with PID_Temp .....	206
6.6	Override control with PID_Temp .....	209
6.7	Simulating PID_Temp with PLCSIM .....	213
<b>7</b>	<b>Using PID basic functions .....</b>	<b>214</b>
7.1	CONT_C .....	214
7.1.1	Technology object CONT_C .....	214
7.1.2	Configure controller difference CONT_C .....	215
7.1.3	Configure the controller algorithm CONT_C .....	216
7.1.4	Configure the output value CONT_C .....	217
7.1.5	Programming a pulse controller .....	218
7.1.6	Commissioning CONT_C .....	219
7.2	CONT_S .....	220
7.2.1	Technology object CONT_S .....	220
7.2.2	Configure controller difference CONT_S .....	221
7.2.3	Configuring control algorithm CONT_S .....	221
7.2.4	Configure manipulated value CONT_S .....	222
7.2.5	Commissioning CONT_S .....	222
7.3	TCONT_CP .....	223
7.3.1	Technology object TCONT_CP .....	223
7.3.2	Configure TCONT_CP .....	224
7.3.2.1	Controller difference .....	224
7.3.2.2	Controlling algorithm .....	225
7.3.2.3	Manipulated value continual controller .....	227
7.3.2.4	Manipulated value pulse controller .....	227
7.3.3	Commissioning TCONT_CP .....	230
7.3.3.1	Optimization of TCONT_CP .....	230
7.3.3.2	Requirements for an optimization .....	232
7.3.3.3	Possibilities for optimization .....	234
7.3.3.4	Tuning result .....	237
7.3.3.5	Parallel tuning of controller channels .....	238
7.3.3.6	Fault descriptions and corrective measures .....	239
7.3.3.7	Performing pretuning .....	242
7.3.3.8	Performing fine tuning .....	242
7.3.3.9	Cancelling pretuning or fine tuning .....	243
7.3.3.10	Manual fine-tuning in control mode .....	243
7.3.3.11	Performing fine tuning manually .....	245
7.4	TCONT_S .....	246
7.4.1	Technology object TCONT_S .....	246
7.4.2	Configure controller difference TCONT_S .....	246
7.4.3	Configure controller algorithm TCONT_S .....	248
7.4.4	Configure manipulated value TCONT_S .....	248
7.4.5	Commissioning TCONT_S .....	249



<b>8</b>	<b>Instructions .....</b>	<b>250</b>
8.1	PID_Compact.....	250
8.1.1	New features of PID_Compact .....	250
8.1.2	Compatibility with CPU and FW.....	253
8.1.3	PID_Compact V2 .....	254
8.1.3.1	Description of PID_Compact V2 .....	254
8.1.3.2	Mode of operation of PID_Compact V2 .....	257
8.1.3.3	Input parameters of PID_Compact V2 .....	260
8.1.3.4	Output parameters of PID_Compact V2 .....	262
8.1.3.5	In/out parameters of PID_Compact V2.....	263
8.1.3.6	Static tags of PID_Compact V2 .....	264
8.1.3.7	Changing the PID_Compact V2 interface.....	272
8.1.3.8	Parameters State and Mode V2 .....	274
8.1.3.9	Parameter ErrorBits V2.....	278
8.1.3.10	Tag ActivateRecoverMode V2 .....	280
8.1.3.11	Tag Warning V2.....	282
8.1.3.12	IntegralResetMode V2 tag.....	283
8.1.3.13	Sample program for PID_Compact.....	285
8.1.4	CPU processing time and memory requirement PID_Compact V2.x.....	291
8.1.5	PID_Compact V1 .....	292
8.1.5.1	Description of PID_Compact V1 .....	292
8.1.5.2	Input parameters of PID_Compact V1 .....	296
8.1.5.3	Output parameters of PID_Compact V1 .....	297
8.1.5.4	Static tags of PID_Compact V1 .....	298
8.1.5.5	Parameters State and sRet.i_Mode V1 .....	303
8.1.5.6	Parameter Error V1.....	307
8.1.5.7	Parameter Reset V1 .....	308
8.1.5.8	Tag sd_warning V1 .....	310
8.1.5.9	Tag i_Event_SUT V1.....	310
8.1.5.10	Tag i_Event_TIR V1.....	311
8.2	PID_3Step.....	312
8.2.1	New features of PID_3Step .....	312
8.2.2	Compatibility with CPU and FW.....	314
8.2.3	CPU processing time and memory requirement PID_3Step V2.x.....	315
8.2.4	PID_3Step V2 .....	316
8.2.4.1	Description of PID_3Step V2 .....	316
8.2.4.2	Mode of operation of PID_3Step V2 .....	322
8.2.4.3	Changing the PID_3Step V2 interface.....	326
8.2.4.4	Input parameters of PID_3Step V2.....	327
8.2.4.5	Output parameters of PID_3Step V2 .....	329
8.2.4.6	In/out parameters of PID-3Step V2.....	330
8.2.4.7	Static tags of PID_3Step V2 .....	331
8.2.4.8	Parameters State and Mode V2 .....	342
8.2.4.9	Parameter ErrorBits V2.....	347
8.2.4.10	Tag ActivateRecoverMode V2 .....	350
8.2.4.11	Tag Warning V2.....	352

8.2.5	PID_3Step V1.....	353
8.2.5.1	Description PID_3Step V1 .....	353
8.2.5.2	Operating principle PID_3Step V1 .....	359
8.2.5.3	PID_3Step V1 input parameters .....	362
8.2.5.4	PID_3Step V1 output parameters .....	364
8.2.5.5	PID_3Step V1 static tags .....	366
8.2.5.6	Parameter State and Retain.Mode V1 .....	374
8.2.5.7	Parameter ErrorBits V1 .....	382
8.2.5.8	Parameter Reset V1 .....	384
8.2.5.9	Tag ActivateRecoverMode V1 .....	385
8.2.5.10	Tag Warning V1 .....	387
8.2.5.11	Tag SUT.State V1 .....	388
8.2.5.12	Tag TIR.State V1 .....	388
8.3	PID_Temp .....	389
8.3.1	New features of PID_Temp.....	389
8.3.2	Compatibility with CPU and FW.....	389
8.3.3	PID_Temp .....	390
8.3.3.1	Description of PID_Temp .....	390
8.3.3.2	Mode of operation of PID_Temp .....	395
8.3.3.3	Input parameters of PID_Temp.....	401
8.3.3.4	Output parameters of PID_Temp .....	403
8.3.3.5	In/out parameters of PID_Temp V2 .....	405
8.3.3.6	PID_Temp static tags .....	407
8.3.3.7	PID_Temp state and mode parameters .....	440
8.3.3.8	PID_Temp ErrorBits parameter.....	450
8.3.3.9	PID_Temp ActivateRecoverMode tag.....	453
8.3.3.10	PID_Temp Warning tag.....	455
8.3.3.11	PwmPeriode tag .....	456
8.3.3.12	IntegralResetMode tag.....	458
8.3.4	CPU processing time and memory requirement PID_Temp V1 .....	460
8.4	PID basic functions .....	461
8.4.1	CONT_C.....	461
8.4.1.1	Description CONT_C .....	461
8.4.1.2	How CONT_C works.....	462
8.4.1.3	CONT_C block diagram .....	464
8.4.1.4	Input parameter CONT_C .....	465
8.4.1.5	Output parameters CONT_C .....	467
8.4.2	CONT_S.....	468
8.4.2.1	Description CONT_S.....	468
8.4.2.2	Mode of operation CONT_S.....	469
8.4.2.3	Block diagram CONT_S.....	470
8.4.2.4	Input parameters CONT_S .....	471
8.4.2.5	Output parameters CONT_S.....	472
8.4.3	PULSEGEN.....	473
8.4.3.1	Description PULSEGEN .....	473
8.4.3.2	Mode of operation PULSEGEN .....	474
8.4.3.3	Mode of operation PULSEGEN .....	477
8.4.3.4	Three-step control.....	477
8.4.3.5	Two-step control.....	480

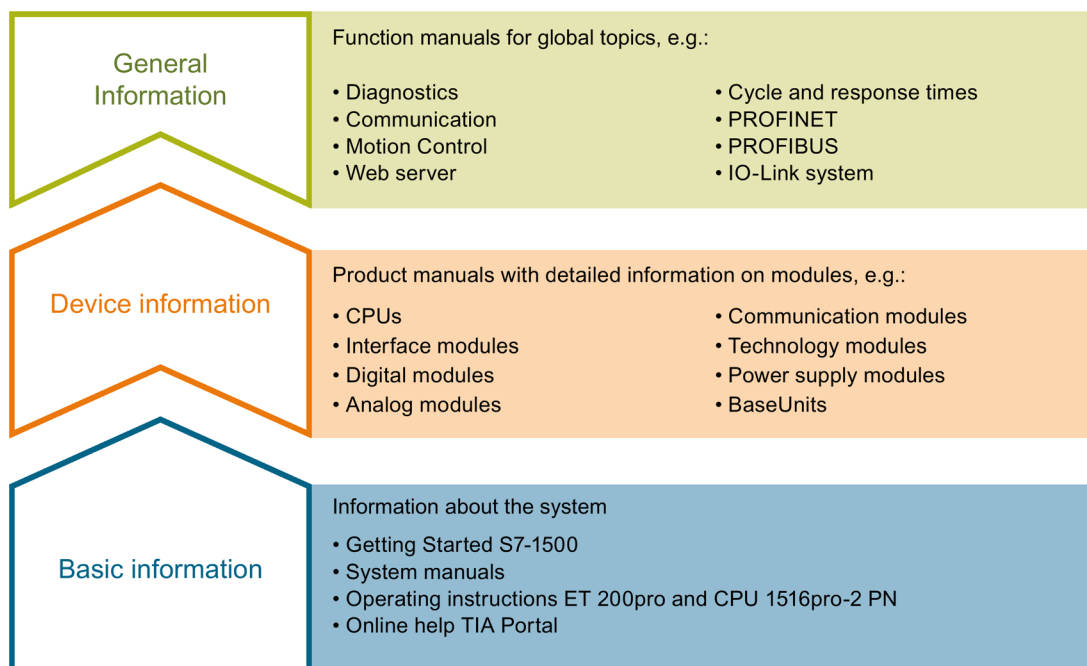
---

8.4.3.6	Input parameters PULSEGEN .....	481
8.4.3.7	Output parameter PULSEGEN .....	482
8.4.4	TCONT_CP .....	483
8.4.4.1	Description TCONT_CP.....	483
8.4.4.2	Mode of operation TCONT_CP.....	484
8.4.4.3	Operating principle of the pulse generator .....	493
8.4.4.4	Block diagram TCONT_CP .....	496
8.4.4.5	Input parameters TCONT_CP .....	498
8.4.4.6	Output parameters TCONT_CP.....	499
8.4.4.7	In/out parameters TCONT_CP .....	500
8.4.4.8	Static variables TCONT_CP .....	501
8.4.4.9	Parameter STATUS_H .....	506
8.4.4.10	Parameters STATUS_D.....	507
8.4.5	TCONT_S .....	508
8.4.5.1	Description TCONT_S .....	508
8.4.5.2	Mode of operation TCONT_S .....	509
8.4.5.3	Block diagram TCONT_S .....	513
8.4.5.4	Input parameters TCONT_S.....	515
8.4.5.5	Output parameters TCONT_S .....	516
8.4.5.6	In/out parameters TCONT_S.....	516
8.4.5.7	Static variables TCONT_S.....	517
8.4.6	Integrated system functions.....	518
8.4.6.1	CONT_C_SF .....	518
8.4.6.2	CONT_S_SF .....	518
8.4.6.3	PULSEGEN_SF .....	519
<b>A</b>	<b>Service &amp; Support.....</b>	<b>520</b>
	<b>Index.....</b>	<b>523</b>

# Documentation guide

The documentation for the SIMATIC S7-1500 automation system, for CPU 1516pro-2 PN based on SIMATIC S7-1500, and for the distributed I/O systems SIMATIC ET 200MP, ET 200SP and ET 200AL is divided into three areas.

This division allows you easier access to the specific information you require.



## Basic information

System manuals and Getting Started manuals describe in detail the configuration, installation, wiring and commissioning of the SIMATIC S7-1500, ET 200MP, ET 200SP and ET 200AL systems; use the corresponding operating instructions for CPU 1516pro-2 PN. The STEP 7 online help supports you in configuration and programming.

## Device information

Product manuals contain a compact description of the module-specific information, such as properties, terminal diagrams, characteristics and technical specifications.

### General information

The function manuals contain detailed descriptions on general topics such as diagnostics, communication, Motion Control, Web server, OPC UA.

You can download the documentation free of charge from the Internet (<http://w3.siemens.com/mcms/industrial-automation-systems-simatic/en/manual-overview/Pages/Default.aspx>).

Changes and additions to the manuals are documented in product information sheets.

You will find the product information on the Internet:

- S7-1500/ET 200MP (<https://support.industry.siemens.com/cs/us/en/view/68052815>)
- ET 200SP (<https://support.industry.siemens.com/cs/us/en/view/73021864>)
- ET 200AL (<https://support.industry.siemens.com/cs/us/en/view/99494757>)

### Manual Collections

The Manual Collections contain the complete documentation of the systems put together in one file.

You will find the Manual Collections on the Internet:

- S7-1500/ET 200MP (<https://support.industry.siemens.com/cs/ww/en/view/86140384>)
- ET 200SP (<https://support.industry.siemens.com/cs/ww/en/view/84133942>)
- ET 200AL (<https://support.industry.siemens.com/cs/ww/en/view/95242965>)

### "mySupport"

With "mySupport", your personal workspace, you make the best out of your Industry Online Support.

In "mySupport", you can save filters, favorites and tags, request CAx data and compile your personal library in the Documentation area. In addition, your data is already filled out in support requests and you can get an overview of your current requests at any time.

You must register once to use the full functionality of "mySupport".

You can find "mySupport" on the Internet (<https://support.industry.siemens.com/My/ww/en>).

### "mySupport" - Documentation

In the Documentation area in "mySupport" you can combine entire manuals or only parts of these to your own manual.

You can export the manual as PDF file or in a format that can be edited later.

You can find "mySupport" - Documentation on the Internet (<http://support.industry.siemens.com/My/ww/en/documentation>).

## "mySupport" - CAx data

In the CAx data area in "mySupport", you can access the current product data for your CAx or CAe system.

You configure your own download package with a few clicks.

In doing so you can select:

- Product images, 2D dimension drawings, 3D models, internal circuit diagrams, EPLAN macro files
- Manuals, characteristics, operating manuals, certificates
- Product master data

You can find "mySupport" - CAx data on the Internet (<http://support.industry.siemens.com/my/ww/en/CAxOnline>).

## Application examples

The application examples support you with various tools and examples for solving your automation tasks. Solutions are shown in interplay with multiple components in the system - separated from the focus on individual products.

You will find the application examples on the Internet (<https://support.industry.siemens.com/sc/ww/en/sc/2054>).

## TIA Selection Tool

With the TIA Selection Tool, you can select, configure and order devices for Totally Integrated Automation (TIA).

This tool is the successor of the SIMATIC Selection Tool and combines the known configurators for automation technology into one tool.

With the TIA Selection Tool, you can generate a complete order list from your product selection or product configuration.

You can find the TIA Selection Tool on the Internet (<http://w3.siemens.com/mcms/topics/en/simatic/tia-selection-tool>).

## SIMATIC Automation Tool

You can use the SIMATIC Automation Tool to run commissioning and maintenance activities simultaneously on different SIMATIC S7 stations as a bulk operation, independently of the TIA Portal.

The SIMATIC automation tool provides a variety of functions:

- Scanning of a PROFINET/Ethernet plant network and identification of all connected CPUs
- Address assignment (IP, subnet, gateway) and station name (PROFINET device) to a CPU
- Transfer of the date and programming device/PC time converted to UTC time to the module
- Program download to CPU

- Operating mode switchover RUN/STOP
- CPU localization by means of LED flashing
- Reading out CPU error information
- Reading of CPU diagnostic buffer
- Reset to factory settings
- Updating the firmware of the CPU and connected modules

You can find the SIMATIC Automation Tool on the Internet (<https://support.industry.siemens.com/cs/ww/en/view/98161300>).

## PRONETA

With SIEMENS PRONETA (PROFINET network analysis), you analyze the plant network during commissioning. PRONETA features two core functions:

- The topology overview independently scans PROFINET and all connected components.
- The IO check is a fast test of the wiring and the module configuration of a plant.

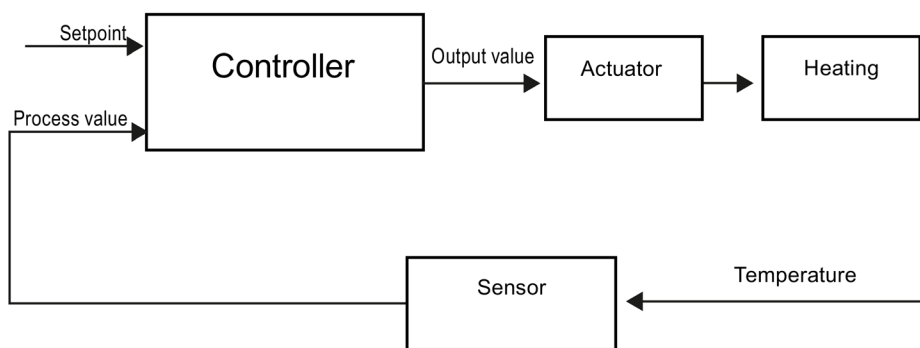
You can find SIEMENS PRONETA on the Internet (<https://support.industry.siemens.com/cs/ww/en/view/67460624>).

## Principles for control

### 2.1 Controlled system and actuators

#### Controlled system

Room temperature control by means of a heating system is a simple example of a controlled system. A sensor measures the room temperature and transfers the value to a controller. The controller compares the current room temperature with a setpoint and calculates an output value (manipulated variable) for heating control.



A properly set PID controller reaches this setpoint as quickly as possible and then holds it a constant value. After a change in the output value, the process value often changes only with a time delay. The controller has to compensate for this response.

#### Actuators

The actuator is an element of the controlled system and is influenced by the controller. Its function modifies mass and energy flows.

The table below provides an overview of actuator applications.

Application	Actuator
Liquid and gaseous mass flow	Valve, shutter, gate valve
Solid mass flow, e.g., bulk material	Articulated baffle, conveyor, vibrator channel
Flow of electrical power	Switching contact, contactor, relay, thyristor
	Variable resistor, variable transformer, transistor



Actuators are distinguished as follows:

- Proportional actuators with constant actuating signal

These elements set degrees of opening, angular positions or positions in proportion to the output value. The output value has an analog effect on the process within the control range.

Actuators in this group include spring-loaded pneumatic drives, as well as motorized drives with position feedback for which a position control system is formed.

An continuous controller, such as PID\_Compact, generates the output value.

- Proportional actuators with pulse-width modulated signal

These actuators are used to generate the output of pulses with a length proportional to the output value within the sampling time intervals. The actuator - e.g. a heating resistor or cooling apparatus - is switched on in isochronous mode for durations that differ depending on the output value.

The actuating signal can assume unipolar "On" or "Off" states, or represent bipolar states such as "open/close", "forward/backward", "accelerate/brake".

The output value is generated by a two-step controller such as PID\_Compact with pulse-width modulation.

- Actuators with integral action and three-step actuating signal

Actuators are frequently operated by motors with an on period that is proportional to the actuator travel of the choke element. This includes elements such as valves, shutters, and gate valves. In spite of their different design, all of these actuators follow the effect of an integral action at the input of the controlled system.

A step controller, such as PID\_3Step, generates the output value.

## 2.2 Controlled systems

The properties of a controlled system can hardly be influenced as these are determined by the technical requirements of the process and machinery. Acceptable control results can only be achieved by selecting a suitable controller type for the specific controlled system and adapting the controller to the time response of the controlled system. Therefore, it is indispensable for the configuration of the proportional, integral and derivative actions of the controller to have precise knowledge of the type and parameters of the controlled system.

### Controlled system types

Controlled systems are classified based on their time response to step changes of the output value.

We distinguish between the following controlled systems:

- Self-regulating controlled systems
  - Proportional-action controlled systems
  - PT1 controlled systems
  - PT2 controlled systems
- Non-self-regulating controlled systems
- Controlled systems with and without dead time

### Self-regulating controlled systems

#### Proportional-action controlled systems

In proportional-action controlled systems, the process value follows the output value almost immediately. The ratio between the process value and output value is defined by the proportional Gain of the controlled system.

Examples:

- Gate valve in a piping system
- Voltage dividers
- Step-down function in hydraulic systems

#### PT1 controlled systems

In a PT1 controlled system, the process value initially changes in proportion to the change of the output value. The rate of change of the process value is reduced as a function of the time until the end value is reached, i.e., it is delayed.

Examples:

- Spring damping system
- Charge of RC elements
- Water container that is heated with steam.

The time constants are often identical for heating and cooling processes, or for charging and discharge characteristics. With different time constants, controlling is clearly more complex.

### **PT2 controlled systems**

In a PT2 controlled system, the process value does not immediately follow a step change of the output value, i.e., it increases in proportion to the positive rate of rise and then approaches the setpoint at a decreasing rate of rise. The controlled system shows a proportional response characteristic with second order delay element.

Examples:

- Pressure control
- Flow rate control
- Temperature control

### **Non-self-regulating controlled systems**

Non-self-regulating controlled systems have an integral response. The process value approaches an infinite maximum value.

Example:

- Liquid flow into a container

### **Controlled systems with dead time**

A dead time always represents the runtime or transport time that has to expire before a change to the system input can be measured at the system output.

In controlled systems with dead time, the process value change is delayed by the amount of the dead time.

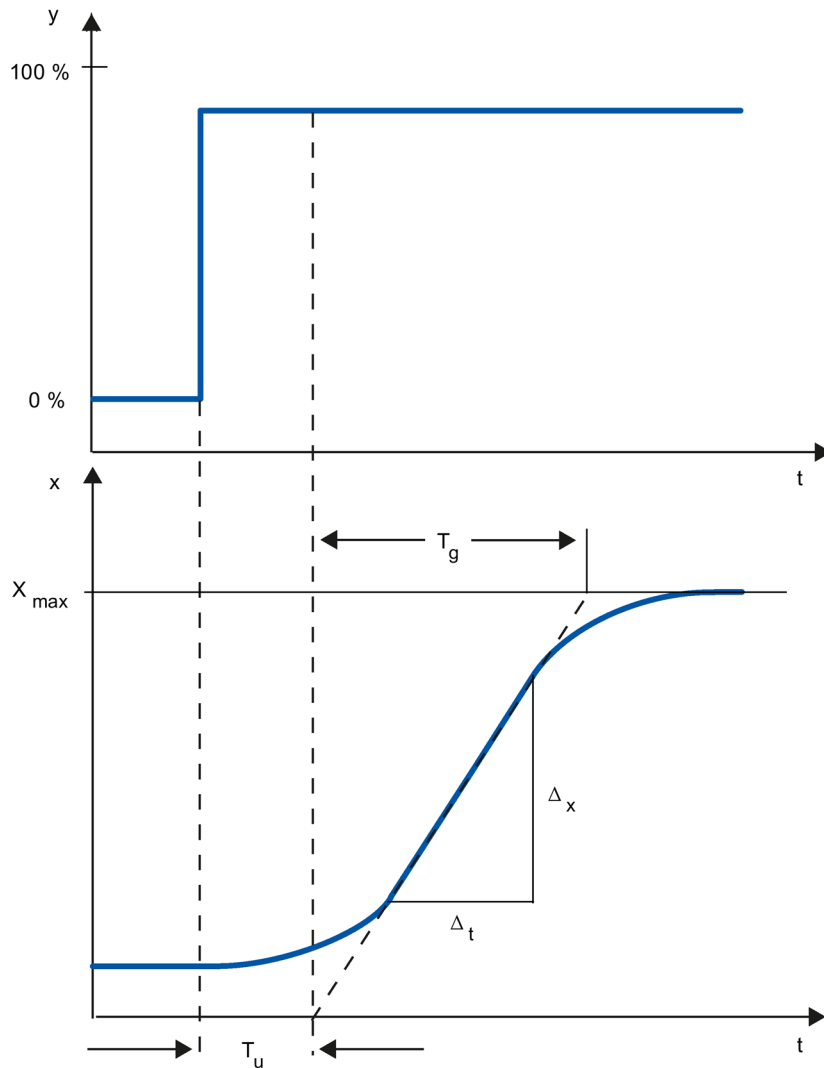
Example:

Conveyor

## 2.3 Characteristic values of the control section

### Determining the time response from the step response

Time response of the controlled system can be determined based on the time characteristic of process value  $x$  following a step change of output value  $y$ . Most controlled systems are self-regulating controlled systems.



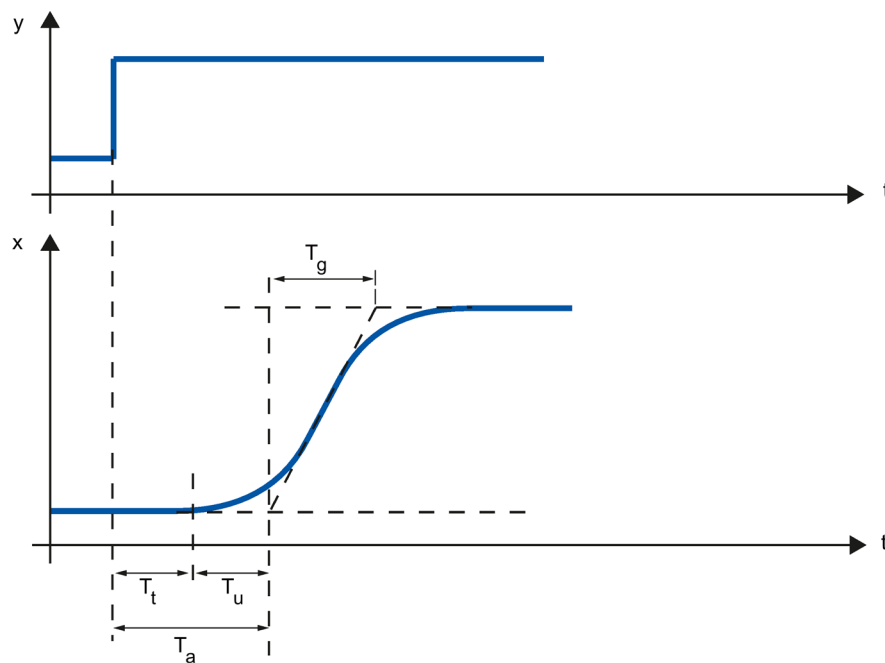
The time response can be determined by approximation using the variables Delay time  $T_u$ , Recovery time  $T_g$  and Maximum value  $X_{max}$ . The variables are determined by applying tangents to the maximum value and the inflection point of the step response. In many situations, it is not possible to record the response characteristic up to the maximum value because the process value cannot exceed specific values. In this case, the rate of rise  $v_{max}$  is used to identify the controlled system ( $v_{max} = \Delta x / \Delta t$ ).

The controllability of the controlled system can be estimated based on the ratio  $T_u/T_g$ , or  $T_u \times v_{\max}/X_{\max}$ . Rule:

Process type	$T_u / T_g$	Suitability of the controlled system for controlling
I	< 0,1	can be controlled well
II	0.1 to 0.3	can still be controlled
III	> 0,3	difficult to control

### Influence of the dead time on the controllability of a controlled system

A controlled system with dead time and recovery reacts as follows to a jump of the output value.



$T_t$	Dead time
$T_u$	Delay time
$T_g$	Recovery time
$y$	Output value
$x$	Process value

The controllability of a self-regulating controlled system with dead time is determined by the ratio of  $T_t$  to  $T_g$ .  $T_t$  must be small compared to  $T_g$ . Rule:

$$T_t/T_g \leq 1$$

**Response rate of controlled systems**

Controlled systems can be judged on the basis of the following values:

$T_u < 0.5 \text{ min}$ ,  $T_g < 5 \text{ min}$  = fast controlled system

$T_u > 0.5 \text{ min}$ ,  $T_g > 5 \text{ min}$  = slow controlled system

**Parameters of certain controlled systems**

Physical quantity	Controlled system	Delay time $T_u$	Recovery time $T_g$	Rate of rise $v_{max}$
Temperature	Small electrically heated furnace	0.5 to 1 min	5 to 15 min	Up to 60 K/min.
	Large electrically heated annealing furnace	1 to 5 min	10 to 20 min	Up to 20 K/min.
	Large gas-heated annealing furnace	0.2 to 5 min	3 to 60 min	1 to 30 K/min
	Distillation tower	1 to 7 min	40 to 60 min	0.1 to 0.5° C/s
	Autoclaves (2.5 m <sup>3</sup> )	0.5 to 0.7 min	10 to 20 min	Not specified
	High-pressure autoclaves	12 to 15 min	200 to 300 min	Not specified
	Steam superheater	30 s to 2.5 min	1 to 4 min	2° C/s
	Injection molding machines	0.5 to 3 min	3 to 30 min	5 to 20 K/min
	Extruders	1 to 6 min	5 to 60 min	
	Packaging machines	0.5 to 4 min	3 to 40 min	2 to 35 K/min
	Room heating	1 to 5 min	10 to 60 min	1° C/min
Flow rate	Pipeline with gas	0 to 5 s	0.2 to 10 s	Not relevant
	Pipeline with liquid	None	None	
Pressure	Gas pipeline	None	0.1 s	Not relevant
	Drum boiler with gas or oil firing	None	150 s	Not relevant
	Drum boiler with impact grinding mills	1 to 2 min	2 to 5 min	Not relevant
Vessel level	Drum boiler	0.6 to 1 min	Not specified	0.1 to 0.3 cm/s
Speed	Small electric drive	None	0.2 to 10 s	Not relevant
	Large electric drive	None	5 to 40 s	Not relevant
	Steam turbine	None	Not specified	50 min <sup>-1</sup>
Voltage	Small generators	None	1 to 5 s	Not relevant
	Large generators	None	5 to 10 s	Not relevant

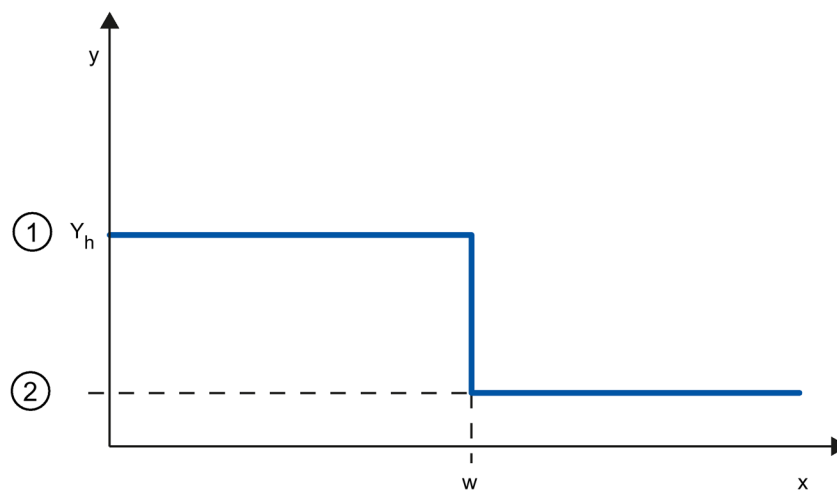
## 2.4 Pulse controller

### Two-step controllers without feedback

Two-step controllers have the state "ON" and "OFF" as the switching function. This corresponds to 100% or 0% output. This behavior generates a sustained oscillation of process value  $x$  around setpoint  $w$ .

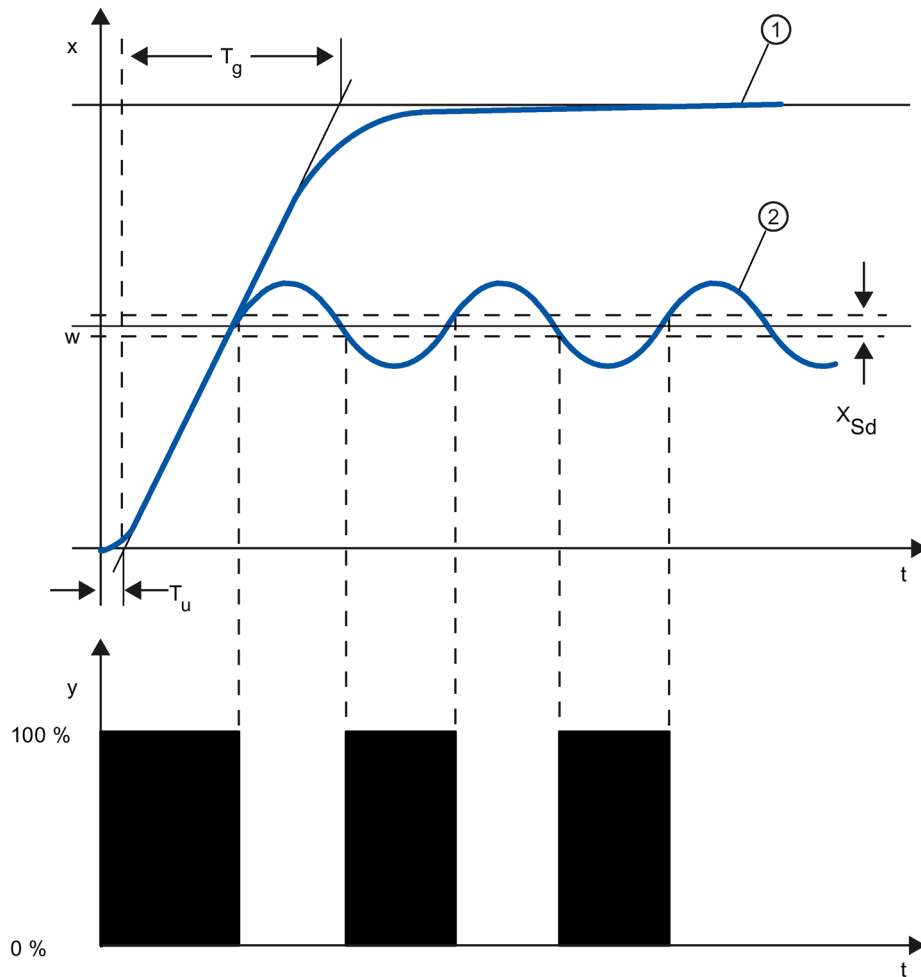
The amplitude and duration of the oscillation increase in proportion to the ratio between the delay time  $T_u$  and recovery time  $T_g$  of the controlled system. These controllers are used mainly for simple temperature control systems (such as electrically directly heated furnaces) or as limit-value signaling units.

The following diagram shows the characteristic of a two-step controller



- ① ON
- ② OFF
- $Y_h$  Control range
- $w$  Setpoint

The following diagram shows the control function of a two-step controller



- ① Response characteristic without controller
- ② Response characteristic with two-step controller
- $T_u$  Delay time
- $T_g$  Recovery time
- $x_{Sd}$  Switching difference



### **Two-step controllers with feedback**

The behavior of two-step controllers in the case of controlled systems with larger delay times, such as furnaces where the functional space is separated from the heating, can be improved by the use of electronic feedback.

The feedback is used to increase the switching frequency of the controller, which reduces the amplitude of the process value. In addition, the control-action results can be improved substantially in dynamic operation. The limit for the switching frequency is set by the output level. It should not exceed 1 to 5 switches per minute at mechanical actuators, such as relays and contactors. In the case of voltage and current outputs with downstream thyristor or Triac controllers high switching frequencies can be selected that exceed the limit frequency of the controlled system by far.

Since the switching pulses can no longer be determined at the output of the controlled system, results comparable with those of continuous controllers are obtained.

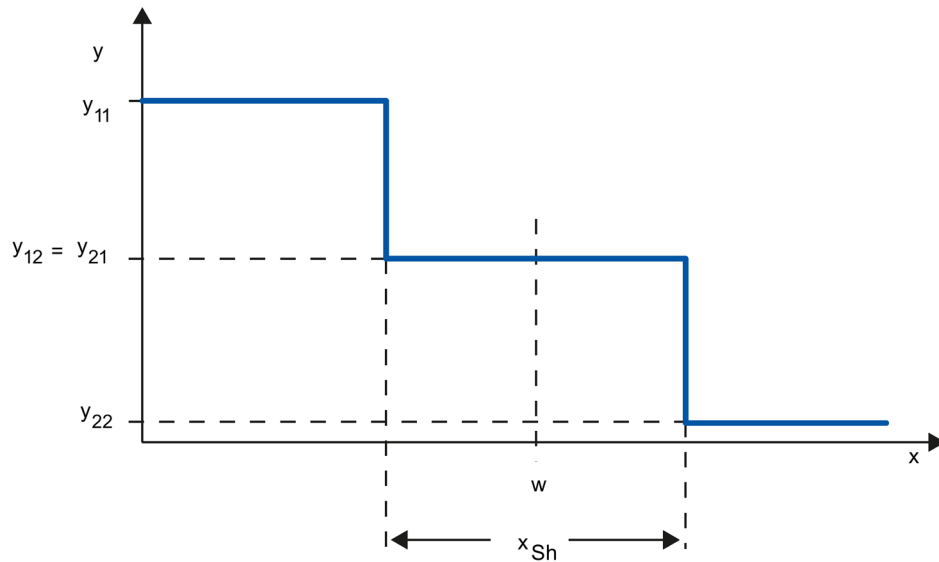
The output value is generated by pulse-width modulation of the output value of a continuous controller.

Two-step controllers with feedback are used for temperature control in furnaces, at processing machines in the plastics, textile, paper, rubber and foodstuff industries as well as for heating and cooling devices.

### Three-step controllers

Three-step controllers are used for heating / cooling. These controllers have two switching points as their output. The control-action results are optimized through electronic feedback structures. Fields of applications for such controllers are heating, low-temperature, climatic chambers and tool heating units for plastic-processing machines.

The following diagram shows the characteristic of a three-step controller

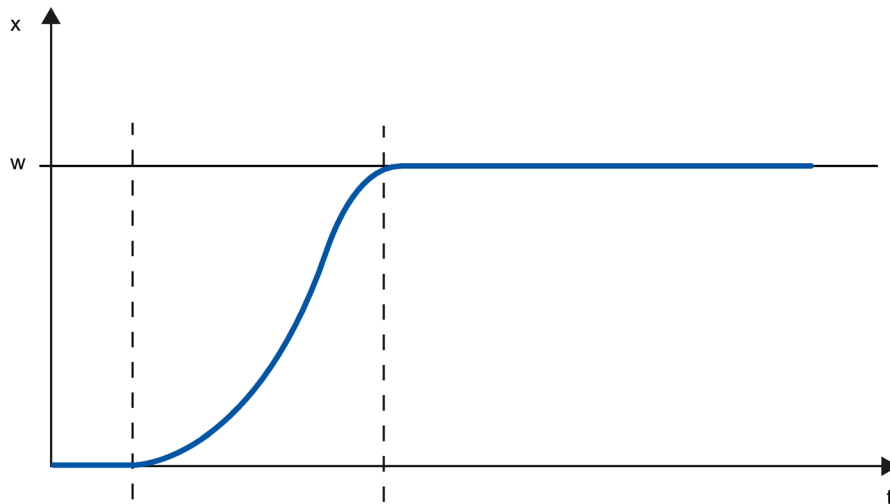


- y      Output value, e.g.
  - y11 = 100% heating
  - y12 = 0% heating
  - y21 = 0% cooling
  - y22 = 100% cooling
- x      Physical quantity of the process value, e.g., temperature in° C
- w      Setpoint
- xSh    Distance between Switching Point 1 and Switching Point 2

## 2.5 Response to setpoint changes and disturbances

### Response to setpoint changes

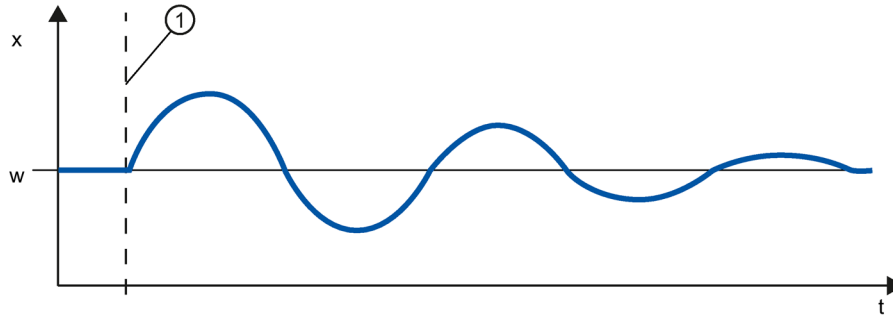
The process value should follow a setpoint change as quickly as possible. The response to setpoint changes is improved by minimizing fluctuation of the process value and the time required to reach the new setpoint.



$x$	Process value
$w$	Setpoint

**Response to disturbances**

The setpoint is influenced by disturbance variables. The controller has to eliminate the resulting control deviations in the shortest time possible. The response to disturbances is improved by minimizing fluctuation of the process value and the time required to reach the new setpoint.



x	Process value
w	Setpoint
①	Influencing a disturbance variable

Disturbance variables are corrected by a controller with integral action. A persistent disturbance variable does not reduce control quality because the control deviation is relatively constant. Dynamic disturbance variables have a more significant impact on control quality because of control deviation fluctuation. The control deviation is eliminated again only by means of the slow acting integral action.

A measurable disturbance variable can be included in the controlled system. This inclusion would significantly accelerated the response of the controller.

## 2.6 Control Response at Different Feedback Structures

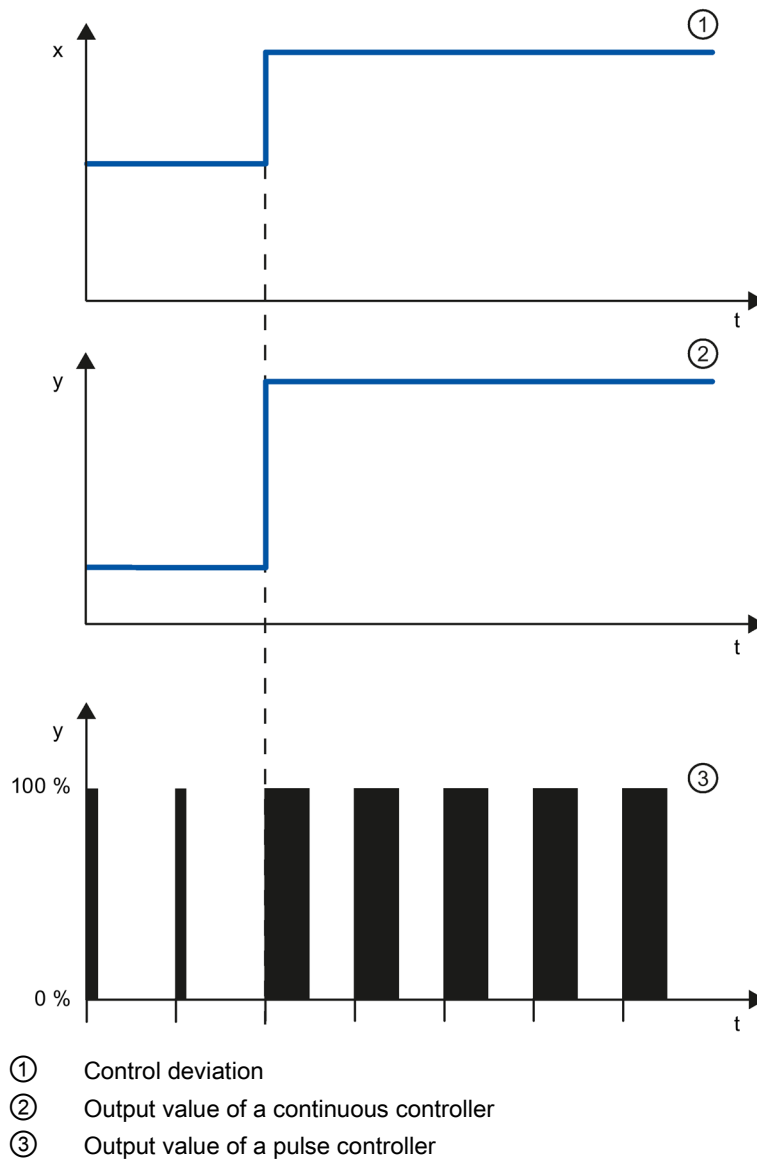
### Control behavior of controllers

A precise adaptation of the controller to the time response of the controlled system is decisive for the controller's precise settling to the setpoint and optimum response to disturbance variables.

The feedback circuit can have a proportional action (P), proportional-derivative action (PD), proportional-integral action (PI), or proportional-integral-derivative action (PID).

If step functions are to be triggered by control deviations, the step responses of the controllers differ depending on their type.

### Step response of a proportional action controller



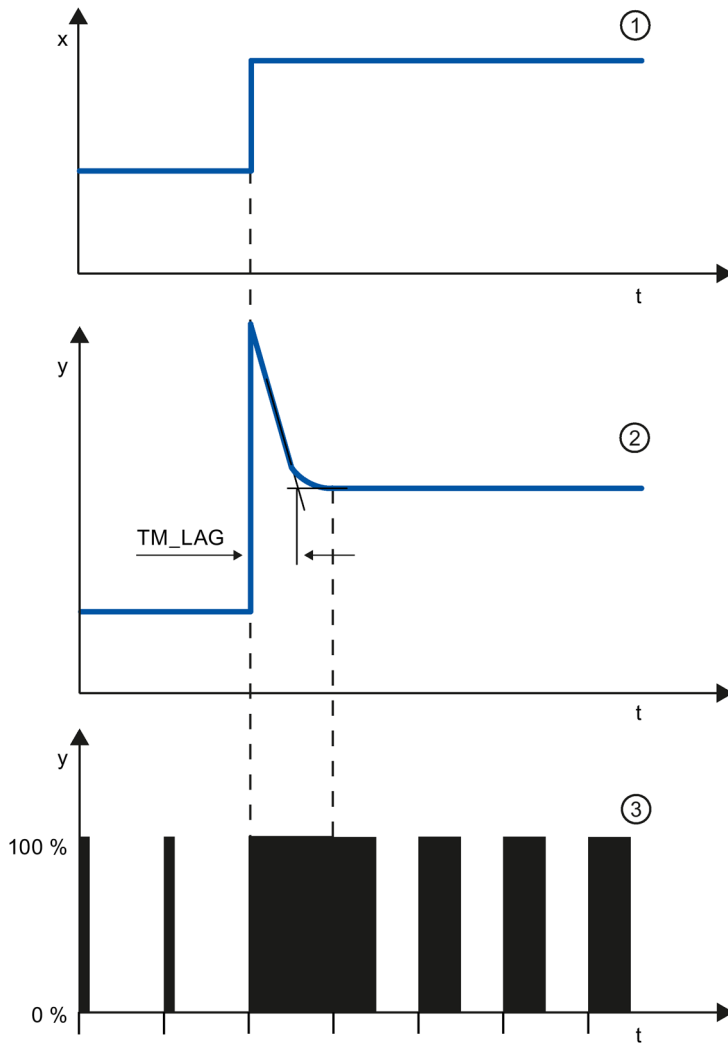
**Equation for proportional action controller**

Output value and control deviation are directly proportional, meaning:

Output value = proportional gain × control deviation

$$y = \text{GAIN} \times x$$

**Step response of a PD-action controller**



- ① Control deviation
- ② Output value of a continuous controller
- ③ Output value of a pulse controller
- TM\_LAG Delay of the Derivative action

**Equation for PD-action controller**

The following applies for the step response of the PD-action controller in the time range:

$$y = \text{GAIN} \cdot X_W \cdot \left( 1 + \frac{\text{TD}}{\text{TM\_LAG}} \cdot e^{-\frac{t}{\text{TM\_LAG}}} \right)$$

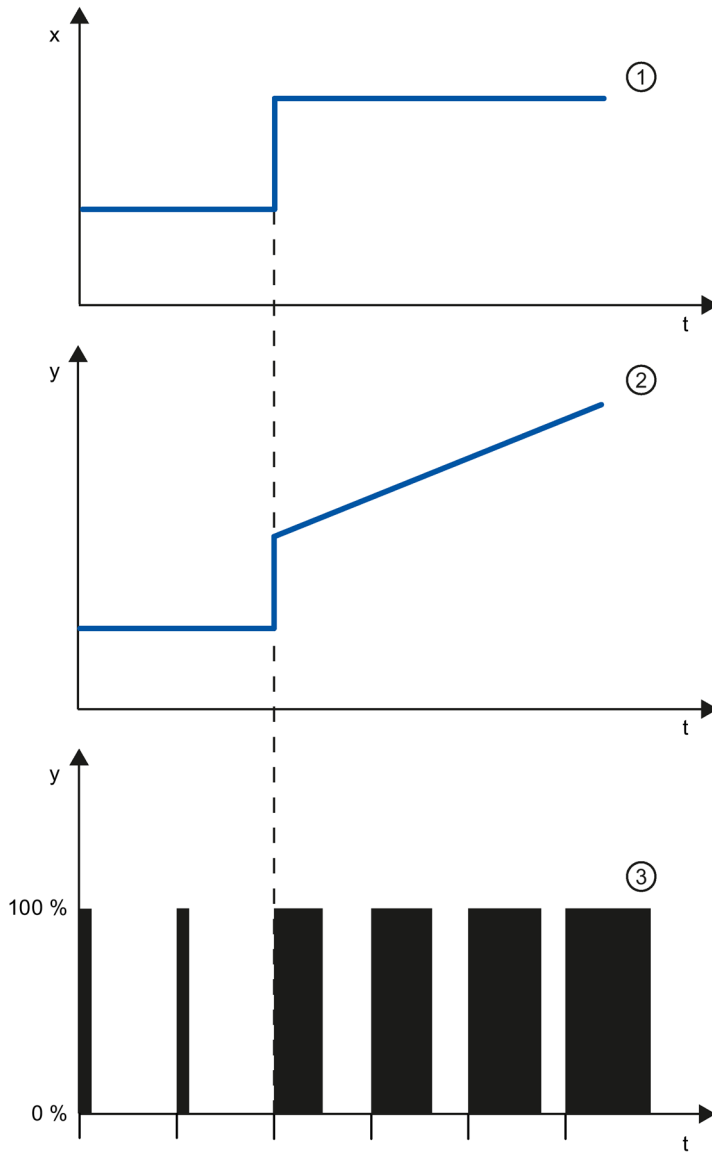
t = time interval since the step of the control deviation

The derivative action generates a output value as a function of the rate of change of the process value. A derivative action by itself is not suitable for controlling because the output value only follows a step of the process value. As long as the process value remains constant, the output value will no longer change.

The response to disturbances of the derivative action is improved in combination with a proportional action. Disturbances are not corrected completely. The good dynamic response is advantageous. A well attenuated, non-oscillating response is achieved during approach and setpoint change.

A controller with derivative action is not appropriate if a controlled system has pulsing measured quantities, for example, in the case of pressure or flow control systems.

### Step response of a PI-action controller



- ① Control deviation
- ② Output value of a continuous controller
- ③ Output value of a pulse controller

An integral action in the controller adds the control deviation as a function of the time. This means that the controller corrects the system until the control deviation is eliminated. A sustained control deviation is generated at controllers with proportional action only. This effect can be eliminated by means of an integral action in the controller.

In practical experience, a combination of the proportional, integral and derivative actions is ideal, depending on the requirements placed on the control response. The time response of the individual components can be described by the controller parameters proportional gain GAIN, integral action time TI (integral action), and derivative action time TD (derivative action).



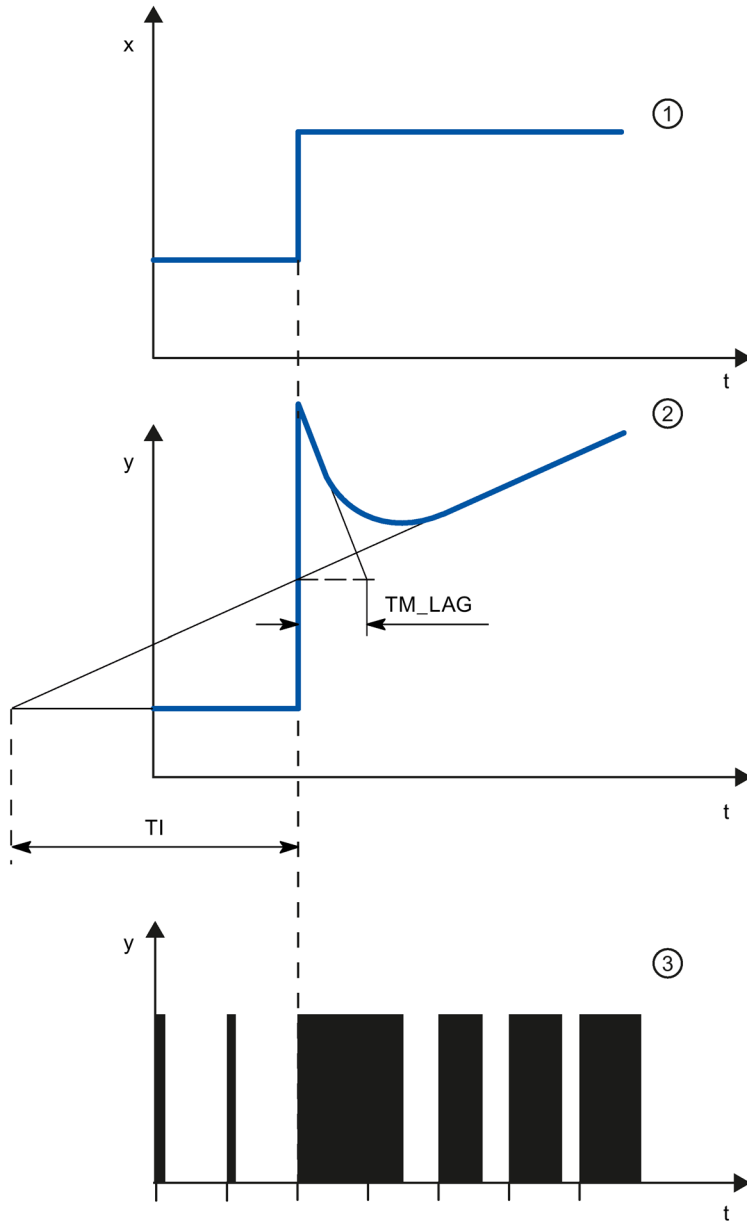
**Equation for PI-action controller**

The following applies for the step response of the PI-action controller in the time range:

$$y = \text{GAIN} \cdot X_W \cdot \left( 1 + \frac{1}{\text{TI} \cdot t} \right)$$

t = time interval since the step of the control deviation

### Step response of a PID controller



- ① Control deviation
- ② Output value of a continuous controller
- ③ Output value of a pulse controller
- TM\_LAG Delay of the Derivative action
- T<sub>i</sub> Integral action time

**Equation for PID controller**

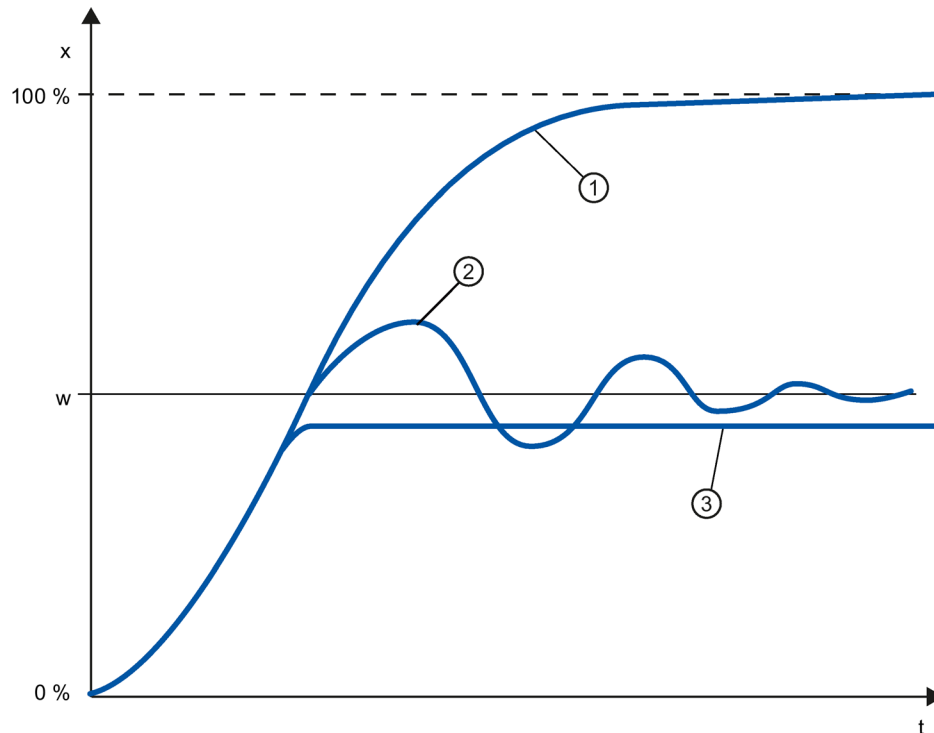
The following applies for the step response of the PID controller in the time range:

$$y = \text{GAIN} \cdot X_w \cdot \left( 1 + \frac{1}{\text{TI} \cdot t} + \frac{\text{TD}}{\text{TM\_LAG}} \cdot e^{-\frac{t}{\text{TM\_LAG}}} \right)$$

t = time interval since the step of the control deviation

**Response of a controlled system with different controller structures**

Most of the controller systems occurring in process engineering can be controlled by means of a controller with PI-action response. In the case of slow controlled system with a large dead time, for example temperature control systems, the control result can be improved by means of a controller with PID action.



- |   |                      |
|---|----------------------|
| ① | No controller        |
| ② | PID controller       |
| ③ | PD-action controller |
| w | Setpoint             |
| x | Process value        |





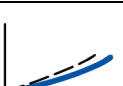
Controllers with PI and PID action have the advantage that the process value does not have any deviation from the setpoint value after settling. The process value oscillates over the setpoint during approach.

## 2.7 Selection of the controller structure for specified controlled systems

### Selection of the Suitable Controller Structures

To achieve optimum control results, select a controller structure that is suitable for the controlled system and that you can adapt to the controlled system within specific limits.

The table below provides an overview of suitable combinations of a controller structure and controlled system.

Controlled system		Controller structure			
		P	PD	PI	PID
	With dead time only	Unsuitable	Unsuitable	Suitable	Unsuitable
	PT1 with dead time	Unsuitable	Unsuitable	Well suited	Well suited
	PT2 with dead time	Unsuitable	Suited conditionally	Well suited	Well suited
	Higher order	Unsuitable	Unsuitable	Suited conditionally	Well suited
	Not self-regulating	Well suited	Well suited	Well suited	Well suited

The table below provides an overview of suitable combinations of a controller structure and physical quantity.

Physical quantity	Controller structure			
	P	PD	PI	PID
	Sustained control deviation		No sustained control deviation	
Temperature	For low performance requirements and proportional action controlled systems with $T_u/T_g < 0,1$	Well suited	The most suitable controller structures for high performance requirements (except for specially adapted special controllers)	
Pressure	Suitable, if the delay time is inconsiderable	Unsuitable	The most suitable controller structures for high performance requirements (except for specially adapted special controllers)	
Flow rate	Unsuitable, because required GAIN range is usually too large	Unsuitable	Suitable, but integral action controller alone often better	Hardly required

## 2.8 PID parameter settings

### Rule of Thumb for the Parameter Setting

Controller structure	Setting
P	GAIN $\approx v_{\max} \times T_u [^{\circ} \text{C}]$
PI	GAIN $\approx 1.2 \times v_{\max} \times T_u [^{\circ} \text{C}]$ TI $\approx 4 \times T_u [ \text{min} ]$
PD	GAIN $\approx 0.83 \times v_{\max} \times T_u [^{\circ} \text{C}]$ TD $\approx 0.25 \times v_{\max} \times T_u [ \text{min} ]$ TM_LAG $\approx 0.5 \times TD [ \text{min} ]$
PID	GAIN $\approx 0.83 \times v_{\max} \times T_u [^{\circ} \text{C}]$ TI $\approx 2 \times T_u [ \text{min} ]$ TD $\approx 0.4 \times T_u [ \text{min} ]$ TM_LAG $\approx 0.5 \times TD [ \text{min} ]$
PD/PID	GAIN $\approx 0.4 \times v_{\max} \times T_u [^{\circ} \text{C}]$ TI $\approx 2 \times T_u [ \text{min} ]$ TD $\approx 0.4 \times T_u [ \text{min} ]$ TM_LAG $\approx 0.5 \times TD [ \text{min} ]$

Instead of  $v_{\max} = \Delta_x / \Delta_t$ , you can use  $X_{\max} / T_g$ .

In the case of controllers with PID structure the setting of the integral action time and differential-action time is usually coupled with each other.

The ratio TI / TD lies between 4 and 5 and is optimal for most controlled systems.

Non-observance of the differential-action time TD is uncritical at PD controllers.

In the case of PI and PID controllers, control oscillations occur if the integral action time TI has been select by more than half too small.

An integral action time that is too large slows down the settling times of disturbances. One cannot expect that the control loops operate "optimally" after the first parameter settings. Experience shows that adjusting is always necessary, when a system exists that is "difficult to control" with  $T_u / T_g > 0.3$ .

## Configuring a software controller

### 3.1 Overview of software controller

For the configuration of a software controller, you need an instruction with the control algorithm and a technology object. The technology object for a software controller corresponds with the instance DB of the instruction. The configuration of the controller is saved in the technology object. In contrast to the instance DBs of other instructions, technology objects are not stored for the program resources, but rather under CPU > Technology objects.

#### Technology objects and instructions

CPU	Library	Instruction	Technology object	Description
S7-1200	Compact PID	PID_Compact V1.x	PID_Compact V1.x	Universal PID controller with integrated tuning
S7-1200		PID_3Step V1.x	PID_3Step V1.x	PID controller with integrated tuning for valves
S7-1500 S7-1200 V4.x		PID_Compact V2.x	PID_Compact V2.x	Universal PID controller with integrated tuning
S7-1500 S7-1200 V4.x		PID_3Step V2.x	PID_3Step V2.x	PID controller with integrated tuning for valves
S7-1500 ≥ V1.7 S7-1200 ≥ V4.1		PID_Temp V1.x	PID_Temp V1.x	Universal PID temperature controller with integrated tuning
S7-1500/300/400	PID basic functions	CONT_C	CONT_C	Continuous controller
S7-1500/300/400		CONT_S	CONT_S	Step controller for actuators with integrating behavior
S7-1500/300/400		PULSEGEN	-	Pulse generator for actuators with proportional behavior
S7-1500/300/400		TCONT_CP	TCONT_CP	Continuous temperature controller with pulse generator
S7-1500/300/400		TCONT_S	TCONT_S	Temperature controller for actuators with integrating behavior
S7-300/400	PID Self Tuner	TUN_EC	TUN_EC	Optimization of a continuous controller
S7-300/400		TUN_ES	TUN_ES	Optimization of a step controller

CPU	Library	Instruction	Technology object	Description
S7-300/400	Standard PID Control (PID Professional optional package)	PID_CP	PID_CP	Continuous controller with pulse generator
S7-300/400		PID_ES	PID_ES	Step controller for actuators with integrating behavior
S7-300/400		LP_SCHED	-	Distribute controller calls
S7-300/400	Modular PID Control (PID Professional optional package)	A_DEAD_B	-	Filter interfering signal from control deviation
S7-300/400		CRP_IN	-	Scale analog input signal
S7-300/400		CRP_OUT	-	Scale analog output signal
S7-300/400		DEAD_T	-	Delay output of input signal
S7-300/400		DEADBAND	-	Suppress small fluctuations to the process value
S7-300/400		DIF	-	Differentiate input signals over time
S7-300/400		ERR_MON	-	Monitor control deviation
S7-300/400		INTEG	-	Integrate input signals over time
S7-300/400		LAG1ST	-	First-order delay element
S7-300/400		LAG2ND	-	Second-order delay element
S7-300/400		LIMALARM	-	Report limit values
S7-300/400		LIMITER	-	Limiting the manipulated variable
S7-300/400		LMNGEN_C	-	Determine manipulated variable for continuous controller
S7-300/400		LMNGEN_S	-	Determine manipulated variable for step controller
S7-300/400		NONLIN	-	Linearize encoder signal
S7-300/400		NORM	-	Scale process value physically
S7-300/400		OVERRIDE	-	Switch manipulated variable from 2 PID controllers to 1 actuator
S7-300/400		PARA_CTL	-	Switch parameter sets
S7-300/400		PID	-	PID algorithm
S7-300/400		PUSLEGEN_M	-	Generate pulse for proportional actuators
S7-300/400		RMP_SOAK	-	Specify setpoint according to ramp / soak
S7-300/400		ROC_LIM	-	Limit rate of change
S7-300/400		SCALE_M	-	Scale process value
S7-300/400	SP_GEN	-	Specify setpoint manually	
S7-300/400	SPLT_RAN	-	Split manipulated variable range	
S7-300/400	SWITCH	-	Switch analog values	
S7-300/400	LP_SCHED_M	-	Distribute controller calls	

### 3.2 Steps for the configuration of a software controller

All SW-controllers are configured according to the same scheme:

Step	Description
1	Add technology object (Page 40)
2	Configure technology object (Page 45)
3	Call instruction in the user program (Page 46)
4	Download technology object to device (Page 72)
5	Commission software controller (Page 74)
6	Save optimized PID parameters in the project (Page 74)
7	Comparing values (Page 43)
8	Display instances of a technology object (Page 75)

### 3.3 Add technology objects

#### Add technology object in the project navigator

When a technology object is added, an instance DB is created for the instruction of this technology object. The configuration of the technology object is stored in this instance DB.

#### Requirement

A project with a CPU has been created.

#### Procedure

To add a technology object, proceed as follows:

1. Open the CPU folder in the project tree.
2. Open the "Technology objects" folder.
3. Double-click "Add new object".  
The "Add new object" dialog box opens.
4. Click on the "PID" button.  
All available PID-controllers for this CPU are displayed.
5. Select the instruction for the technology object, for example, PID\_Compact.
6. Enter an individual name for the technology object in the "Name" input field.
7. Select the "Manual" option if you want to change the suggested data block number of the instance DB.
8. Click "Further information" if you want to add own information to the technology object.
9. Confirm with "OK".



## **Result**

The new technology object has been created and stored in the project tree in the "Technology objects" folder. The technology object is used if the instruction for this technology object is called in a cyclic interrupt OB.

---

### **Note**

You can select the "Add new and open" check box at the bottom of the dialog box. This opens the configuration of the technology object after adding has been completed.

---

## 3.4 Comparing values








### 3.4.1 Comparison display and boundary conditions

The "Compare values" function provides the following options:

- Comparison of configured start values of the project with the start values in the CPU and the actual values
- Direct editing of actual values and the start values of the project
- Immediate detection and display of input errors with suggested corrections
- Backup of actual values in the project
- Transfer of start values of the project to the CPU as actual values

#### Icons and operator controls

The following icons and operator controls are available:

Icon	Function
	Start value PLC matches the configured Start value project
	Start value PLC does not match the configured Start value project
	The comparison of the Start value PLC with the configured Start value project cannot be performed
	At least one of the two comparison values has a process-related or syntax error.
	Transfers actual values to the offline project
	Transfers updated start values in the project to the CPU (initialize setting values)
	Opens the "Compare values" dialog

#### Boundary conditions

The "Compare values" function is available for S7-1200 and S7-1500 without limitations.

The following limitation applies to S7-300 and S7-400:

In monitoring mode, an S7-300/S7-400 cannot transfer the start values to the CPU. These values cannot be displayed online with "Compare values".

The actual values of the technology object are displayed and can be changed directly.


### 3.4.2 Comparing values

The procedure is shown in the following using "PID Parameters" as an example.


#### Requirements

- A project with a software controller is configured.
- The project is downloaded to the CPU.
- The configuration dialog is open in the project navigator.

#### Procedure

1. Open the desired software controller in the project navigation.
2. Double-click the "Configuration" object.
3. Navigate within the configuration window to the "PID Parameters" dialog.
4. Click the  icon to activate monitoring mode.

The icons and operator controls (Page 42) of the "Compare values" function are shown behind the parameters.

5. Click the desired parameter in the input box and change the parameter values manually by entering them directly.
  - If the background of the input box is gray, this value is a read-only value and cannot be changed.
  - To change the values in the "PID Parameters" dialog, enable manual entry by selecting the "Enable manual entry" check box beforehand.
6. Click the  icon to open the dialog for the start values.

This dialog indicates two values of the parameter:

  - Start value in CPU: The start value in the CPU is shown in the top part.
  - Start value in the project: The configured start value in the project is shown in the bottom part.
7. Enter the desired value in the input box for the project.

### Error detection

The input of incorrect values is detected. Corrections are suggested in this case.


If you enter a value with incorrect syntax, a rollout containing the corresponding error message opens below the parameter. The incorrect value is not applied.

If you enter a value that is incorrect for the process, a dialog opens containing the error message and a suggested correction:


- Click "No" to accept this suggested correction and correct your input.
- Click "OK" to apply the incorrect value.


<b>NOTICE</b>
<b>Malfunctions of the controller</b>
Values incorrect for the process can result in controller malfunctions.

### Backing up actual values

Click the  icon to transfer the actual controller values to the start values of your configured project.

### Transferring project values to the CPU

Click the  icon to transfer the configured values of your project to the CPU.

 <b>CAUTION</b>
<b>Prevent personal injury and property damage!</b>
Downloading and resetting of the user program while the plant is operating may result in significant property damage and severe personal injuries in the event of malfunctions or program errors.
Make sure that dangerous states cannot occur before you download and reset the user program.

## 3.5 Configure technology objects

The properties of a technology object on a S7-1200 CPU can be configured in two ways.

- In the Inspector window of the programming editor
- In the configuration editor

The properties of a technology object on a S7-300/400 CPU can only be configured in the configuration editor.

### Inspector window of the programming editor

In the Inspector window of the programming editor you can only configure the parameters required for operation.

The offline values of the parameters are also shown in online mode. You can only change the online values in the commissioning window.

To open the Inspector window of the technology object, follow these steps:

1. Open the "Program blocks" folder in the project tree.
2. Double click the block (cyclic interrupt OB) in which you open the instruction of the SW-controller.  
The block is opened in the work area.
3. Click on the instruction of the SW-controller.
4. In the Inspector window, select the "Properties" and "Configuration" tabs consecutively.

### Configuration window




For each technology object, there is a specific configuration window in which you can configure all properties.

To open the configuration window of the technology object, follow these steps:

1. Open the "Technology objects" folder in the project tree.
2. Open the technology object in the project tree.
3. Double-click the "Configuration" object.

### Symbols

Icons in the area navigation of the configuration and in the Inspector window show additional details about the completeness of the configuration:

	<b>The configuration contains default values and is complete.</b> The configuration exclusively contains default values. With these default values the use of the technology object is possible without further changes.
	<b>The configuration contains user-defined or automatically adjusted values and is complete</b> All input fields of the configuration contain valid values and at least one default setting was changed.
	<b>The configuration is incomplete or faulty</b> At least one input field or a collapsible list contains no or one invalid value. The corresponding field or the drop-down list box has a red background. When clicked, the roll-out error message indicates the cause of the error.

The properties of a technology object are described in detail in the section for the technology object.

## 3.6 Call instruction in the user program

The instruction of the software controller must be called in a cyclic interrupt OB. The sampling time of the software controller is determined by the interval between the calls in the cyclic interrupt OB.

### Requirement

The cyclic interrupt OB is created and the cycle time of the cyclic interrupt OB is correctly configured.

### Procedure

Proceed as follows to call the instruction in the user program:

1. Open the CPU folder in the project tree.
2. Open the "Program blocks" folder.
3. Double-click the cyclic interrupt OB.  
The block is opened in the work area.
4. Open the "Technology" group in the "Instructions" window and the "PID Control" folder.  
The folder contains all instructions for software controllers that can be configured on the CPU.
5. Select the instruction and drag it to your cyclic interrupt OB.  
The "Call options" dialog box opens.
6. Select a technology object or type the name for a new technology object from the "Name" list.

### Result

If the technology object does not exist yet, it is added. The instruction is added in the cyclic interrupt OB. The technology object is assigned to this call of the instruction.

## 3.7 Parameter view

### 3.7.1 Introduction to the parameter view

The Parameter view provides you with a general overview of all relevant parameters of a technology object. You obtain an overview of the parameter settings and can easily change them in offline and online mode.

The screenshot shows the 'Parameter view' tab of a software controller configuration tool. The interface includes a toolbar (2) at the top left, a navigation pane (3) on the left, and a main table (4) displaying parameter details. The table columns are: Name in functional view, Name in DB, Start value project, Data type, and Comment. The parameters listed include 'Invert the control logic', 'Enable last mode after CPU ..', 'Physical quantity', 'Unit of measurement', 'Set Mode to', 'Selection Input', 'Process value high limit', 'Process value low limit', 'Scaled high process value', 'Scaled low process value', 'Input\_PER low', 'Input\_PER high', 'Warning low limit', 'Warning high limit', 'Minimum OFF time', 'Proportional gain', 'Integral action time', and 'Derivative action time'.

Name in functional view	Name in DB	Start value project	Data type	Comment
Invert the control logic	../InvertControl	FALSE	Bool	Enables inversion of control log
Enable last mode after CPU ..	RunModeBySta...	TRUE	Bool	Activates the operating mode s
Physical quantity	PhysicalQuantity	General	Int	Selection of physical quantity.
Unit of measurement	PhysicalUnit	%	Int	Selection of unit of measureme
Set Mode to	Mode	Manual mode	Int	Selection of operating mode.
Selection Input	../InputPerOn	Input_PER (analog)	Bool	Selection of process value.
Process value high limit	../InputUpperLi...	120.0	Real	Entry for process value high lim
Process value low limit	../InputLowerLi...	0.0	Real	Entry for process value low limit
Scaled high process value	../UpperPointOut	100.0	Real	Entry for scaled high process va
Scaled low process value	../LowerPointOut	0.0	Real	Entry for scaled low process val
Input_PER low	../LowerPointIn	0	Real	Entry for low value of Input_PER.
Input_PER high	../UpperPointIn	27648	Real	Entry for high value of Input_PEF
Warning low limit	../InputLowerW...	-3.402822e+38	Real	Entry for warning low limit.
Warning high limit	../InputUpperW...	3.402822e+38	Real	Entry for warning high limit.
Minimum OFF time	../MinimumOff...	0.0	Real	Entry for minimum OFF time.
Proportional gain	../Gain	1.0	Real	Entry for proportional gain.
Integral action time	../Ti	20.0	Real	Entry for integral action time.
Derivative action time	../Td	0.0	Real	Entry for derivative action time.

- ① "Parameter view" tab
- ② Toolbar (Page 49)
- ③ Navigation (Page 50)
- ④ Parameter table (Page 51)

## Function scope

The following functions are available for analyzing the parameters of the technology objects and for enabling targeted monitoring and modification.

Display functions:

- Display of parameter values in offline and online mode
- Display of status information of the parameters
- Display of value deviations and option for direct correction
- Display of configuration errors
- Display of value changes as a result of parameter dependencies
- Display of all memory values of a parameter: Start value PLC, Start value project, Monitor value
- Display of the parameter comparison of the memory values of a parameter

Operator control functions:

- Navigation for quickly changing between the parameters and parameter structures.
- Text filter for faster searches for particular parameters.
- Sorting function for customizing the order of parameters and parameter groups to requirements.
- Memory function for backing up structural settings of the Parameter view.
- Monitoring and modifying of parameter values online.
- Function for saving a snapshot of parameter values of the CPU in order to capture momentary situations and to respond to them.
- Function for applying a snapshot of parameter values as start values.
- Download of modified start values to the CPU.
- Comparison functions for comparing parameter values with one another.

## Validity

The Parameter view described here is available for the following technology objects:






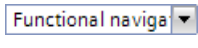



- PID\_Compact
- PID\_3Step
- PID\_Temp
- CONT\_C (S7-1500 only)
- CONT\_S (S7-1500 only)
- TCONT\_CP (S7-1500 only)
- TCONT\_S (S7-1500 only)
- TO\_Axis\_PTO (S7-1200 Motion Control)
- TO\_Positioning\_Axis (S7-1200 Motion Control)
- TO\_CommandTable\_PTO (S7-1200 Motion Control)
- TO\_CommandTable (S7-1200 Motion Control)



## 3.7.2 Structure of the parameter view

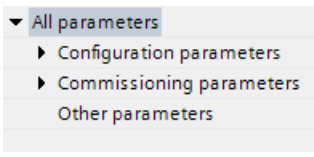
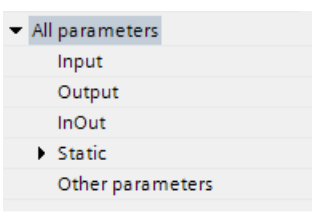
### 3.7.2.1 Toolbar

The following functions can be selected in the toolbar of the parameter view.

Icon	Function	Explanation
	Monitor all	Starts the monitoring of visible parameters in the active Parameter view (online mode).
	Create snapshot of monitor values and accept setpoints of this snapshot as start values	Applies the current monitor values to the "Snapshot" column and updates the start values in the project. Only in online mode for PID_Compact, PID_3Step and PID_Temp.
	Initialize setpoints	Transfers the start values updated in the project to the CPU. Only in online mode for PID_Compact, PID_3Step and PID_Temp.
	Create snapshot of monitor values	Applies the current monitor values to the "Snapshot" column. Only in online mode.
	Modify all selected parameters immediately and once	This command is executed once and as quickly as possible without reference to any particular point in the user program. Only in online mode.
	Select navigation structure	Toggles between functional navigation and data navigation.
	Text filter...	After entry of a character string: Display of all parameters containing the specified string in one of the currently visible columns.
	Selection of compare values	Selection of parameter values that are to be compared with one another in online mode (Start value project, Start value PLC, Snapshot) Only in online mode.
	Save window settings	Saves your display settings for the Parameter view (e.g., selected navigation structure, activated table columns, etc.)

### 3.7.2.2 Navigation

Within the "Parameter view" tab, the following alternative navigation structures can be selected.




Navigation	Explanation
Functional navigation  A screenshot of a navigation menu with a dropdown arrow next to "All parameters". The menu items are: "All parameters" (highlighted), "Configuration parameters", "Commissioning parameters", and "Other parameters".	In the functional navigation, the structure of the parameters is based on the structure in the configuration dialog ("Functional view" tab), commissioning dialog, and diagnostics dialog. The last group "Other parameters" contains all other parameters of the technology object.
Data navigation  A screenshot of a navigation menu with a dropdown arrow next to "All parameters". The menu items are: "All parameters" (highlighted), "Input", "Output", "InOut", "Static", and "Other parameters".	In the data navigation, the structure of the parameters is based on the structure in the instance DB / technology DB. The last group "Other parameters" contains the parameters that are not contained in the instance DB / technology DB.

You can use the "Select navigation structure" drop-down list to toggle the navigation structure.


### 3.7.2.3 Parameter table

The table below shows the meaning of the individual columns of the parameter table. You can show or hide the columns as required.

- Column "Offline" = X: Column is visible in offline mode.
- Column "Online" = X: Column is visible in online mode (online connection to the CPU).

Column	Explanation	Offline	Online
Name in functional view	Name of the parameter in the functional view. The display field is empty for parameters that are not configured via the technology object.	X	X
Full name in DB	Complete path of the parameter in the instance DB / technology DB. The display field is empty for parameters that are not contained in the instance DB / technology DB.	X	X
Name in DB	Name of the parameter in the instance DB / technology DB. If the parameter is part of a structure or UDT, the prefix ". ." is added. The display field is empty for parameters that are not contained in the instance DB / technology DB.	X	X
Status of configuration	Display of the completeness of the configuration using status symbols. see Status of configuration (offline) (Page 63)	X	
Compare result	Result of the "Compare values" function. This column is shown if there is an online connection and the "Monitor all" button  is selected.		X
Start value project	Configured start value in the project. Error indication if entered values have a syntax or process-related error.	X	X
Default value	Value that is pre-assigned to the parameter. The display field is empty for parameters that are not contained in the instance DB / technology DB.	X	X
Snapshot	Snapshot of the current values in the CPU (monitor values). Error indication if values have a process-related error.	X	X
Start value PLC	Start value in the CPU. This column is shown if there is an online connection and the "Monitor all" button  is selected. Error indication if values have a process-related error.		X
Monitor value	Current value in the CPU. This column is shown if there is an online connection and the "Monitor all" button  is selected. Error indication if values have a process-related error.		X
Modify value	Value that is to be used to change the monitor value. This column is shown if there is an online connection and the "Monitor all" button  is selected. Error indication if entered values have a syntax or process-related error.		X

3.7 Parameter view

Column	Explanation	Offline	Online
Selection for transmission 	Selection of the Modify values that are to be transmitted using the "Modify all selected parameters immediately and once" button. This column is displayed together with the "Modify value" column.		X
Minimum value	Minimum process-related value of the parameter. If the minimum value is dependent on other parameters, it is defined: <ul style="list-style-type: none"> <li>Offline: By the Start value project.</li> <li>Online: By the Monitor values.</li> </ul>	X	X
Maximum value	Maximum process-related value of the parameter. If the maximum value is dependent on other parameters, it is defined: <ul style="list-style-type: none"> <li>Offline: By the Start value project.</li> <li>Online: By the Monitor values.</li> </ul>	X	X
Setpoint	Designates the parameter as a setpoint. These parameters can be initialized online.	X	X
Data type	Data type of the parameter. The display field is empty for parameters that are not contained in the instance DB / technology DB.	X	X
Retain	Designates the value as a retentive value. The values of retentive parameters are retained even after the voltage supply is switched off.	X	X
Accessible from HMI	Indicates whether the HMI can access this parameter during runtime.	X	X
Visible in HMI	Indicates whether the parameter is visible in the selection list of the HMI by default.	X	X
Comment	Brief description of the parameter.	X	X

See also

Comparing values (Page 42)

### 3.7.3 Opening the parameter view

#### Requirement

The technology object has been added in the project tree, i.e., the associated instance DB / technology DB of the instruction has been created.

#### Procedure

1. Open the "Technology objects" folder in the project tree.
2. Open the technology object in the project tree.
3. Double-click the "Configuration" object.
4. Select the "Parameter view" tab in the top right corner.

#### Result

The Parameter view opens. Each displayed parameter is represented by one row in the parameter table.

The displayable parameter properties (table columns) vary depending on whether you are working with the Parameter view in offline or online mode.

In addition, you can selectively display and hide individual table columns.

#### See also

Default setting of the parameter view (Page 54)

### 3.7.4 Default setting of the parameter view

#### Default settings

To enable you to work efficiently with the Parameter view, you can customize the parameter display and save your settings.

The following customizations are possible and can be saved:

- Show and hide columns
- Change column width
- Change order of the columns
- Toggle navigation
- Select parameter group in the navigation
- Selection of compare values

#### Show and hide columns

To show or hide columns in the parameter table, follow these steps:

1. Position the cursor in the header of the parameter table.
2. Select the "Show/Hide" command in the shortcut menu.  
The selection of available columns is displayed.
3. To show a column, select the check box for the column.
4. To hide a column, clear the check box for the column.

or

1. Position the cursor in the header of the parameter table.
2. Select the "Show all columns" command in the shortcut menu if all columns of the offline or online mode are to be displayed.

Some columns can only be displayed in online mode: see Parameter table (Page 51).

## Change column width

To customize the width of a column so that all texts in the rows can be read, follow these steps:

1. Position the cursor in the header of the parameter table to the right of the column to be customized until the shape of the cursor changes to a cross.
2. Then double-click this location.

or

1. Open the shortcut menu on the header of the parameter table.
2. Click
  - "Optimize column width" or
  - "Optimize width of all columns".

If the column width setting is too narrow, the complete content of individual fields are shown if you hover the cursor briefly over the relevant field.

## Change order of the columns

The columns of the parameter table can be arranged in any way.

To change the order of the columns, follow these steps:

1. Click on the column header and use a drag-and-drop operation to move it to the desired location.

When you release the mouse button, the column is anchored to the new position.

## Toggle navigation

To toggle the display form of the parameters, follow these steps:

1. Select the desired navigation in the "Select navigation structure" drop-down list.
  - Data navigation
  - Functional navigation

See also Navigation (Page 50).

## Select parameter group in the navigation

Within the selected navigation, you choose between the "All parameters" display or the display of a subordinate parameter group of your choice.

1. Click the desired parameter group in the navigation.

The parameter table only displays the parameters of the parameter group.

### Selection of compare values (online)


To set the compare values for the “Compare values” function, follow these steps:

1. Select the desired compare values in the “Selection of compare values” drop-down list.
  - Start value project / Start value PLC
  - Start value project / Snapshot
  - Start value PLC / Snapshot

The “Start value project / Start value PLC” option is set by default.

### Saving the default setting of the Parameter view

To save the above customizations of the Parameter view, follow these steps:

1. Customize the Parameter view according to your requirements.
2. Click the “Save window settings” button  at the top right of the Parameter view.



## 3.7.5 Working with the parameter view

### 3.7.5.1 Overview

The following table provides an overview of the functions of the Parameter view in online and offline mode described in the following.

- Column "Offline" = X: This function is possible in offline mode.
- Column "Online" = X: This function is possible in online mode.

Function/action	Offline	Online
Filtering the parameter table (Page 58)	X	X
Sorting the parameter table (Page 59)	X	X
Transferring parameter data to other editors (Page 59)	X	X
Indicating errors (Page 60)	X	X
Editing start values in the project (Page 61)	X	X
Status of configuration (offline) (Page 63)	X	
Monitoring values online in the parameter view (Page 64)		X
Create snapshot of monitor values (Page 65)		X
Modifying values (Page 66)		X
Comparing values (Page 68)		X
Applying values from the online program as start values (Page 70)		X
Initializing setpoints in the online program (Page 71)		X

### 3.7.5.2 Filtering the parameter table

You can filter the parameters in the parameter table in the following ways:

- With the text filter
- With the subgroups of the navigation

Both filter methods can be used simultaneously.

#### With the text filter

Texts that are visible in the parameter table can be filtered. This means only texts in displayed parameter rows and columns can be filtered.

1. Enter the desired character string for filtering in the "Text filter..." input box.

The parameter table displays only the parameters containing the character string.

The text filtering is reset.

- When another parameter group is selected in the navigation.
- When navigation is changed from data navigation to functional navigation, or vice versa.

#### With the subgroups of the navigation

1. Click the desired parameter group in the navigation, e.g., "Static".

The parameter table only shows the static parameters. You can select further subgroups for some groups of the navigation.

2. Click "All parameters" in the navigation if all parameters are to be shown again.

### 3.7.5.3 Sorting the parameter table

The values of the parameters are arranged in rows. The parameter table can be sorted by any displayed column.

- In columns containing numerical values, sorting is based on the magnitude of the numerical value.
- In text columns, sorting is alphabetical.

#### Sorting by column

1. Position the cursor in the header cell of the desired column.  
The background of this cell turns blue.
2. Click the column header.

#### Result

The entire parameter table is sorted by the selected column. A triangle with tip facing up appears in the column header.

Clicking the column header again changes the sorting as follows:

- Symbol “▲”: Parameter table is sorted in ascending order.
- Symbol “▼”: Parameter table is sorted in descending order.
- No symbol: The sorting is removed again. The parameter table assumes the default display.

The “./” prefix in the “Name in DB” column is ignored when sorting.

### 3.7.5.4 Transferring parameter data to other editors

After selecting an entire parameter row of the parameter table, you can use the following:

- Drag-and-drop
- <Ctrl+C>/<Ctrl+V>
- Copy/Paste via shortcut menu

Transfer parameters to the following editors of the TIA Portal:

- Program editor
- Watch table
- Signal table for trace function

The parameter is inserted with its full name: See information in “Full name in DB” column.

### 3.7.5.5 Indicating errors

#### Error indication

Parameter assignment errors that result in compilation errors (e.g., limit violation) are indicated in the Parameter view.

Every time a value is input in the Parameter view, a check is made for process-related and syntax errors and the result is indicated.

Bad values are indicated by:

- Red error symbol in the "Status of configuration" (offline mode) or "Compare result" (online mode, depending on the selected comparison type) columns

and/or

- Table field with red background

If you click the bad field, a roll-out error message appears with information of the permissible value range or the required syntax (format)

#### Compilation error

From the error message of the compiler, you can directly open the Parameter view (functional navigation) containing the parameter causing the error in situations where the parameter is not displayed in the configuration dialog.

### 3.7.5.6 Editing start values in the project

With the Parameter view, you can edit the start values in the project in offline mode and online mode.

- You make value changes in the “Start value project” column of the parameter table.
- In the “Status of configuration” column of the parameter table, the progress of the configuration is indicated by the familiar status symbols from the configuration dialog of the technology object.

### Boundary conditions

- If other parameters depend on the parameter whose start value was changed, the start value of the dependent parameters are also adapted.
- If a parameter of a technology object is not editable, it is also not editable in the parameter view. The ability to edit a parameter can also depend on the values of other parameters.

### Defining new start values

To define start values for parameters in the Parameter view, follow these steps:

1. Open the Parameter view of the technology object.
2. Enter the desired start values in the "Start value project" column. The value must match the data type of the parameter and must not exceed the value range of the parameter. The limits of the value range can be seen in the “Maximum value” and “Minimum value” columns.

The "Status of configuration" column indicates the progress of the configuration with colored symbols.

See also Status of configuration (offline) (Page 63)

Following adaptation of the start values and downloading of the technology object to the CPU, the parameters take the defined value at startup if they are not declared as retentive (“Retain” column).

### Error indication

When a start value is input, a check is made for process-related and syntax errors and the result is indicated.

Bad start values are indicated by:

- Red error symbol in the "Status of configuration" (offline mode) or "Compare result" (online mode, depending on the selected comparison type) columns

and/or

- Red background in the “Start value project” field  
If you click on the bad field, a roll-out error message appears with information of the permissible value range or the necessary syntax (format)

### **Correcting bad start values**

1. Correct bad start values using information from the roll-out error message.  
Red error symbol, red field background, and roll-out error message are no longer displayed.






The project cannot be successfully compiled unless the start values are error-free.

### 3.7.5.7 Status of configuration (offline)

The status of the configuration is indicated by icons:

- In the “Status of configuration” column in the parameter table
- In the navigation structure of the functional navigation and data navigation

#### Symbol in “Status of configuration” column

Symbol	Meaning
	The start value of the parameter corresponds to the default value and is valid. A start value has not yet been defined by the user.
	The start value of the parameter contains a value defined by the user or an automatically adjusted value. The start value is different than the default value. The start value is error-free and valid.
	The start value of the parameter is invalid (syntax or process-related error). The input box has a red background. When clicked, the roll-out error message indicates the cause of the error.
	Only for S7-1200 Motion Control: The start value of the parameter is valid but contains warnings. The input box has a yellow background.
	The parameter is not relevant in the current configuration.

#### Symbol in the navigation

The symbols in the navigation indicate the progress of the configuration in the same way as in the configuration dialog of the technology object.

#### See also

Configure technology objects (Page 45)

### 3.7.5.8 Monitoring values online in the parameter view

You can monitor the values currently taken by the parameters of the technology object in the CPU (monitor values) directly in the Parameter view.

#### Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.

#### Procedure


1. Start the monitoring by clicking .

As soon as the Parameter view is online, the following columns are additionally displayed:

- Compare result
- Start value PLC
- Monitor value
- Modify value
- Selection for transmission


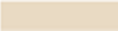
The "Monitor value" column shows the current parameter values on the CPU.

Meaning of the additional columns: see Parameter table (Page 51)

2. Stop the monitoring by clicking  again.

#### Display

All columns that are only available online have an orange background:


- Values in light-orange cells  can be changed.
- Values in cells with a dark orange background  cannot be changed.



### 3.7.5.9 Create snapshot of monitor values

You can back up the current values of the technology object on the CPU (monitor values) and display them in the Parameter view.

#### Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.
- The "Monitor all" button  is selected.

#### Procedure

To show the current parameter values, follow these steps:

1. In the Parameter view, click the "Create snapshot of monitor values" icon .

#### Result


The current monitor values are transferred once to the "Snapshot" column of the parameter table.

You can analyze the values "frozen" in this way while the monitor values continue to be updated in the "Monitor values" column.


### 3.7.5.10 Modifying values

With the Parameter view, you can modify values of the technology object in the CPU.

You can assign values to the parameter once (Modify value) and modify them immediately. The modify request is executed as quickly as possible without reference to any particular point in the user program.

 <b>DANGER</b>
Danger when modifying: Changing the parameter values while the plant is operating may result in severe damage to property and personal injury in the event of malfunctions or program errors. Make sure that dangerous states cannot occur before you use the "Modify" function.

### Requirements


- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.
- The "Monitor all" button  is selected.
- The parameter can be modified (associated field in the "Modify value" column has a light-orange background).

### Procedure

To modify parameters immediately, follow these steps:

1. Enter the desired modify values in the "Modify values" column of the parameter table.
2. Check whether the check box for modifying is selected in the "Select for transmission" column.

The modify values and associated check boxes of dependent parameters are automatically adapted at the same time.

3. Click the "Modify all selected parameters immediately and once" icon .

The selected parameters are modified once and immediately with the specified values and can be monitored in the "Modify values" column. The check boxes for modifying in the "Selection for transmission" column are automatically cleared after the modify request is complete.

## Error indication

When a start value is input, a check is made immediately for process-related and syntax errors and the result is indicated.

Bad start values are indicated by:

- Red background in the “Modify value” field

and

- If you click the bad field, a roll-out error message appears with information of the permissible value range or the necessary syntax (format)

## Bad modify values


- Modify values with process-related errors can be transmitted.
- Modify values with syntax errors **cannot** be transmitted.

### 3.7.5.11 Comparing values

You can use comparison functions to compare the following memory values of a parameter:


- Start value project
- Start value PLC
- Snapshot

#### Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.
- The "Monitor all" button  is selected.

#### Procedure

To compare the start values on the various target systems, follow these steps:

1. Click the "Selection of compare values" icon .

A selection list containing the comparison options opens:






- Start value project - Start value PLC (default setting)
- Start value project - Snapshot
- Start value PLC - Snapshot

2. Select the desired comparison option.

The selected comparison option is executed as follows:

- A scales symbol appears in the header cells of the two columns selected for comparison.
- Symbols are used in the "Compare result" column to indicate the result of the comparison of the selected columns.

**Symbol in "Compare result" column**

Symbol	Meaning
	The compare values are equal and error-free.
	The compare values are not equal and error-free.
	At least one of the two compare values has a process-related or syntax error.
	The comparison cannot be performed. At least one of the two comparison values is not available (e.g. snapshot).
	Comparison of the value is inappropriate since it is not relevant in one of the configurations.


**Symbol in the navigation**

The symbols are shown in the same way in the navigation if the comparison result applies to at least one of the parameters below the displayed navigation structure.

### 3.7.5.12 Applying values from the online program as start values


In order to apply optimized values from the CPU to the project as start values, you create a snapshot of the monitor values. Values of the snapshot marked as a "Setpoint" are then applied to the project as start values.

#### Requirements

- The technology object is of type "PID\_Compact" or "PID\_3Step".
- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.
- The "Monitor all" button  is selected.

#### Procedure

To apply optimized values from the CPU, follow these steps:

1. Click the "Create snapshot of monitor values and accept setpoints of this snapshot as start values" icon .

#### Result

The current monitor values are applied to the "Snapshot" column and their setpoints are copied to the "Start value project" column as new start values.

---

#### Note

##### Applying values of individual parameters

You can also apply the values of individual parameters that are not marked as a setpoint from the "Snapshot" column to the "Start values project" column. To do so, copy the values and insert them into the "Start value project" column using the "Copy" and "Paste" commands in the shortcut menu.

---

### 3.7.5.13 Initializing setpoints in the online program

You can initialize all parameters that are marked as a "Setpoint" in the Parameter view with new values in the CPU in one step. In so doing, the start values are downloaded from the project to the CPU. The CPU remains in "RUN" mode.

To avoid data loss on the CPU during a cold restart or warm restart, you must also download the technology object to the CPU.


#### DANGER

##### Danger when changing parameter values

Changing the parameter values while the plant is operating may result in severe damage to property and personal injury in the event of malfunctions or program errors.


Make sure that dangerous states cannot occur before you reinitialize the setpoints.

### Requirements

- The technology object is of type "PID\_Compact" or "PID\_3Step".
- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.
- The "Monitor all" button  is selected.
- The parameters marked as a "Setpoint" have a "Start value project" that is free of process-related and syntax errors

### Procedure

To initialize all setpoints, follow these steps:

1. Enter the desired values in the "Start value project" column.  
Ensure that the start values are free of process-related and syntax errors.
2. Click the "Initialize setpoints" icon .

### Result

The setpoints in the CPU are initialized with the start values from the project.

## 3.8 Downloading technology objects to device

A new or modified configuration of the technology object must be downloaded to the CPU for the online mode. The following characteristics apply when downloading retentive data:

- **Software (changes only)**
  - S7-1200, S7-1500:  
Retentive data is retained.
  - S7-300/400:  
Retentive data is updated immediately. CPU does not change to Stop.
- **Download PLC program to device and reset**
  - S7-1200, S7-1500:  
Retentive data is updated at the next change from Stop to RUN. The PLC program can only be downloaded completely.
  - S7-300/400:  
Retentive data is updated at the next change from Stop to RUN.

### Downloading retentive data to an S7-1200 or S7-1500 CPU

---

#### Note

The download and reset of the PLC program during ongoing system operation can result in serious damages or injuries in the case of malfunctions or program errors.

Make sure that dangerous states cannot occur before you download and reset the PLC program.

---

Proceed as follows to download the retentive data:

1. Select the entry of the CPU in the project tree.
2. Select the command "Download and reset PLC program" from the "Online" menu.
  - If you have not established an online connection yet, the "Extended download" dialog opens. In this case, set all required parameters for the connection and click "Download".
  - If the online connection has been defined, the project data is compiled, if necessary, and the dialog "Load preview" opens. This dialog displays messages and recommends actions necessary for download.
3. Check the messages.

As soon as download is possible, the "Download" button becomes active.
4. Click on "Download".

The complete PLC program is downloaded and the "Load results" dialog opens. This dialog displays the status and the actions after the download.
5. If the modules are to restart immediately after the download, select the check box "Start all".
6. Close the dialog "Download results" with "Finish".



## **Result**

The complete PLC program is downloaded to the device. Blocks that only exist online in the device are deleted. By downloading all affected blocks and by deleting any blocks in the device that are not required, you avoid inconsistencies between the blocks in the user program.

The messages under "Info > General" in the Inspector window indicate whether the download was successful.

## 3.9 Commissioning software controller

### Procedure

To open the "Commissioning" work area of the technology object, follow these steps:

1. Open the "Technology objects" folder in the project tree.
2. Open the technology object in the project tree.
3. Double-click the "Commissioning" object.

The commissioning functions are specific for each controller and are described there.

## 3.10 Save optimized PID parameter in the project


The software controller is optimized in the CPU. Through this, the values in the instance-DB on the CPU no longer agree with those in the project.

To update the PID parameter in the project with the optimized PID parameters, proceed as follows:

### Requirement

- An online connection to the CPU is established and the CPU is in "RUN" mode.
- The functions of the commissioning window have been enabled by means of the "Start" button.

### Procedure

1. Open the CPU folder in the project tree.
2. Open the "Technology objects" folder.
3. Open a technology object.
4. Double click on "Commissioning".
5. Click on the  icon "Upload PID parameters".
6. Save the project.

### Result

The currently active PID parameters are stored in the project data. When reloading the project data in the CPU, the optimized parameters are used.

### **3.11 Display instance DB of a technology object.**

An instance DB, in which the parameter and static variables are saved, is created for each technology object.

#### **Procedure**

To display the instance DB of a technology object, proceed as follows:

1. Open the CPU folder in the project tree.
2. Open the "Technology objects" folder.
3. Highlight a technology object.
4. Select the command "Open DB editor" in the shortcut menu.

## Using PID\_Compact

### 4.1 PID\_Compact V2

#### 4.1.1 Configuring PID\_Compact V2

##### 4.1.1.1 Basic settings V2

#### Introduction V2

Configure the following properties of the "PID\_Compact" technology object under "Basic settings" in the Inspector window or in the configuration window:

- Physical quantity
- Control logic
- Start-up behavior after reset
- Setpoint (only in the Inspector window)
- Process value (only in the Inspector window)
- Output value (only in the Inspector window)

#### Setpoint, process value and output value

You can only configure the setpoint, process value and output value in the Inspector window of the programming editor. Select the source for each value:

- Instance DB
  - The value saved in the instance DB is used.
  - Value must be updated in the instance DB by the user program.
  - There should be no value at the instruction.
  - Change via HMI possible.
- Instruction
  - The value connected to the instruction is used.
  - The value is written to the instance DB each time the instruction is called.
  - No change via HMI possible.

## Control mode V2

### Physical quantity

Select the physical quantity and unit of measurement for setpoint, process value, and disturbance variable in the "Controller type" group. Setpoint, process value, and disturbance variable is displayed in this unit of measurement.

### Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic.

PID\_Compact does not work with negative proportional gain. Select the check box "Invert control logic" to reduce the process value with a higher output value.

Examples

- Opening the drain valve will reduce the level of a container's contents.
- Increasing cooling will reduce the temperature.

### Startup characteristics

1. To switch to "Inactive" mode after CPU restart, clear the "Activate Mode after CPU restart" check box.

To switch to the operating mode saved in the Mode parameter after CPU restart, select the "Activate Mode after CPU restart" check box.

2. In the "Set Mode to" drop-down list, select the mode that is to be enabled after a complete download to the device.

After a complete download to the device, PID\_Compact starts in the selected operating mode. With each additional restart, PID\_Compact starts in the mode that was last saved in Mode.

Example

You have selected the "Activate Mode after CPU restart" check box and the entry "Pretuning" in the "Set Mode to" list. After a complete download to the device, PID\_Compact starts in the "Pretuning" mode. If pretuning is still active, PID\_Compact starts in "Pretuning" mode again after restart of the CPU. If pretuning was successfully completed and automatic mode is active, PID\_Compact starts in "Automatic mode" after restart of the CPU.

## Setpoint V2

### Procedure

Proceed as follows to define a fixed setpoint:

1. Select "Instance DB".
2. Enter a setpoint, e.g. 80° C.
3. Delete any entry in the instruction.

Proceed as follows to define a variable setpoint:

1. Select "Instruction".
2. Enter the name of the REAL variable in which the setpoint is saved.

Program-controlled assignment of various values to the REAL variable is possible, for example for the time controlled change of the setpoint.

## Process value V2

PID\_Compact will scale the value of the analog input to the physical quantity if you use the analog input value directly.

You will need to write a program for processing if you wish first to process the analog input value. The process value is, for example, not directly proportional to the value at the analog input. The processed process value must be in floating point format.

### Procedure

Proceed as follows to use the analog input value without processing:

1. Select the entry "Input\_PER" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the address of the analog input.

Proceed as follows to use the processed process value in floating point format:

1. Select the entry "Input" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the name of the variable in which the processed process value is saved.

## Output value V2

PID\_Compact offers three output values. Your actuator will determine which output value you use.

- Output\_PER  
The actuator is triggered via an analog output and controlled with a continuous signal, e.g. 0...10V, 4...20mA.
- Output  
The output value needs to be processed by the user program, for example because of nonlinear actuator response.
- Output\_PWM  
The actuator is controlled via a digital output. Pulse width modulation creates minimum ON and minimum OFF times.

## Procedure

Proceed as follows to use the analog output value:

1. Select the entry "Output\_PER (analog)" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the address of the analog output.

Proceed as follows to process the output value using the user program:

1. Select the entry "Output" in the drop-down list "Output".
2. Select "Instance DB".  
The calculated output value is saved in the instance data block.
3. For the preparation of the output value, use the output parameter Output.
4. Transfer the processed output value to the actuator via a digital or analog CPU output.

Proceed as follows to use the digital output value:

1. Select the entry "Output\_PWM" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the address of the digital output.

### 4.1.1.2 Process value settings V2

#### Process value scaling V2

If you have configured the use of Input\_PER in the basic setting, you must convert the value of the analog input to the physical quantity of the process value. The current configuration is displayed in the Input\_PER display.

Input\_PER will be scaled using a low and high value pair if the process value is directly proportional to the value of the analog input.

#### Procedure

To scale the process value, follow these steps:

1. Enter the low pair of values in the "Scaled low process value" and "Low" input fields.
2. Enter the high pair of values in the "Scaled high process value" and "High" input boxes.

Default settings for the value pairs are stored in the hardware configuration. To use the value pairs from the hardware configuration, follow these steps:

1. Select the PID\_Compact instruction in the programming editor.
2. Interconnect Input\_PER with an analog input in the basic settings.
3. Click the "Automatic setting" button in the process value settings.

The existing values will be overwritten with the values from the hardware configuration.

#### Process value limits V2

You must specify an appropriate absolute high limit and low limit for the process value as limit values for your controlled system. As soon as the process value violates these limits, an error occurs (ErrorBits = 0001h). Tuning is canceled when the process value limits are violated. You can configure how PID\_Compact reacts to an error in automatic mode in the output value settings.



### 4.1.1.3 Advanced settings V2

#### Process value monitoring V2

Configure a warning high and low limit for the process value in the "Process value monitoring" configuration window. If one of the warning limits is exceeded or undershot during operation, a warning will be displayed at the PID\_Compact instruction:

- At the InputWarning\_H output parameter if the warning high limit has been exceeded
- At the InputWarning\_L output parameter if the warning low limit has been undershot

The warning limits must be within the process value high and low limits.

The process value high and low limits will be used if you do not enter values.

#### Example

Process value high limit = 98 °C; warning high limit = 90 °C

Warning low limit = 10 °C; process value low limit = 0 °C

PID\_Compact will respond as follows:

Process value	InputWarning_H	InputWarning_L	Error-Bits	Operating mode
> 98 °C	TRUE	FALSE	0001h	Inactive or Substitute output value with error monitoring
≤ 98 °C and > 90 °C	TRUE	FALSE	0000h	Automatic mode
≤ 90 °C and ≥ 10 °C	FALSE	FALSE	0000h	Automatic mode
< 10 °C and ≥ 0 °C	FALSE	TRUE	0000h	Automatic mode
< 0 °C	FALSE	TRUE	0001h	Inactive or Substitute output value with error monitoring

In the output value settings, you can specify the reaction of PID\_Compact when the process value high limit or low limit is violated.

#### See also

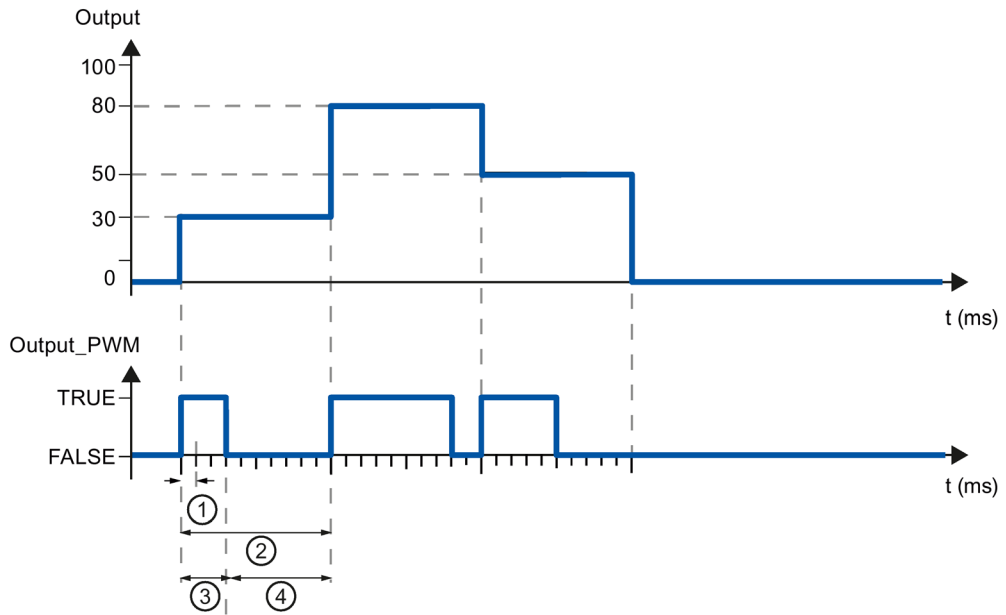
Parameters State and Mode V2 (Page 274)

### PWM limits V2

The value at the output parameter Output is transformed into a pulse sequence that is output at output parameter Output\_PWM by means of a pulse width modulation. Output is calculated in the PID algorithm sampling time, Output\_PWM is output in the PID\_Compact sampling time.

The PID algorithm sampling time is determined during pretuning or fine tuning. If manually setting the PID parameters, you will also need to configure the PID algorithm sampling time. The PID\_Compact sampling time is equivalent to the cycle time of the calling OB.

The pulse duration is proportional to the value at Output and is always an integer multiple of the PID\_Compact sampling time.



- ① PID\_Compact sampling time
- ② PID algorithm sampling time
- ③ Pulse duration
- ④ Break time

The "Minimum ON time" and the "Minimum OFF time" are rounded to an integer multiple of the PID\_Compact sampling time.

A pulse or a break is never shorter than the minimum ON or OFF time. The inaccuracies this causes are added up and compensated in the next cycle.

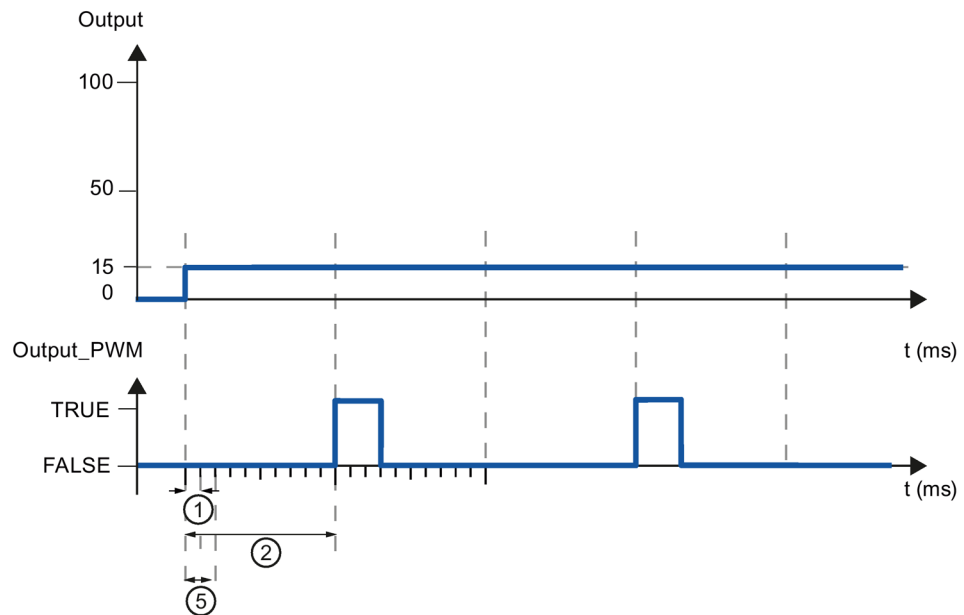
### Example

PID\_Compact sampling time = 100 ms

PID algorithm sampling time = 1000 ms

Minimum ON time = 200 ms

Output is a constant 15%. The smallest pulse that PID\_Compact can output is 20%. In the first cycle, no pulse is output. In the second cycle, the pulse not output in the first cycle is added to the pulse of the second cycle.



- ① PID\_Compact sampling time
- ② PID algorithm sampling time
- ⑤ Minimum ON time

In order to minimize operation frequency and conserve the actuator, extend the minimum ON and OFF times.

If you are using "Output" or "Output\_PER", you must configure the value 0.0 for the minimum ON and OFF times.

---

### Note

The minimum ON and OFF times only affect the output parameter Output\_PWM and are not used for any pulse generators integrated in the CPU.

---

## Output value V2

### Output value limits

In the "Output value limits" configuration window, configure the absolute limits of your output value in percent. Absolute output value limits are not violated in neither manual mode nor automatic mode. If an output value outside the limits is specified in manual mode, the effective value is limited in the CPU to the configured limits.

The output value limits must match the control logic.

The valid output value limit values depend on the Output used.

Output	-100.0 to 100.0%
Output_PER	-100.0 to 100.0%
Output_PWM	0.0 to 100.0%

### Reaction to error

**NOTICE**

**Your system may be damaged.**

If you output "Current value while error pending" or "Substitute output value while error pending" in the event of an error, PID\_Compact remains in automatic mode. This may cause a violation of the process value limits and damage your system.

It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

PID\_Compact is preset so that the controller stays active in most cases in the event of an error. If errors occur frequently in controller mode, this default reaction has a negative effect on the control response. In this case, check the Errorbits parameter and eliminate the cause of the error.

PID\_Compact generates a programmable output value in response to an error:

- Zero (inactive)

PID\_Compact outputs 0.0 as output value for all errors and switches to "Inactive" mode. The controller is only reactivated by a falling edge at Reset or a rising edge at ModeActivate.

- Current value while error is pending

If the following errors occur in **automatic mode**, PID\_Compact returns to automatic mode as soon as the errors are no longer pending.

If one or more of the following errors occur, PID\_Compact stays in automatic mode:

- 0001h: The "Input" parameter is outside the process value limits.
- 0800h: Sampling time error
- 40000h: Invalid value at Disturbance parameter.

If one or more of the following errors occur in **automatic mode**, PID\_Compact switches to "Substitute output value with error monitoring" mode and outputs the last valid output value:

- 0002h: Invalid value at Input\_PER parameter.
- 0200h: Invalid value at Input parameter.
- 0400h: Calculation of output value failed.
- 1000h: Invalid value at Setpoint parameter.

If an error occurs in **manual mode**, PID\_Compact continues using the manual value as the output value. If the manual value is invalid, the substitute output value is used. If the manual value and substitute output value are invalid, the output value low limit is used.

If the following error occurs during a **pretuning or fine tuning**, PID\_Compact remains in active mode:

- 0020h: Pretuning is not permitted during fine tuning.

When any other error occurs, PID\_Compact cancels the tuning and switches to the mode from which tuning was started.

As soon as no errors are pending, PID\_Compact returns to automatic mode.

- Substitute output value while error is pending

PID\_Compact outputs the substitute output value.

If the following error occurs, PID\_Compact stays in "Substitute output value with error monitoring" mode and outputs the output value low limit:

- 20000h: Invalid value at SubstituteOutput tag.

For all other errors, PID\_Compact reacts as described for "Current value while error is pending".

## See also

Parameters State and Mode V2 (Page 274)

**PID parameters V2**

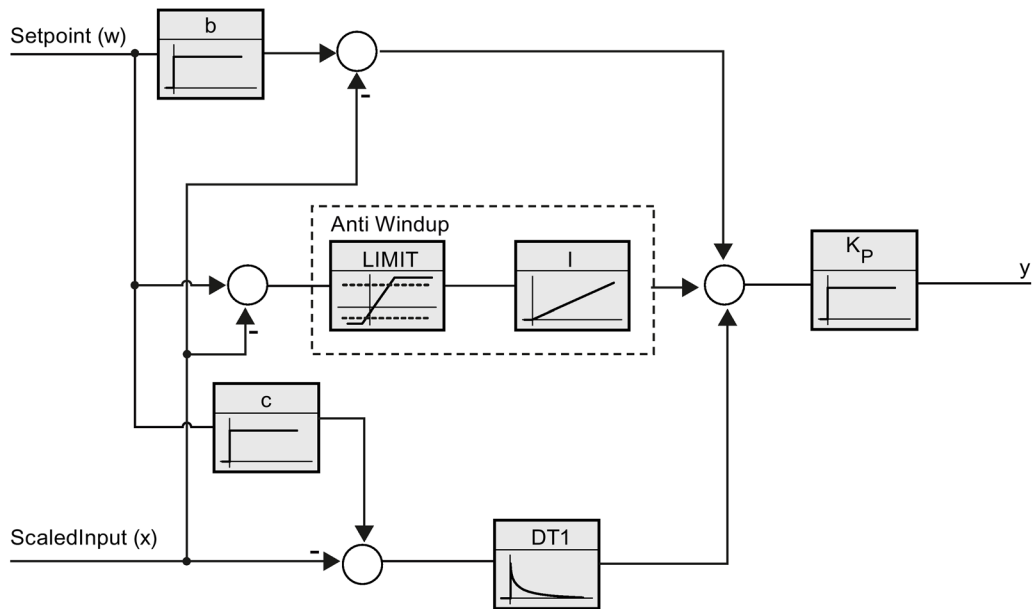
The PID parameters are displayed in the "PID Parameters" configuration window. The PID parameters will be adapted to your controlled system during controller tuning. You do not need to enter the PID parameters manually.

The PID algorithm operates according to the following equation:

$$y = K_p \left[ (b \cdot w - x) + \frac{1}{T_i \cdot s} (w - x) + \frac{T_d \cdot s}{a \cdot T_d \cdot s + 1} (c \cdot w - x) \right]$$

Symbol	Description
y	Output value of the PID algorithm
K <sub>p</sub>	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T <sub>i</sub>	Integral action time
a	Derivative delay coefficient (derivative delay T1 = a × T <sub>D</sub> )
T <sub>D</sub>	Derivative action time
c	Derivative action weighting

The diagram below illustrates the integration of the parameters into the PID algorithm:



All PID parameters are retentive. If you enter the PID parameters manually, you must completely download PID\_Compact.

Downloading technology objects to device (Page 72)

### Proportional gain

The value specifies the proportional gain of the controller. PID\_Compact does not work with a negative proportional gain. Control logic is inverted under Basic settings > Controller type.

### Integral action time

The integral action time determines the time behavior of the integral action. The integral action is deactivated with integral action time = 0.0.

### Derivative action time

The derivative action time determines the time behavior of the derivative action. Derivative action is deactivated with derivative action time = 0.0.

### Derivative delay coefficient

The derivative delay coefficient delays the effect of the derivative action.

Derivative delay = derivative action time × derivative delay coefficient

- 0.0: Derivative action is effective for one cycle only and therefore almost not effective.
- 0.5: This value has proved useful in practice for controlled systems with **one** dominant time constant.
- > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed.

### Proportional action weighting

The proportional action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Proportional action for setpoint change is fully effective
- 0.0: Proportional action for setpoint change is not effective

The proportional action is always fully effective when the process value is changed.

### Derivative action weighting

The derivative action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Derivative action is fully effective upon setpoint change
- 0.0: Derivative action is not effective upon setpoint change

The derivative action is always fully effective when the process value is changed.

### PID algorithm sampling time

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the cycle time. All other functions of PID\_Compact are executed at every call.

If you use Output\_PWM, the accuracy of the output signal is determined by the ratio of the PID algorithm sampling time to the cycle time of the OB. The PID algorithm sampling time corresponds to the time period of the pulse width modulation. The cycle time should be at least 10 times the PID algorithm sampling time.

### Rule for tuning

Select whether PI or PID parameters are to be calculated in the "Controller structure" drop-down list.

- **PID**  
Calculates PID parameters during pretuning and fine tuning.
- **PI**  
Calculates PI parameters during pretuning and fine tuning.

- **User-defined**  
The drop-down list displays "User-defined" if you have configured different controller structures for pretuning and fine tuning via a user program.



## 4.1.2 Commissioning PID\_Compact V2

### 4.1.2.1 Pretuning V2

The pretuning determines the process response to a jump change of the output value and searches for the point of inflection. The PID parameters are calculated from the maximum rate of rise and dead time of the controlled system. You obtain the best PID parameters when you perform pretuning and fine tuning.

The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. This is most likely the case in operating modes "Inactive" and "manual mode". The PID parameters are backed up before being recalculated.

#### Requirement

- The "PID\_Compact" instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- Reset = FALSE
- PID\_Compact is in one of the following modes: "Inactive", "Manual mode", or "Automatic mode".
- The setpoint and the process value lie within the configured limits (see "Process value monitoring" configuration).
- The difference between setpoint and process value is greater than 30% of the difference between process value high limit and process value low limit.
- The distance between the setpoint and the process value is > 50% of the setpoint.

## Procedure

To perform pretuning, follow these steps:

1. Double-click the "PID\_Compact > Commissioning" entry in the project tree.
2. Select the entry "Pretuning" in the "Tuning mode" drop-down list.
3. Click the "Start" icon.
  - An online connection will be established.
  - Value recording is started.
  - Pretuning is started.
  - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

---

### Note

Click the "Stop" icon when the progress bar has reached 100% and it can be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

---

## Result

If pretuning was performed without an error message, the PID parameters have been tuned. PID\_Compact switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If pretuning is not possible, PID\_Compact responds with the configured reaction to errors.

## See also

Parameters State and Mode V2 (Page 274)

### 4.1.2.2 Fine tuning V2

Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are tuned for the operating point from the amplitude and frequency of this oscillation. All PID parameters are recalculated from the results. PID parameters from fine tuning usually have better master control and disturbance characteristics than PID parameters from pretuning. You obtain the best PID parameters when you perform pretuning and fine tuning.

PID\_Compact automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The PID parameters are backed up before being recalculated.

#### Requirement

- The PID\_Compact instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- Reset = FALSE
- The setpoint and the process value lie within the configured limits.
- The control loop has stabilized at the operating point. The operating point is reached when the process value corresponds to the setpoint.
- No disturbances are expected.
- PID\_Compact is in one of the following operating modes: Inactive, automatic mode, or manual mode.

#### Process depends on initial situation

Fine tuning can be started from the following operating modes: "Inactive", "automatic mode", or "manual mode". Fine tuning proceeds as follows when started from:

- Automatic mode

Start fine tuning from automatic mode if you wish to improve the existing PID parameters through tuning.

PID\_Compact controls the system using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.

- Inactive or manual mode

If the requirements for pretuning are met, pretuning is started. The determined PID parameters will be used for control until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. If pretuning is not possible, PID\_Compact responds with the configured reaction to errors.

An attempt is made to reach the setpoint with the minimum or maximum output value if the process value for pretuning is already too near the setpoint. This can produce increased overshoot.

## Procedure

To perform fine tuning, follow these steps:

1. Select the entry "Fine tuning" in the "Tuning mode" drop-down list.
2. Click the "Start" icon.
  - An online connection will be established.
  - Value recording is started.
  - The process of fine tuning is started.
  - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

---

### Note

Click the "Stop" icon in the "Tuning mode" group when the progress bar has reached 100% and it is to be assumed that tuning is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

---

## Result

If no errors occurred during fine tuning, the PID parameters have been tuned. PID\_Compact switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If errors occurred during "fine tuning", PID\_Compact responds with the configured response to errors.

## See also

Parameters State and Mode V2 (Page 274)

### 4.1.2.3 "Manual" mode V1


The following section describes how you can use the "manual mode" operating mode in the commissioning window of the "PID\_Compact" technology object. Manual mode is also possible when an error is pending.

#### Requirement

- The "PID\_Compact" instruction is called in a cyclic interrupt OB.
- An online connection to the CPU has been established and the CPU is in the "RUN" mode.

#### Procedure

Use "Manual mode" in the commissioning window if you want to test the controlled system by specifying a manual value. To define a manual value, follow these steps:

1. Click the "Start" icon.
2. Select the "Manual mode" check box in the "Online status of controller" area.  
PID\_Compact operates in manual mode. The most recent current output value remains in effect.
3. Enter the manual value in the "Output" field as a % value.
4. Click the  icon.

#### Result

The manual value is written to the CPU and immediately goes into effect.

Clear the "Manual mode" check box if the output value is to be specified again by the PID controller. The switchover to automatic mode is bumpless.

#### See also

Parameters State and Mode V2 (Page 274)

### 4.1.3 Override control with PID\_Compact V2

#### Override control

In case of override control, two or more controllers share one actuator. Only one controller has access to the actuator at any time and influences the process.

A logic operation decides which controller has access to the actuator. This decision is often made based on a comparison of the output values of all controllers, for example, in case of a maximum selection, the controller with the largest output value gets access to the actuator.

The selection based on the output value requires that all controllers operate in automatic mode. The controllers that do not have an effect on the actuator are updated. This is necessary to prevent windup effects and their negative impacts on the control response and the switchover between the controllers.

PID\_Compact supports override controls as of version 2.3 by offering a simple process for updating the controllers that are not active: By using the tags `OverwriteInitialOutputValue` and `PIDCtrl.PIDInit`, you can pre-assign the integral action of the controller in automatic mode as though the PID algorithm had calculated  $Output = OverwriteInitialOutputValue$  for the output value in the last cycle. To do this, `OverwriteInitialOutputValue` is interconnected with the output value of the controller that currently has access to the actuator. By setting the bit `PIDCtrl.PIDInit`, you trigger the pre-assignment of the integral action as well as the restart of the controller cycle and the PWM period. The subsequent calculation of the output value in the current cycle takes place based on the pre-assigned (and synchronized for all controllers) integral action as well as the proportional action and integral action from the current control deviation. The derivative action is not active during the call with `PIDCtrl.PIDInit = TRUE` and therefore does not contribute to the output value.

This procedure ensures that the calculation of the current output value and thus the decision on which controller is to have access to the actuator is only based on the current process state and the PI parameters. Windup effects for controllers that are not active and thus incorrect decisions of the switchover logic are prevented.

## Requirements

- PIDCtrl.PIDInit is only effective if the integral action is activated (Retain.CtrlParams.Ti tag > 0.0).
- You must assign PIDCtrl.PIDInit and OverwriteInitialOutputValue in your user program yourself (see example below). PID\_Compact does not automatically change these tags.
- PIDCtrl.PIDInit is only effective when PID\_Compact is in automatic mode (parameter State = 3)
- If possible, select the sampling time of the PID algorithm (Retain.CtrlParams.Cycle tag) in such a way that it is identical for all controllers, and call all controllers in the same cyclic interrupt OB. In this way, you ensure that the switchover does not take place within a controller cycle or a PWM period.

---

### Note

#### Constant adaptation of the output value limits

Instead of the active updating of the controllers without access to the actuator described here, this is implemented alternatively by constant adaptation of the output value limits in other controller systems.

This is not possible with PID\_Compact, because a change of the output value limits is not supported in automatic mode.

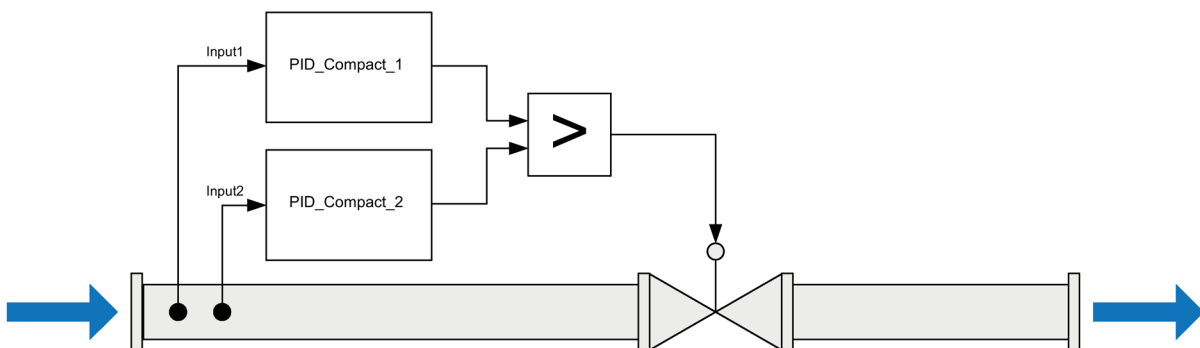
---

## Example: Control of a gas pipeline

PID\_Compact is used for control of a gas pipeline.

The main goal is to control the flow rate Input1. The controller PID\_Compact\_1 is used for this purpose. In addition, the pressure Input2 (measured in flow direction in front of the valve) is to be kept below the high limit with the limiting controller PID\_Compact\_2.

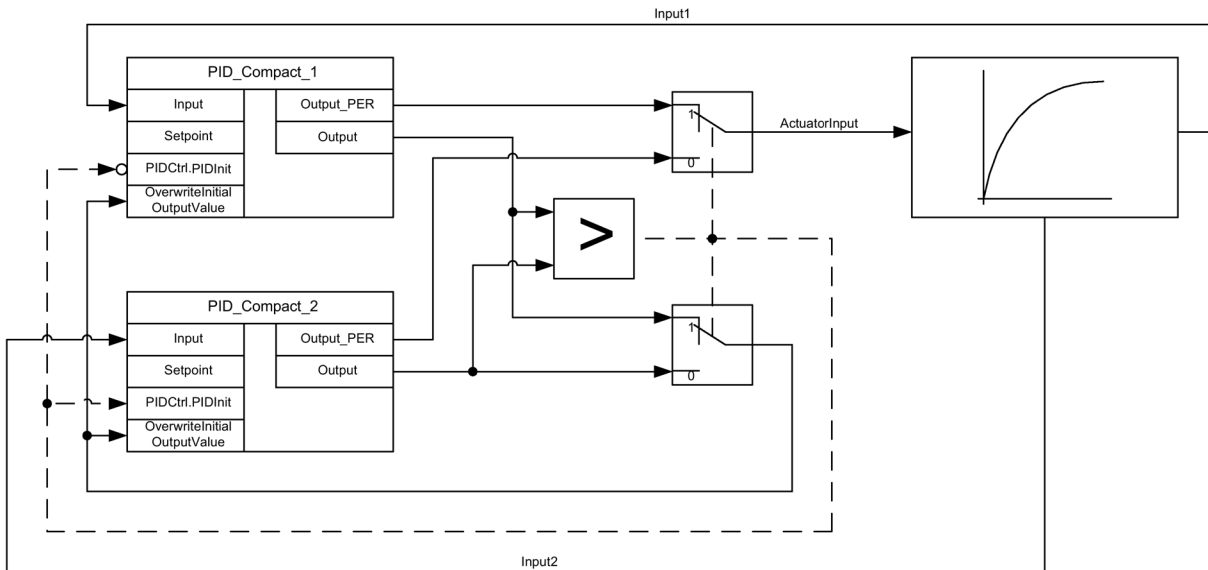
Flow rate and pressure are controlled by a single solenoid valve. The output value of the controller corresponds to the valve opening: The valve is opened when the output value increases. This means the flow rate increases (normal control logic) while the pressure drops (inverted control logic).



The valve is controlled with the output value of PID\_Compact in I/O format (parameter Output\_PER) by writing the program tag ActuatorInput.

The setpoint for the flow rate is specified at the parameter PID\_Compact\_1.Setpoint.

The pressure high limit is specified as setpoint at the parameter PID\_Compact\_2.Setpoint.



Both controllers must share one valve as shared actuator. The logic that decides which controller gets access to the actuator is implemented by a maximum selection of the output value (in Real format, parameter Output) in this case. Because the output value corresponds to the opening of the valve, the controller that requires the larger valve opening gets the control.

**Note**

**Activate inversion of the control logic**

Because a decrease of the actual value (pressure) is to be achieved with the pressure regulator PID\_Compact\_2 when the output value increases (valve opening), the inversion of the control logic must be activated: PID\_Compact\_2.Config.InvertControl = TRUE.

In normal operation of the plant, the actual value of the flow rate corresponds to the setpoint. The flow controller PID\_Compact\_1 has settled on a stationary output value PID\_Compact\_1.Output. The actual value of the pressure in normal operation is significantly below the high limit that is specified as setpoint for PID\_Compact\_2. The pressure regulator therefore wants to close the valve even further to increase the pressure, which means it will calculate an output value PID\_Compact\_2.Output that is smaller than the output value of the flow controller PID\_Compact\_1.Output. The maximum selection of the switchover logic therefore gives the flow controller PID\_Compact\_1 continued access to the actuator. In addition, it is ensured that PID\_Compact\_2 is updated by means of the assignments PID\_Compact\_2.OverwriteInitialOutputValue = PID\_Compact\_1.Output and PID\_Compact\_2.PIDCtrl.PIDInit = TRUE.



If the pressure now approaches the high limit or exceeds it, for example due to a fault, the pressure regulator PID\_Compact\_2 calculates a higher output value to open the valve even further and thus reduce the pressure. If PID\_Compact\_2.Output is greater than PID\_Compact\_1.Output, the pressure regulator PID\_Compact\_2 receives access to the actuator through the maximum selection and opens it. It is ensured that PID\_Compact\_1 is updated by means of the assignments PID\_Compact\_1.OverwriteInitialOutputValue = PID\_Compact\_2.Output and PID\_Compact\_1.PIDCtrl.PIDInit = TRUE.

The pressure is reduced while the flow rate increases and can no longer be kept at the setpoint.

Once the fault has been remedied, the pressure will continue to drop and the opening of the valve is reduced by the pressure regulator. If the flow controller calculates a larger opening as output value, the plant returns to normal operation so that the flow controller PID\_Compact\_1 once again has access to the actuator.

This example can be implemented with the following SCL program code:

```
"PID Compact 1"(Input := "Input1");  
"PID Compact 2"(Input := "Input2");  
IF "PID Compact 1".Output >= "PID Compact 2".Output THEN  
  "ActuatorInput" := "PID_Compact_1".Output_PER;  
  "PID_Compact_1".PIDCtrl.PIDInit := FALSE;  
  "PID_Compact_2".PIDCtrl.PIDInit := TRUE;  
  "PID_Compact_2".OverwriteInitialOutputValue := "PID_Compact_1".Output;  
ELSE  
  "ActuatorInput" := "PID_Compact_2".Output_PER;  
  "PID_Compact_1".PIDCtrl.PIDInit := TRUE;  
  "PID_Compact_2".PIDCtrl.PIDInit := FALSE;  
  "PID_Compact_1".OverwriteInitialOutputValue := "PID_Compact_2".Output;  
END IF;
```

#### 4.1.4 Simulating PID\_Compact V2 with PLCSIM

---

##### Note

##### Simulation with PLCSIM

The simulation of PID\_Compact V2.x with PLCSIM for CPU S7-1200 is not supported.

PID\_Compact V2.x can only be simulated for CPU S7-1500 with PLCSIM.

For the simulation with PLCSIM, the time behavior of the simulated PLC is not exactly identical to that of a "real" PLC. The actual cycle clock of a cyclic interrupt OB can have larger fluctuations with a simulated PLC than with "real" PLCs.

In the standard configuration, PID\_Compact determines the time between calls automatically and monitors them for fluctuations.

For the simulation of PID\_Compact with PLCSIM, for example, a sampling time error (ErrorBits = DW#16#00000800) can therefore be detected.

This results in ongoing tuning being aborted.

The response in automatic mode depends on the value of the ActivateRecoverMode tag.

To prevent this from happening, you should configure PID\_Compact for simulation with PLCSIM as follows:

- CycleTime.EnEstimation = FALSE
  - CycleTime.EnMonitoring = FALSE
  - CycleTime.Value: Assign the cycle clock of the calling cyclic interrupt OB in seconds to this tag.
-

## 4.2 PID\_Compact V1

### 4.2.1 Configuring PID\_Compact V1

#### 4.2.1.1 Basic settings V1

##### Introduction V1

Configure the following properties of the "PID\_Compact" technology object under "Basic settings" in the Inspector window or in the configuration window:

- Physical quantity
- Control logic
- Start-up behavior after reset
- Setpoint (only in the Inspector window)
- Process value (only in the Inspector window)
- Output value (only in the Inspector window)

##### Setpoint, process value and output value

You can only configure the setpoint, process value and output value in the Inspector window of the programming editor. Select the source for each value:

- Instance DB  
The value saved in the instance DB is used.  
Value must be updated in the instance DB by the user program.  
There should be no value at the instruction.  
Change via HMI possible.
- Instruction  
The value connected to the instruction is used.  
The value is written to the instance DB each time the instruction is called.  
No change via HMI possible.

## Control mode V1

### Physical quantity

Select the unit of measurement and physical quantity for the setpoint and process value in the "Controller type" group. The setpoint and process value will be displayed in this unit.

### Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic.

PID\_Compact does not work with negative proportional gain. Select the check box "Invert control logic" to reduce the process value with a higher output value.

Examples

- Opening the drain valve will reduce the level of a container's contents.
- Increasing cooling will reduce the temperature.

### Start-up behavior after reset

To change straight to the last active mode after restarting the CPU, select the "Enable last mode after CPU restart" check box.

PID\_Compact will remain in "Inactive" mode if the check box is cleared.

## Setpoint V1

### Procedure

Proceed as follows to define a fixed setpoint:

1. Select "Instance DB".
2. Enter a setpoint, e.g. 80° C.
3. Delete any entry in the instruction.

Proceed as follows to define a variable setpoint:

1. Select "Instruction".
2. Enter the name of the REAL variable in which the setpoint is saved.

Program-controlled assignment of various values to the REAL variable is possible, for example for the time controlled change of the setpoint.

## Process value V1

PID\_Compact will scale the value of the analog input to the physical quantity if you use the analog input value directly.

You will need to write a program for processing if you wish first to process the analog input value. The process value is, for example, not directly proportional to the value at the analog input. The processed process value must be in floating point format.

## Procedure

Proceed as follows to use the analog input value without processing:

1. Select the entry "Input\_PER" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the address of the analog input.

Proceed as follows to use the processed process value in floating point format:

1. Select the entry "Input" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the name of the variable in which the processed process value is saved.

## Output value V1

PID\_Compact offers three output values. Your actuator will determine which output value you use.

- Output\_PER

The actuator is triggered via an analog output and controlled with a continuous signal, e.g. 0...10V, 4...20mA.

- Output

The output value needs to be processed by the user program, for example because of nonlinear actuator response.

- Output\_PWM

The actuator is controlled via a digital output. Pulse width modulation creates minimum ON and minimum OFF times.

## Procedure

Proceed as follows to use the analog output value:

1. Select the entry "Output\_PER (analog)" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the address of the analog output.

Proceed as follows to process the output value using the user program:

1. Select the entry "Output" in the drop-down list "Output".
2. Select "Instance DB".

The calculated output value is saved in the instance data block.

3. For the preparation of the output value, use the output parameter Output.
4. Transfer the processed output value to the actuator via a digital or analog CPU output.

Proceed as follows to use the digital output value:

1. Select the entry "Output\_PWM" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the address of the digital output.

### 4.2.1.2 Process value settings V1

Configure the scaling of your process value and specify the process value absolute limits in the "Process value settings" configuration window.

#### Scaling the process value

If you have configured the use of Input\_PER in the basic settings, you will need to convert the value of the analog input into the physical quantity of the process value. The current configuration will be displayed in the Input\_PER display.

Input\_PER will be scaled using a low and high value pair if the process value is directly proportional to the value of the analog input.

1. Enter the low pair of values in the "Scaled low process value" and "Low" input fields.
2. Enter the high pair of values in the "Scaled high process value" and "High" input boxes.

Default settings for the value pairs are saved in the hardware configuration. Proceed as follows to use the value pairs from the hardware configuration:

1. Select the instruction PID\_Compact in the programming editor.
2. Connect Input\_PER with an analog input in the basic settings.
3. Click on the "Automatic setting" button in the process value settings.

The existing values will be overwritten with the values from the hardware configuration.

#### Monitoring process value

Specify the absolute high and low limit of the process value. As soon as these limits are violated during operation, the controller switches off and the output value is set to 0%. You must enter reasonable limits for your controlled system. Reasonable limits are important during optimization to obtain optimal PID parameters.

The default for the "High limit process value" is 120 %. At the I/O input, the process value can be a maximum of 18% higher than the standard range (overrange). An error is no longer reported for a violation of the "High limit process value". Only a wire-break and a short-circuit are recognized and the PID\_Compact switches to "Inactive" mode.

#### WARNING

If you set very high process value limits (for example  $-3.4 \cdot 10^{38} \dots +3.4 \cdot 10^{38}$ ), process value monitoring will be disabled. Your system may then be damaged if an error occurs.

#### See also

- Process value monitoring V1 (Page 104)
- PWM limits V1 (Page 105)
- Output value limits V1 (Page 108)
- PID parameters V1 (Page 109)

### 4.2.1.3 Advanced settings V1

#### Process value monitoring V1

Configure a warning high and low limit for the process value in the "Process value monitoring" configuration window. If one of the warning limits is exceeded or undershot during operation, a warning will be displayed at the PID\_Compact instruction:

- At the InputWarning\_H output parameter if the warning high limit has been exceeded
- At the InputWarning\_L output parameter if the warning low limit has been undershot

The warning limits must be within the process value high and low limits.

The process value high and low limits will be used if you do not enter values.

#### Example

Process value high limit = 98° C; warning high limit = 90° C

Warning low limit = 10° C; process value low limit = 0° C

PID\_Compact will respond as follows:

Process value	InputWarning_H	InputWarning_L	Operating mode
> 98° C	TRUE	FALSE	Inactive
≤ 98° C and > 90° C	TRUE	FALSE	Automatic mode
≤ 90° C and ≥ 10° C	FALSE	FALSE	Automatic mode
< 10° C and ≥ 0° C	FALSE	TRUE	Automatic mode
< 0° C	FALSE	TRUE	Inactive

#### See also

Process value settings V1 (Page 103)

PWM limits V1 (Page 105)

Output value limits V1 (Page 108)

PID parameters V1 (Page 109)

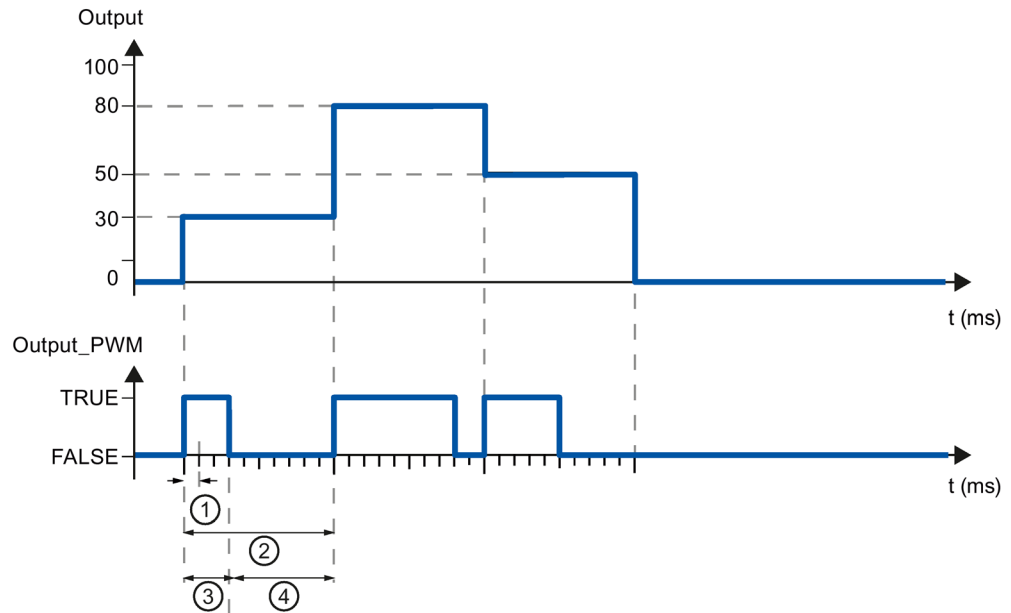


## PWM limits V1

The value at the output parameter Output is transformed into a pulse sequence that is output at output parameter Output\_PWM by means of a pulse width modulation. Output is calculated in the PID algorithm sampling time, Output\_PWM is output in the PID\_Compact sampling time.

The PID algorithm sampling time is determined during pretuning or fine tuning. If manually setting the PID parameters, you will also need to configure the PID algorithm sampling time. The PID\_Compact sampling time is equivalent to the cycle time of the calling OB.

The pulse duration is proportional to the value at Output and is always an integer multiple of the PID\_Compact sampling time.



- ① PID\_Compact sampling time
- ② PID algorithm sampling time
- ③ Pulse duration
- ④ Break time

The "Minimum ON time" and the "Minimum OFF time" are rounded to an integer multiple of the PID\_Compact sampling time.

A pulse or a break is never shorter than the minimum ON or OFF time. The inaccuracies this causes are added up and compensated in the next cycle.

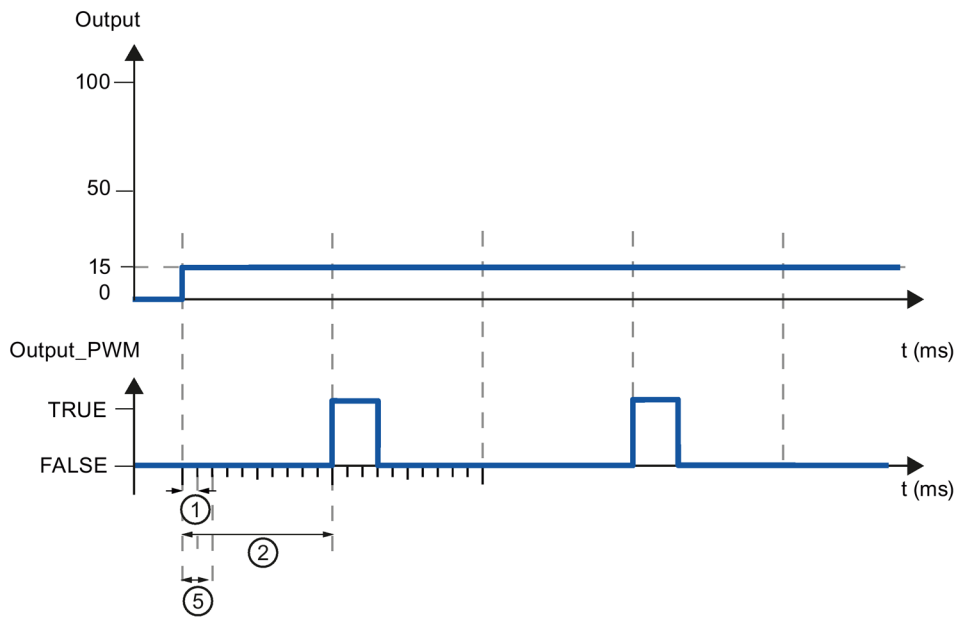
**Example**

PID\_Compact sampling time = 100 ms

PID algorithm sampling time = 1000 ms

Minimum ON time = 200 ms

Output is a constant 15%. The smallest pulse that PID\_Compact can output is 20%. In the first cycle, no pulse is output. In the second cycle, the pulse not output in the first cycle is added to the pulse of the second cycle.



- ① PID\_Compact sampling time
- ② PID algorithm sampling time
- ⑤ Minimum ON time

In order to minimize operation frequency and conserve the actuator, extend the minimum ON and OFF times.

If you are using "Output" or "Output\_PER", you must configure the value 0.0 for the minimum ON and OFF times.

---

**Note**

The minimum ON and OFF times only affect the output parameter Output\_PWM and are not used for any pulse generators integrated in the CPU.

---

**See also**

Process value settings V1 (Page 103)

Process value monitoring V1 (Page 104)

Output value limits V1 (Page 108)

PID parameters V1 (Page 109)

### Output value limits V1

In the "Output value limits" configuration window, configure the absolute limits of your output value in percent. Absolute output value limits are not violated in neither manual mode nor in automatic mode. If a output value outside the limits is specified in manual mode, the effective value is limited in the CPU to the configured limits.

The valid output value limit values depend on the Output used.

Output	-100.0 to 100.0
Output_PER	-100.0 to 100.0
Output_PWM	0.0 to 100.0

PID\_Compact sets the output value to 0.0 if an error occurs. 0.0 must therefore always be within the output value limits. You will need to add an offset to Output and Output\_PER in the user program if you want an output value low limit of greater than 0.0.

### See also

- Process value settings V1 (Page 103)
- Process value monitoring V1 (Page 104)
- PWM limits V1 (Page 105)
- PID parameters V1 (Page 109)

## PID parameters V1

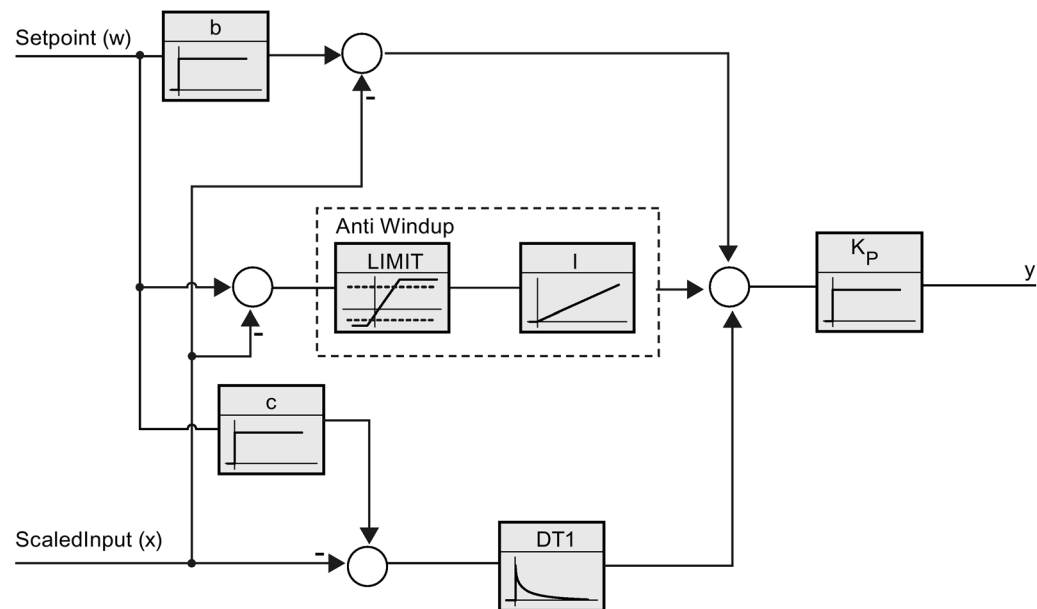
The PID parameters are displayed in the "PID Parameters" configuration window. The PID parameters will be adapted to your controlled system during controller tuning. You do not need to enter the PID parameters manually.

The PID algorithm operates according to the following equation:

$$y = K_p \left[ (b \cdot w - x) + \frac{1}{T_i \cdot s} (w - x) + \frac{T_d \cdot s}{a \cdot T_d \cdot s + 1} (c \cdot w - x) \right]$$

Symbol	Description
y	Output value of the PID algorithm
K <sub>p</sub>	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T <sub>i</sub>	Integral action time
a	Derivative delay coefficient (derivative delay T <sub>1</sub> = a × T <sub>D</sub> )
T <sub>D</sub>	Derivative action time
c	Derivative action weighting

The diagram below illustrates the integration of the parameters into the PID algorithm:



All PID parameters are retentive. If you enter the PID parameters manually, you must completely download PID\_Compact.

### Proportional gain

The value specifies the proportional gain of the controller. PID\_Compact does not work with a negative proportional gain. Control logic is inverted under Basic settings > Controller type.

### Integral action time

The integral action time determines the time behavior of the integral action. The integral action is deactivated with integral action time = 0.0.

### Derivative action time

The derivative action time determines the time behavior of the derivative action. Derivative action is deactivated with derivative action time = 0.0.

### Derivative delay coefficient

The derivative delay coefficient delays the effect of the derivative action.

Derivative delay = derivative action time × derivative delay coefficient

- 0.0: Derivative action is effective for one cycle only and therefore almost not effective.
- 0.5: This value has proved useful in practice for controlled systems with **one** dominant time constant.
- > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed.

### Proportional action weighting

The proportional action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Proportional action for setpoint change is fully effective
- 0.0: Proportional action for setpoint change is not effective

The proportional action is always fully effective when the process value is changed.

### Derivative action weighting

The derivative action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Derivative action is fully effective upon setpoint change
- 0.0: Derivative action is not effective upon setpoint change

The derivative action is always fully effective when the process value is changed.

## **PID algorithm sampling time**

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the cycle time. All other functions of PID\_Compact are executed at every call.

If you use Output\_PWM, the accuracy of the output signal is determined by the ratio of the PID algorithm sampling time to the cycle time of the OB. The PID algorithm sampling time corresponds to the time period of the pulse width modulation. The cycle time should be at least 10 times the PID algorithm sampling time.

## **Rule for tuning**

Select whether PI or PID parameters are to be calculated in the "Controller structure" drop-down list.

- **PID**  
Calculates PID parameters during pretuning and fine tuning.
- **PI**  
Calculates PI parameters during pretuning and fine tuning.
- **User-defined**  
The drop-down list displays "User-defined" if you have configured different controller structures for pretuning and fine tuning via a user program.

## **See also**

Downloading technology objects to device (Page 72)

## 4.2.2 Commissioning PID\_Compact V1

### 4.2.2.1 Commissioning V1

The commissioning window helps you commission the PID controller. You can monitor the values for the setpoint, process value and output value along the time axis in the trend view. The following functions are supported in the commissioning window:

- Controller pretuning
- Controller fine tuning
- Monitoring the current closed-loop control in the trend view
- Testing the controlled system by specifying a manual output value

All functions require an online connection to the CPU to have been established.

### Basic handling

- Select the desired sampling time in the "Sampling time" drop-down list.  
All values in the commissioning window are updated in the selected update time.
- Click the "Start" icon in the measuring group if you want to use the commissioning functions.  
Value recording is started. The current values for the setpoint, process value and output value are entered in the trend view. Operation of the commissioning window is enabled.
- Click the "Stop" icon if you want to end the commissioning functions.  
The values recorded in the trend view can continue to be analyzed.

Closing the commissioning window will terminate recording in the trend view and delete the recorded values.

### See also

Pretuning V1 (Page 113)

Fine tuning V1 (Page 115)

"Manual" mode V1 (Page 117)



### 4.2.2.2 Pretuning V1

The pretuning determines the process response to a jump change of the output value and searches for the point of inflection. The tuned PID parameters are calculated as a function of the maximum slope and dead time of the controlled system.

The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. The PID parameters are backed up before being recalculated.

#### Requirement

- The "PID\_Compact" instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- PID\_Compact is in "inactive" or "manual" mode.
- The setpoint may not be changed during controller tuning. PID\_Compact will otherwise be deactivated.
- The setpoint and the process value lie within the configured limits (see "Process value monitoring" configuration).
- The difference between setpoint and process value is greater than 30% of the difference between process value high limit and process value low limit.
- The distance between the setpoint and the process value is > 50% of the setpoint.

#### Procedure

To perform pretuning, follow these steps:

1. Double-click the "PID\_Compact > Commissioning" entry in the project tree.
2. Select the entry "Pretuning" in the "Tuning mode" drop-down list.
3. Click the "Start" icon.
  - An online connection will be established.
  - Value recording is started.
  - Pretuning is started.
  - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

---

#### Note

Click the "Stop" icon when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

---

## **Result**

If pretuning was performed without an error message, the PID parameters have been tuned. PID\_Compact switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If pretuning is not possible, PID\_Compact will change to "Inactive" mode.

## **See also**

Parameters State and sRet.i\_Mode V1 (Page 303)

Commissioning V1 (Page 112)

Fine tuning V1 (Page 115)

"Manual" mode V1 (Page 117)

### 4.2.2.3 Fine tuning V1

Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are optimized for the operating point from the amplitude and frequency of this oscillation. All PID parameters are recalculated on the basis of the findings. PID parameters from fine tuning usually have better master control and disturbance behavior than PID parameters from pretuning.

PID\_Compact automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The PID parameters are backed up before being recalculated.

#### Requirement

- The PID\_Compact instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- The setpoint and the process value lie within the configured limits (see "Process value monitoring" configuration).
- The control loop has stabilized at the operating point. The operating point is reached when the process value corresponds to the setpoint.
- No disturbances are expected.
- The setpoint may not be changed during controller tuning.
- PID\_Compact is in inactive mode, automatic mode or manual mode.

#### Process depends on initial situation

Fine tuning can be started in "inactive", "automatic" or "manual" mode. Fine tuning proceeds as follows when started in:

- Automatic mode

Start fine tuning in automatic mode if you wish to improve the existing PID parameters using controller tuning.

PID\_Compact will regulate using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.

- Inactive or manual mode

If the requirements for pretuning are met, pretuning is started. The PID parameters established will be used for adjustment until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. If pretuning is not possible, PID\_Compact will change to "Inactive" mode.

An attempt is made to reach the setpoint with a minimum or maximum output value if the process value for pretuning is already too near the setpoint. This can produce increased overshoot.

## Procedure

Proceed as follows to carry out "fine tuning":

1. Select the entry "Fine tuning" in the "Tuning mode" drop-down list.
2. Click the "Start" icon.
  - An online connection will be established.
  - Value recording is started.
  - The process of fine tuning is started.
  - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

---

### Note

Click the "Stop" icon in the "Tuning mode" group when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

---

## Result

The PID parameters will have been optimized if fine tuning has been executed without errors. PID\_Compact changes to automatic mode and uses the optimized parameters. The optimized PID parameters will be retained during power OFF and a restart of the CPU.

If errors occurred during "fine tuning", PID\_Compact will change to "inactive" mode.

## See also

Parameters State and sRet.i\_Mode V1 (Page 303)

Commissioning V1 (Page 112)

Pretuning V1 (Page 113)

"Manual" mode V1 (Page 117)

#### 4.2.2.4 "Manual" mode V1


The following section describes how you can use the "Manual" operating mode in the commissioning window of the "PID Compact" technology object.

##### Requirement

- The "PID\_Compact" instruction is called in a cyclic interrupt OB.
- An online connection to the CPU has been established and the CPU is in the "RUN" mode.
- The functions of the commissioning window have been enabled via the "Start" icon.

##### Procedure

Use "Manual mode" in the commissioning window if you want to test the process by specifying a manual value. To define a manual value, proceed as follows:

1. Select the check box "Manual mode" in the "Online status of the controller" area.  
PID\_Compact operates in manual mode. The most recent current output value remains in effect.
2. Enter the manual value in the "Output" field as a % value.
3. Click the control icon .

##### Result

The manual value is written to the CPU and immediately goes into effect.

---

##### Note

PID\_Compact continues to monitor the process value. If the process value limits are exceeded, PID\_Compact is deactivated.

---

Clear the "Manual mode" check box if the output value is to be specified again by the PID controller. The change to automatic mode is bumpless.

##### See also

Parameters State and sRet.i\_Mode V1 (Page 303)

Commissioning V1 (Page 112)

Pretuning V1 (Page 113)

Fine tuning V1 (Page 115)

### 4.2.3 Simulating PID\_Compact V1 with PLCSIM

---

#### Note

##### Simulation with PLCSIM

For the simulation with PLCSIM, the time behavior of the simulated PLC is not exactly identical to that of a "real" PLC. The actual cycle clock of a cyclic interrupt OB can have larger fluctuations with a simulated PLC than with "real" PLCs.

In the standard configuration, PID\_Compact determines the time between calls automatically and monitors them for fluctuations.

For a simulation of PID\_Compact with PLCSIM, for example, a sampling time error (ErrorBits = DW#16#00000800) can therefore be detected.

PID\_Compact switches to "Inactive" mode (State = 0) in this case.

To prevent this from happening, you should configure PID\_Compact for simulation with PLCSIM as follows:

- sb\_EnCyclEstimation = FALSE
  - sb\_EnCyclMonitoring = FALSE
  - sPid\_Calc.r\_Cycle: Assign the cycle clock of the calling cyclic interrupt OB in seconds to this tag.
-

## 4.3 Technology object PID\_Compact

The technology object PID\_Compact provides a continuous PID controller with integrated optimization. You can alternatively configure a pulse controller. Both manual and automatic mode are possible.

PID-Compact continuously acquires the measured process value within a control loop and compares it with the required setpoint. From the resulting control deviation, the instruction PID\_Compact calculates an output value by which the process value is adapted as quickly and stable as possible to the setpoint. The output value for the PID controller consists of three actions:

- **Proportional** action

The proportional action of the output value increases in proportion to the control deviation.

- **I** action

The integral action of the output value increases until the control deviation has been balanced.

- **D** action

The derivative action increases with the rate of change of control deviation. The process value is corrected to the setpoint as quickly as possible. The derivative action will be reduced again if the rate of change of control deviation drops.

The instruction PID\_Compact calculates the proportional, integral and derivative parameters for your controlled system during pretuning. Fine tuning can be used to tune the parameters further. You do not need to manually determine the parameters.

### Additional information

- Overview of software controller (Page 38)
- Add technology objects (Page 40)
- Configure technology objects (Page 45)
- Configuring PID\_Compact V2 (Page 76)
- Configuring PID\_Compact V1 (Page 99)

### FAQ

For more information, see the following FAQs in the Siemens Industry Online Support:

- Entry ID 79047707 (<https://support.industry.siemens.com/cs/ww/en/view/79047707>)

## Using PID\_3Step

### 5.1 Technology object PID\_3Step

The technology object PID\_3Step provides a PID controller with tuning for valves or actuators with integral response.

You can configure the following controllers:

- Three-point step controller with position feedback
- Three-point step controller without position feedback
- Valve controller with analog output value

PID\_3Step continuously acquires the measured process value within a control loop and compares it with the setpoint. From the resulting control deviation, PID\_3Step calculates an output value through which the process value reaches the setpoint as quickly and steadily as possible. The output value for the PID controller consists of three actions:

- **P**roportional action

The proportional action of the output value increases in proportion to the control deviation.

- **I** action

The integral action of the output value increases until the control deviation has been balanced.

- **D** action

The derivative action increases with the rate of change of control deviation. The process value is corrected to the setpoint as quickly as possible. The derivative action will be reduced again if the rate of change of control deviation drops.

The instruction PID\_3Step calculates the proportional, integral and derivative parameters for your controlled system during pretuning. Fine tuning can be used to tune the parameters further. You do not need to manually determine the parameters.

#### Additional information

- Overview of software controller (Page 38)
- Add technology objects (Page 40)
- Configure technology objects (Page 45)
- Configuring PID\_3Step V2 (Page 121)
- Configuring PID\_3Step V1 (Page 144)

#### Principle

For more information, see the following FAQs in the Siemens Industry Online Support:

- Entry ID 68011827 (<https://support.industry.siemens.com/cs/ww/en/view/68011827>)



## 5.2 PID\_3Step V2

### 5.2.1 Configuring PID\_3Step V2

#### 5.2.1.1 Basic settings V2

##### Introduction V2

Configure the following properties of the "PID\_3Step" technology object under "Basic settings" in the Inspector window or in the configuration window:

- Physical quantity
- Control logic
- Start-up behavior after reset
- Setpoint (only in the Inspector window)
- Process value (only in the Inspector window)
- Output value (only in the Inspector window)
- Position feedback (only in the Inspector window)

##### Setpoint, process value, output value and position feedback

You can only configure the setpoint, process value, output value and position feedback in the Inspector window of the programming editor. Select the source for each value:

- Instance DB  
The value saved in the instance DB is used.  
Value must be updated in the instance DB by the user program.  
There should be no value at the instruction.  
Change via HMI possible.
- Instruction  
The value connected to the instruction is used.  
The value is written to the instance DB each time the instruction is called.  
No change via HMI possible.

## Control mode V2

### Physical quantity

Select the physical quantity and unit of measurement for setpoint, process value, and disturbance variable in the "Controller type" group. Setpoint, process value, and disturbance variable is displayed in this unit of measurement.

### Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic.

PID\_3Step does not work with negative proportional gain. Select the check box "Invert control logic" to reduce the process value with a higher output value.

Examples

- Opening the drain valve will reduce the level of a container's contents.
- Increasing cooling will reduce the temperature.

### Startup characteristics

1. To switch to "Inactive" mode after CPU restart, clear the "Activate Mode after CPU restart" check box.

To switch to the operating mode saved in the Mode parameter after CPU restart, select the "Activate Mode after CPU restart" check box.

2. In the "Set Mode to" drop-down list, select the mode that is to be enabled after a complete download to the device.

After a complete download to the device, PID\_3Step starts in the selected operating mode. With each additional restart, PID\_3Step starts in the mode that was last saved in Mode.

Example

You have selected the "Activate Mode after CPU restart" check box and the entry "Pretuning" in the "Set Mode to" list. After a complete download to the device, PID\_3Step starts in the "Pretuning" mode. If pretuning is still active, PID\_3Step starts in "Pretuning" mode again after restart of the CPU. If pretuning was successfully completed and automatic mode is active, PID\_3Step starts in "Automatic mode" after restart of the CPU.

## Setpoint V2

### Procedure

Proceed as follows to define a fixed setpoint:

1. Select "Instance DB".
2. Enter a setpoint, e.g. 80° C.
3. Delete any entry in the instruction.

Proceed as follows to define a variable setpoint:

1. Select "Instruction".
2. Enter the name of the REAL variable in which the setpoint is saved.

Program-controlled assignment of various values to the REAL variable is possible, for example for the time controlled change of the setpoint.

## Process value V2

PID\_3Step will scale the value of the analog input to the physical quantity if you use the analog input value directly.

You will need to write a program for processing if you wish first to process the analog input value. The process value is, for example, not directly proportional to the value at the analog input. The processed process value must be in floating point format.

### Procedure

Proceed as follows to use the analog input value without processing:

1. Select the entry "Input\_PER" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the address of the analog input.

Proceed as follows to use the processed process value in floating point format:

1. Select the entry "Input" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the name of the variable in which the processed process value is saved.

## Position feedback V2

Position feedback configuration depends upon the actuator used.

- Actuator without position feedback
- Actuator with digital endstop signals
- Actuator with analog position feedback
- Actuator with analog position feedback and endstop signals

### Actuator without position feedback

Proceed as follows to configure PID\_3Step for an actuator without position feedback:

1. Select the entry "No Feedback" in the drop-down list "Feedback".

### Actuator with digital endstop signals

Proceed as follows to configure PID\_3Step for an actuator with endstop signals:

1. Select the entry "No Feedback" in the drop-down list "Feedback".
2. Activate the "Actuator endstop signals" check box.
3. Select "Instruction" as source for Actuator\_H and Actuator\_L.
4. Enter the addresses of the digital inputs for Actuator\_H and Actuator\_L.

### Actuator with analog position feedback

Proceed as follows to configure PID\_3Step for an actuator with analog position feedback:

1. Select the entry "Feedback" or "Feedback\_PER" in the drop-down list "Feedback".
  - Use the analog input value for Feedback\_PER. Configure Feedback\_PER scaling in the actuator settings.
  - Process the analog input value for Feedback using your user program.
2. Select "Instruction" as source.
3. Enter the address of the analog input or the variable of your user program.

### Actuator with analog position feedback and endstop signals

Proceed as follows to configure PID\_3Step for an actuator with analog position feedback and endstop signals:

1. Select the entry "Feedback" or "Feedback\_PER" in the drop-down list "Feedback".
2. Select "Instruction" as source.
3. Enter the address of the analog input or the variable of your user program.
4. Activate the "Actuator endstop signals" check box.
5. Select "Instruction" as source for Actuator\_H and Actuator\_L.
6. Enter the addresses of the digital inputs for Actuator\_H and Actuator\_L.

## Output value V2

PID\_3Step offers an analog output value (Output\_PER) and digital output values (Output\_UP, Output\_DN). Your actuator will determine which output value you use.

- Output\_PER

The actuator has a relevant motor transition time and is triggered via an analog output and controlled with a continuous signal, e.g. 0...10 V or 4...20 mA. The value at Output\_PER corresponds to the target position of the valve, e.g. Output\_PER = 13824, when the valve is to be opened by 50%.

For auto-tuning and anti windup behavior, for example, PID\_3Step takes into consideration that the analog output value has a delayed effect on the process due to the motor transition time. If no relevant motor transition time is in effect in your process (e.g. with solenoid valves), so that the output value has a direct and full effect on the process, use PID\_Compact instead.

- Output\_UP, Output\_DN

The actuator has a relevant motor transition time and is controlled by two digital outputs. Output\_UP moves the valve in the open state direction. Output\_DN moves the valve in the closed state direction.

The motor transition time is taken into consideration in the calculation of the analog output value as well as in the calculation of the digital output values. It is mainly required for correct operation during auto-tuning and the anti-windup behavior. You should therefore configure the motor transition time under "Actuator settings" with the value that the motor requires to move the actuator from the closed to the opened state.

## Procedure

Proceed as follows to use the analog output value:

1. Select the entry "Output (analog)" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the address of the analog output.

Proceed as follows to use the digital output value:

1. Select the entry "Output (digital)" in the drop-down list "Output".
2. Select "Instruction" for Output\_UP and Output\_DN.
3. Enter the addresses of the digital outputs.

Proceed as follows to process the output value using the user program:

1. Select the entry corresponding to the actuator in the drop-down list "Output".
2. Select "Instruction".
3. Enter the name of the tag you are using to process the output value.
4. Transfer the processed output value to the actuator by means of an analog or digital CPU output.

### 5.2.1.2 Process value settings V2

#### Process value scaling V2

If you have configured the use of Input\_PER in the basic setting, you must convert the value of the analog input to the physical quantity of the process value. The current configuration is displayed in the Input\_PER display.

Input\_PER will be scaled using a low and high value pair if the process value is directly proportional to the value of the analog input.

#### Procedure

To scale the process value, follow these steps:

1. Enter the low pair of values in the "Scaled low process value" and "Low" text boxes.
2. Enter the high pair of values in the "Scaled high process value" and "High" input boxes.

Default settings for the value pairs are stored in the hardware configuration. To use the value pairs from the hardware configuration, follow these steps:

1. Select the PID\_3Step instruction in the programming editor.
2. Interconnect Input\_PER with an analog input in the basic settings.
3. Click the "Automatic setting" button in the process value settings.

The existing values will be overwritten with the values from the hardware configuration.

#### Process value limits V2

You must specify an appropriate absolute high limit and low limit for the process value as limit values for your controlled system. As soon as the process value violates these limits, an error occurs (ErrorBits = 0001h). Tuning is canceled when the process value limits are violated. You can specify how PID\_3Step responds to errors in automatic mode in the actuator settings.

### 5.2.1.3 Final controlling element settings V2

#### Final controlling element V2

##### Actuator-specific times

Configure the motor transition time and the minimum ON and OFF times to prevent damage to the actuator. You can find the specifications in the actuator data sheet.

The motor transition time is the time in seconds the motor requires to move the actuator from the closed to the opened state. You can measure the motor transition time during commissioning.

The motor transition time is taken into consideration in the calculation of the analog output value as well as in the calculation of the digital output values. It is mainly required for correct operation during auto-tuning and the anti-windup behavior.

If no relevant motor transition time is in effect in your process (e.g. with solenoid valves), so that the output value has a direct and full effect on the process, use PID\_Compact instead.

The motor transition time is retentive. If you enter the motor transition time manually, you must completely download PID\_3Step.

Downloading technology objects to device (Page 72)

If you are using "Output\_UP" or "Output\_DN", you can reduce the switching frequency with the minimum on and minimum OFF time.

The on or off times calculated are totaled in automatic mode and only become effective when the sum is greater than or equal to the minimum on or OFF time.

Manual\_UP = TRUE or Manual\_DN = TRUE in manual mode operates the actuator for at least the minimum ON or OFF time.

If you have selected the analog output value Output\_PER, the minimum ON time and the minimum OFF time are not evaluated and cannot be changed.

### Reaction to error

PID\_3Step is preset so that the controller stays active in most cases in the event of an error. If errors occur frequently in controller mode, this default reaction has a negative effect on the control response. In this case, check the Errorbits parameter and eliminate the cause of the error.

<b>NOTICE</b>
<b>Your system may be damaged.</b>
If you output "Current value while error pending" or "Substitute output value while error pending" in the event of an error, PID_3Step remains in automatic mode even if the process value limits are violated. This may damage your system.
It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

PID\_3Step generates a programmable output value in the case of an error:

- Current value

PID\_3Step is switched off and no longer modifies the actuator position.

- Current value for error while error is pending

The controller functions of PID\_3Step are switched off and the position of the actuator is no longer changed.

If the following errors occur in automatic mode, PID\_3Step returns to automatic mode as soon as the errors are no longer pending.

- 0002h: Invalid value at Input\_PER parameter.
- 0200h: Invalid value at Input parameter.
- 0400h: Calculation of output value failed.
- 1000h: Invalid value at Setpoint parameter.
- 2000h: Invalid value at Feedback\_PER parameter.
- 4000h: Invalid value at Feedback parameter.
- 8000h: Error during digital position feedback.
- 20000h: Invalid value at SavePosition tag.



If one or more of the following errors occur, PID\_3Step stays in automatic mode:

- 0001h: The Input parameter is outside the process value limits.
- 0800h: Sampling time error
- 40000h: Invalid value at Disturbance parameter.

PID\_3Step remains in manual mode if an error occurs in manual mode.

If an error occurs during tuning or transition time measurement, PID\_3Step switches to the mode in which tuning or transition time measurement was started. Only in the event of the following error is tuning not aborted:

- 0020h: Pretuning is not permitted during fine tuning.
- Substitute output value  
PID\_3Step moves the actuator to the substitute output value and then switches off.
- Substitute output value while error is pending  
PID\_3Step moves the actuator to the substitute output value. When the substitute output value is reached, PID\_3Step reacts as it does with "Current value for while error is pending".

Enter the substitute output value in "%".

Only substitute output values 0% and 100% can be approached precisely in the case of actuators without analog position feedback. A substitute output value not equal to 0% or 100% is approached via an internally simulated position feedback. This procedure does not, however, allow the exact approach of substitute output value.

All substitute output values can be approached precisely with actuators with analog position feedback.

## Scaling V2 position feedback

### Scaling position feedback

If you have configured the use of Feedback\_PER in the basic settings, you will need to convert the value of the analog input into %. The current configuration will be displayed in the "Feedback" display.

Feedback\_PER is scaled using a low and high value pair.

1. Enter the low pair of values in the "Low endstop" and "Low" input boxes.
2. Enter the high pair of values in the "High endstop" and "High" input boxes.

"Low endstop" must be less than "High endstop"; "Low" must be less than "High".

The valid values for "High endstop" and "Low endstop" depend upon:

- No Feedback, Feedback, Feedback\_PER
- Output (analog), Output (digital)

Output	Feedback	Low endstop	High endstop
Output (digital)	No Feedback	Cannot be set (0.0%)	Cannot be set (100.0%)
Output (digital)	Feedback	-100.0% or 0.0%	0.0% or +100.0%
Output (digital)	Feedback_PER	-100.0% or 0.0%	0.0% or +100.0%
Output (analog)	No Feedback	Cannot be set (0.0%)	Cannot be set (100.0%)
Output (analog)	Feedback	-100.0% or 0.0%	0.0% or +100.0%
Output (analog)	Feedback_PER	-100.0% or 0.0%	0.0% or +100.0%

## Output value limits V2

### Limiting the output value

You can exceed or undershoot the output value limits during the transition time measurement and with mode = 10. The output value is limited to these values in all other modes.

Enter the absolute output value limits in the "Output value high limit" and "Output value low limit" input boxes. The output value limits must be within "Low endstop" and "High endstop".

If no Feedback is available and Output (digital) is set, you cannot limit the output value. Output\_UP and Output\_DN are then reset upon Actuator\_H = TRUE or Actuator\_L = TRUE. If no endstop signals are available, Output\_UP and Output\_DN are reset after a travel time of 150% of the motor actuating time.

The default value of 150% can be adjusted using the tagConfig.VirtualActuatorLimit. As of PID\_3Step Version 2.3 the monitoring and limiting of the travel time can be deactivated with Config.VirtualActuatorLimit = 0.0.

## 5.2.1.4 Advanced settings V2

### Actual value monitoring V2

Configure a warning high and low limit for the process value in the "Process value monitoring" configuration window. If one of the warning limits is exceeded or undershot during operation, a warning will be displayed at the PID\_3Step instruction:

- At the InputWarning\_H output parameter if the warning high limit has been exceeded
- At the InputWarning\_L output parameter if the warning low limit has been undershot

The warning limits must be within the process value high and low limits.

The process value high and low limits will be used if you do not enter values.

### Example

Process value high limit = 98° C; warning high limit = 90° C

Warning low limit = 10° C; process value low limit = 0° C

PID\_3Step will respond as follows:

Process value	InputWarning_H	InputWarning_L	Error-Bits	Operating mode
> 98° C	TRUE	FALSE	0001h	As configured
≤ 98° C and > 90° C	TRUE	FALSE	0000h	Automatic mode
≤ 90° C and ≥ 10° C	FALSE	FALSE	0000h	Automatic mode
< 10° C and ≥ 0° C	FALSE	TRUE	0000h	Automatic mode
< 0° C	FALSE	TRUE	0001h	As configured

In the actuator settings, you can configure the response of PID\_3Step when the process value high limit or low limit is violated.

### PID parameters V2

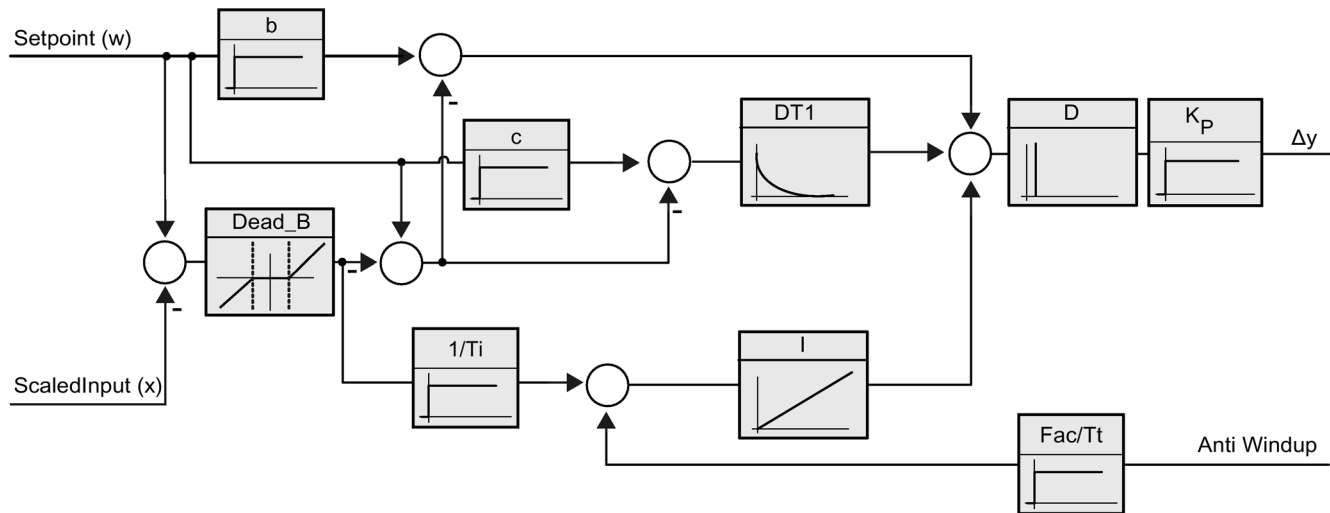
The PID parameters are displayed in the "PID Parameters" configuration window. The PID parameters will be adapted to your controlled system during controller tuning. You do not need to enter the PID parameters manually.

The PID algorithm operates according to the following equation:

$$\Delta y = K_p \cdot s \cdot \left[ (b \cdot w - x) + \frac{1}{T_i \cdot s} (w - x) + \frac{T_d \cdot s}{a \cdot T_d \cdot s + 1} (c \cdot w - x) \right]$$

Symbol	Description
$\Delta y$	Output value of the PID algorithm
$K_p$	Proportional gain
$s$	Laplace operator
$b$	Proportional action weighting
$w$	Setpoint
$x$	Process value
$T_i$	Integration time
$a$	Derivative delay coefficient (derivative delay $T_1 = a \times T_D$ )
$T_D$	Derivative action time
$c$	Derivative action weighting

The diagram below illustrates the integration of the parameters into the PID algorithm:



All PID parameters are retentive. If you enter the PID parameters manually, you must completely download PID\_3Step.

Downloading technology objects to device (Page 72)

### Proportional gain

The value specifies the proportional gain of the controller. PID\_3Step does not work with a negative proportional gain. Control logic is inverted under Basic settings > Controller type.

### Integration time

The integration time determines the time behavior of the integral action. The integral action is deactivated with integration time = 0.0.

### Derivative action time

The derivative action time determines the time behavior of the derivative action. Derivative action is deactivated with derivative action time = 0.0.

### Derivative delay coefficient

The derivative delay coefficient delays the effect of the derivative action.

Derivative delay = derivative action time × derivative delay coefficient

- 0.0: Derivative action is effective for one cycle only and therefore almost not effective.
- 0.5: This value has proved useful in practice for controlled systems with **one** dominant time constant.
- > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed.

### Proportional action weighting

The proportional action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Proportional action for setpoint change is fully effective
- 0.0: Proportional action for setpoint change is not effective

The proportional action is always fully effective when the process value is changed.

### Derivative action weighting

The derivative action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Derivative action is fully effective upon setpoint change
- 0.0: Derivative action is not effective upon setpoint change

The derivative action is always fully effective when the process value is changed.

### **PID algorithm sampling time**

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the PID\_3Step sampling time. All other functions of PID\_3Step are executed at every call.

### **Dead band width**

The deadband suppresses the noise component in the steady controller state. The dead band width specifies the size of the dead band. The dead band is off if the dead band width is 0.0.

If values not equal to 1.0 are configured for the proportional action weighting or the derivative action weighting, setpoint changes even within the dead zone affect the output value. Process value changes within the dead zone do not affect the output value, regardless of the weighting.

## 5.2.2 Commissioning PID\_3Step V2

### 5.2.2.1 Pretuning V2

The pretuning determines the process response to a pulse of the output value and searches for the point of inflection. The tuned PID parameters are calculated as a function of the maximum slope and dead time of the controlled system. You obtain the best PID parameters when you perform pretuning and fine tuning.

The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. This is most likely the case in operating modes "Inactive" and "manual mode". The PID parameters are backed up before being recalculated.

The setpoint is frozen during pretuning.

#### Requirement

- The PID\_3Step instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- Reset = FALSE
- The motor transition time has been configured or measured.
- PID\_3Step is in one of the following modes: "Inactive", "Manual mode", or "Automatic mode".
- The setpoint and the process value lie within the configured limits (see "Process value settings" configuration).

## Procedure

To perform pretuning, follow these steps:

1. Double-click the "PID\_3Step > Commissioning" entry in the project tree.
2. Select the entry "Pretuning" in the "Tuning mode" drop-down list in the working area "Tuning".
3. Click the "Start" icon.
  - An online connection will be established.
  - Value recording is started.
  - Pretuning is started.
  - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

---

### Note

Click the "Stop" icon when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

---

## Result

If pretuning was performed without an error message, the PID parameters have been tuned. PID\_3Step switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If pretuning is not possible, PID\_3Step responds with the configured reaction to errors.



### 5.2.2.2 Fine tuning V2

Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are tuned for the operating point from the amplitude and frequency of this oscillation. All PID parameters are recalculated from the results. PID parameters from fine tuning usually have better master control and disturbance characteristics than PID parameters from pretuning. You obtain the best PID parameters when you perform pretuning and fine tuning.

PID\_3Step automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The PID parameters are backed up before being recalculated.

The setpoint is frozen during fine tuning.

#### Requirement

- The PID\_3Step instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- Reset = FALSE
- The motor transition time has been configured or measured.
- The setpoint and the process value lie within the configured limits (see "Process value settings" configuration).
- The control loop has stabilized at the operating point. The operating point is reached when the process value corresponds to the setpoint.
- No disturbances are expected.
- PID\_3Step is in inactive mode, automatic mode or manual mode.

#### Process depends on initial situation

Fine tuning proceeds as follows when started from:

- Automatic mode

Start fine tuning from automatic mode if you wish to improve the existing PID parameters through tuning.

PID\_3Step controls the system using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.

- Inactive or manual mode

Pretuning is always started first. The determined PID parameters will be used for control until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.

## Procedure

To perform fine tuning, follow these steps:

1. Select the entry "Fine tuning" in the "Tuning mode" drop-down list.
2. Click the "Start" icon.
  - An online connection will be established.
  - Value recording is started.
  - The process of fine tuning is started.
  - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

---

### Note

Click the "Stop" icon in the "Tuning mode" group when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

---

## Result

If no errors occurred during fine tuning, the PID parameters have been tuned. PID\_3Step switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If errors occurred during fine tuning, PID\_3Step responds with the configured response to errors.

### 5.2.2.3 Commissioning with manual PID parameters V2

#### Requirement

- The PID\_3Step instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- Reset = FALSE
- The motor transition time has been configured or measured.
- PID\_3Step is in "inactive" mode.
- The setpoint and the process value lie within the configured limits (see "Process value settings" configuration).

#### Procedure

Proceed as follows to commission PID\_3Step with manual PID parameters:

1. Double-click on "PID\_3Step > Configuration" in the project tree.
2. Click on "Advanced settings > PID Parameters" in the configuration window.
3. Select the check box "Enable direct input".
4. Enter the PID parameters.
5. Double-click the "PID\_3Step > Commissioning" entry in the project tree.
6. Establish an online connection to the CPU.
7. Load the PID parameters to the CPU.
8. Click the "Start PID\_3Step" icon.

#### Result

PID\_3Step changes to automatic mode and controls using the current PID parameters.

#### See also

PID parameters V2 (Page 132)

### 5.2.2.4 Measuring the motor transition time V2

#### Introduction

PID\_3Step requires the motor transition time to be as accurate as possible for good controller results. The data in the actuator documentation contains average values for this type of actuator. The value for the specific actuator used may differ.

You can measure the motor transition time during commissioning if you are using actuators with position feedback or endstop signals. The output value limits are not taken into consideration during the motor transition time measurement. The actuator can travel to the high or the low endstop.

The motor transition time cannot be measured if neither position feedback nor endstop signals are available.

#### Actuators with analog position feedback

Proceed as follows to measure motor transition time with position feedback:

##### Requirement

- Feedback or Feedback\_PER has been selected in the basic settings and the signal has been connected.
  - An online connection to the CPU has been established.
1. Select the "Use position feedback" check box.
  2. Enter the position to which the actuator is to be moved in the "Target position" input field.  

The current position feedback (starting position) will be displayed. The difference between "Target position" and "Position feedback" must be at least 50% of the valid output value range.
  3. Click the "Start" icon.

## Result


The actuator is moved from the starting position to the target position. Time measurement starts immediately and ends when the actuator reaches the target position. The motor transition time is calculated according to the following equation:

Motor transition time = (output value high limit – output value low limit) × Measuring time / AMOUNT (target position – starting position).

The progress and status of transition time measurement are displayed. The transition time measured is saved in the instance data block on the CPU and displayed in the "Measured transition time" field. When the transition time measurement is ended and ActivateRecoverMode = TRUE, PID\_3Step switches to the operating mode from which the transition time measurement was started. If the transition time measurement is ended and ActivateRecoverMode = FALSE, PID\_3Step changes to "Inactive" mode.

---

### Note

Click on the icon  "Upload measured transition time" to load the motor transition time measured to the project.

---

## Actuators with endstop signals

Proceed as follows to measure the transition time of actuators with endstop signals:

### Requirement

- The "Endstop signals" check box in the basic settings has been selected and Actuator\_H and Actuator\_L are connected.
- An online connection to the CPU has been established.

Proceed as follows to measure motor transition time with endstop signals:

1. Select the "Use actuator endstop signals" check box.
2. Select the direction in which the actuator is to be moved.
  - Open - Close - Open  
The actuator is moved first to the high endstop, then to the low endstop and then back to the high endstop.
  - Close - Open - Close  
The actuator is moved first to the low endstop, then to the high endstop and then back to the low endstop.
3. Click the "Start" icon.

## Result

The actuator is moved in the selected direction. Time measurement will start once the actuator has reached the first endstop and will end when the actuator reaches this endstop for the second time. The motor transition time is equal to the time measured divided by two.

The progress and status of transition time measurement are displayed. The transition time measured is saved in the instance data block on the CPU and displayed in the "Measured transition time" field. When the transition time measurement is ended and `ActivateRecoverMode = TRUE`, PID\_3Step switches to the operating mode from which the transition time measurement was started. If the transition time measurement is ended and `ActivateRecoverMode = FALSE`, PID\_3Step changes to "Inactive" mode.

## Canceling transition time measurement

PID\_3Step switches to "Inactive" mode if you cancel transition time measurement by pressing the Stop button.

### 5.2.3 Simulating PID\_3Step V2 with PLCSIM

---

#### Note

##### Simulation with PLCSIM

The simulation of PID\_3Step V2.x with PLCSIM for CPU S7-1200 is not supported.

PID\_3Step V2.x can only be simulated for CPU S7-1500 with PLCSIM.

For the simulation with PLCSIM, the time behavior of the simulated PLC is not exactly identical to that of a "real" PLC. The actual cycle clock of a cyclic interrupt OB can have larger fluctuations with a simulated PLC than with "real" PLCs.

In the standard configuration, PID\_3Step determines the time between calls automatically and monitors them for fluctuations.

For a simulation of PID\_3Step with PLCSIM, for example, a sampling time error ((ErrorBits = DW#16#00000800) can therefore be detected.

This results in ongoing tuning being aborted.

The response in automatic mode depends on the value of the ActivateRecoverMode tag.

To prevent this from happening, you should configure PID\_3Step for simulation with PLCSIM as follows:

- CycleTime.EnEstimation = FALSE
  - CycleTime.EnMonitoring = FALSE
  - CycleTime.Value: Assign the cycle clock of the calling cyclic interrupt OB in seconds to this tag.
-

## 5.3 PID\_3Step V1

### 5.3.1 Configuring PID\_3Step V1

#### 5.3.1.1 Basic settings V1

##### Introduction V1

Configure the following properties of the "PID\_3Step" technology object under "Basic settings" in the Inspector window or in the configuration window:

- Physical quantity
- Control logic
- Start-up behavior after reset
- Setpoint (only in the Inspector window)
- Process value (only in the Inspector window)
- Output value (only in the Inspector window)
- Position feedback (only in the Inspector window)

##### Setpoint, process value, output value and position feedback

You can only configure the setpoint, process value, output value and position feedback in the Inspector window of the programming editor. Select the source for each value:

- Instance DB
  - The value saved in the instance DB is used.
  - Value must be updated in the instance DB by the user program.
  - There should be no value at the instruction.
  - Change via HMI possible.
- Instruction
  - The value connected to the instruction is used.
  - The value is written to the instance DB each time the instruction is called.
  - No change via HMI possible.



## Control mode V1

### Physical quantity

Select the unit of measurement and physical quantity for the setpoint and process value in the "Controller type" group. The setpoint and process value will be displayed in this unit.

### Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic.

PID\_3Step does not work with negative proportional gain. Select the check box "Invert control logic" to reduce the process value with a higher output value.

Examples

- Opening the drain valve will reduce the level of a container's contents.
- Increasing cooling will reduce the temperature.

### Start-up behavior after reset

To change straight to the last active mode after restarting the CPU, select the "Enable last mode after CPU restart" check box.

PID\_3Step will remain in "Inactive" mode if the check box is cleared.

## Setpoint V1

### Procedure

Proceed as follows to define a fixed setpoint:

1. Select "Instance DB".
2. Enter a setpoint, e.g. 80° C.
3. Delete any entry in the instruction.

Proceed as follows to define a variable setpoint:

1. Select "Instruction".
2. Enter the name of the REAL variable in which the setpoint is saved.

Program-controlled assignment of various values to the REAL variable is possible, for example for the time controlled change of the setpoint.

## Process value V1

PID\_3Step will scale the value of the analog input to the physical quantity if you use the analog input value directly.

You will need to write a program for processing if you wish first to process the analog input value. The process value is, for example, not directly proportional to the value at the analog input. The processed process value must be in floating point format.

## Procedure

Proceed as follows to use the analog input value without processing:

1. Select the entry "Input\_PER" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the address of the analog input.

Proceed as follows to use the processed process value in floating point format:

1. Select the entry "Input" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the name of the variable in which the processed process value is saved.

## Position feedback V1

Position feedback configuration depends upon the actuator used.

- Actuator without position feedback
- Actuator with digital endstop signals
- Actuator with analog position feedback
- Actuator with analog position feedback and endstop signals

## Actuator without position feedback

Proceed as follows to configure PID\_3Step for an actuator without position feedback:

1. Select the entry "No Feedback" in the drop-down list "Feedback".

## Actuator with digital endstop signals

Proceed as follows to configure PID\_3Step for an actuator with endstop signals:

1. Select the entry "No Feedback" in the drop-down list "Feedback".
2. Activate the "Actuator endstop signals" check box.
3. Select "Instruction" as source for Actuator\_H and Actuator\_L.
4. Enter the addresses of the digital inputs for Actuator\_H and Actuator\_L.

### Actuator with analog position feedback

Proceed as follows to configure PID\_3Step for an actuator with analog position feedback:

1. Select the entry "Feedback" or "Feedback\_PER" in the drop-down list "Feedback".
  - Use the analog input value for Feedback\_PER. Configure Feedback\_PER scaling in the actuator settings.
  - Process the analog input value for Feedback using your user program.
2. Select "Instruction" as source.
3. Enter the address of the analog input or the variable of your user program.

### Actuator with analog position feedback and endstop signals

Proceed as follows to configure PID\_3Step for an actuator with analog position feedback and endstop signals:

1. Select the entry "Feedback" or "Feedback\_PER" in the drop-down list "Feedback".
2. Select "Instruction" as source.
3. Enter the address of the analog input or the variable of your user program.
4. Activate the "Actuator endstop signals" check box.
5. Select "Instruction" as source for Actuator\_H and Actuator\_L.
6. Enter the addresses of the digital inputs for Actuator\_H and Actuator\_L.

### Output value V1

PID\_3Step offers an analog output value (Output\_PER) and digital output values (Output\_UP, Output\_DN). Your actuator will determine which output value you use.

- Output\_PER

The actuator has a relevant motor transition time and is triggered via an analog output and controlled with a continuous signal, e.g. 0...10 V or 4...20 mA. The value at Output\_PER corresponds to the target position of the valve, e.g. Output\_PER = 13824, when the valve is to be opened by 50%.

For auto-tuning and anti windup behavior, for example, PID\_3Step takes into consideration that the analog output value has a delayed effect on the process due to the motor transition time. If no relevant motor transition time is in effect in your process (e.g. with solenoid valves), so that the output value has a direct and full effect on the process, use PID\_Compact instead.

- Output\_UP, Output\_DN

The actuator has a relevant motor transition time and is controlled by two digital outputs. Output\_UP moves the valve in the open state direction. Output\_DN moves the valve in the closed state direction.

The motor transition time is taken into consideration in the calculation of the analog output value as well as in the calculation of the digital output values. It is mainly required for correct operation during auto-tuning and the anti-windup behavior. You should therefore configure the motor transition time under "Actuator settings" with the value that the motor requires to move the actuator from the closed to the opened state.

## Procedure

Proceed as follows to use the analog output value:

1. Select the entry "Output (analog)" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the address of the analog output.

Proceed as follows to use the digital output value:

1. Select the entry "Output (digital)" in the drop-down list "Output".
2. Select "Instruction" for Output\_UP and Output\_DN.
3. Enter the addresses of the digital outputs.

Proceed as follows to process the output value using the user program:

1. Select the entry corresponding to the actuator in the drop-down list "Output".
2. Select "Instruction".
3. Enter the name of the tag you are using to process the output value.
4. Transfer the processed output value to the actuator by means of an analog or digital CPU output.

### 5.3.1.2 Process value settings V1

Configure the scaling of your process value and specify the process value absolute limits in the "Process value settings" configuration window.

#### Scaling the process value

If you have configured the use of Input\_PER in the basic settings, you will need to convert the value of the analog input into the physical quantity of the process value. The current configuration will be displayed in the Input\_PER display.

Input\_PER will be scaled using a low and high value pair if the process value is directly proportional to the value of the analog input.

1. Enter the low pair of values in the "Scaled low process value" and "Low" input fields.
2. Enter the high pair of values in the "Scaled high process value" and "High" input boxes.

Default settings for the value pairs are saved in the hardware configuration. Proceed as follows to use the value pairs from the hardware configuration:

1. Select the instruction PID\_3Step in the programming editor.
2. Connect Input\_PER to an analog input in the basic settings.
3. Click on the "Automatic setting" button in the process value settings.

The existing values will be overwritten with the values from the hardware configuration.

#### Monitoring process value

Specify the absolute high and low limit of the process value. You must enter reasonable limits for your controlled system. Reasonable limits are important during optimization to obtain optimal PID parameters. The default for the "High limit process value" is 120 %. At the I/O input, the process value can be a maximum of 18% higher than the standard range (overrange). This setting ensures that an error is no longer signaled due to a violation of the "Process value high limit". Only a wire-break and a short-circuit are recognized and PID\_3Step reacts according to the configured reaction to error.

#### NOTICE

##### Your system may be damaged.

If you set very high process value limits (for example  $-3.4 \cdot 10^{38} \dots +3.4 \cdot 10^{38}$ ), process value monitoring will be disabled. Your system may then be damaged if an error occurs. You need to configure useful process value limits for your controlled system.

### 5.3.1.3 V1 final controlling element setting

#### Actuator-specific times

Configure the motor transition time and the minimum ON and OFF times to prevent damage to the actuator. You can find the specifications in the actuator data sheet.

The motor transition time is the time in seconds the motor requires to move the actuator from the closed to the opened state. The maximum time that the actuator is moved in one direction is 110% of the motor transition time. You can measure the motor transition time during commissioning.

The motor transition time is taken into consideration in the calculation of the analog output value as well as in the calculation of the digital output values. It is mainly required for correct operation during auto-tuning and the anti-windup behavior.

If no relevant motor transition time is in effect in your process (e.g. with solenoid valves), so that the output value has a direct and full effect on the process, use PID\_Compact instead.

If you are using "Output\_UP" or "Output\_DN", you can reduce the switching frequency with the minimum on and minimum OFF time.

The on or off times calculated are totaled in automatic mode and only become effective when the sum is greater than or equal to the minimum on or OFF time.

A rising edge at Manual\_UP or Manual\_DN in manual mode will operate the actuator for at least the minimum on or OFF time.

If you have selected the analog output value Output\_PER, the minimum ON time and the minimum OFF time are not evaluated and cannot be changed.

#### Reaction to error

PID\_3Step is preset so that the controller stays active in most cases in the event of an error. If errors occur frequently in controller mode, this default reaction has a negative effect on the control response. In this case, check the Errorbits parameter and eliminate the cause of the error.

PID\_3Step generates a programmable output value in response to an error:

- Current value

PID\_3Step is switched off and no longer modifies the actuator position.

- Current value for error while error is pending

The controller functions of PID\_3Step are switched off and the position of the actuator is no longer changed.

If the following errors occur in automatic mode, PID\_3Step returns to automatic mode as soon as the errors are no longer pending.

- 0002h: Invalid value at Input\_PER parameter.
- 0200h: Invalid value at Input parameter.
- 0800h: Sampling time error
- 1000h: Invalid value at Setpoint parameter.
- 2000h: Invalid value at Feedback\_PER parameter.
- 4000h: Invalid value at Feedback parameter.
- 8000h: Error during digital position feedback.

If one of these error occurs in manual mode, PID\_3Step remains in manual mode.

If an error occurs during the tuning or transition time measurement, PID\_3Step is switched off.

- Substitute output value

PID\_3Step moves the actuator to the substitute output value and then switches off.

- Substitute output value while error is pending

PID\_3Step moves the actuator to the substitute output value. When the substitute output value is reached, PID\_3Step reacts as it does with "Current value for while error is pending".

Enter the substitute output value in "%".

Only substitute output values 0% and 100% can be approached precisely in the case of actuators without analog position feedback. The actuator is moved in one direction at 110% of the motor transition time to ensure the high or low endstop is reached. There endstop signals take priority. A substitute output value not equal to 0% or 100% is approached via an internally simulated position feedback. This procedure does not, however, allow the exact approach of substitute output value.

All substitute output values can be approached precisely with actuators with analog position feedback.

### Scaling position feedback

If you have configured the use of Feedback\_PER in the basic settings, you will need to convert the value of the analog input into %. The current configuration will be displayed in the "Feedback" display.

Feedback\_PER is scaled using a low and high value pair.

1. Enter the low pair of values in the "Low endstop" and "Low" input boxes.
2. Enter the high pair of values in the "High endstop" and "High" input boxes.

"Low endstop" must be less than "High endstop"; "Low" must be less than "High".

The valid values for "High endstop" and "Low endstop" depend upon:

- No Feedback, Feedback, Feedback\_PER
- Output (analog), Output (digital)

Output	Feedback	Low endstop	High endstop
Output (digital)	No Feedback	Cannot be set (0.0%)	Cannot be set (100.0%)
Output (digital)	Feedback	-100.0% or 0.0%	0.0% or +100.0%
Output (digital)	Feedback_PER	-100.0% or 0.0%	0.0% or +100.0%
Output (analog)	No Feedback	Cannot be set (0.0%)	Cannot be set (100.0%)
Output (analog)	Feedback	-100.0% or 0.0%	0.0% or +100.0%
Output (analog)	Feedback_PER	-100.0% or 0.0%	0.0% or +100.0%

### Limiting the output value

You can only exceed or undershoot the output value limits during the transition time measurement. The output value is limited to these values in all other modes.

Enter the absolute output value limits in the "Output value high limit" and "Output value low limit" input boxes. The output value limits must be within "Low endstop" and "High endstop".

If no Feedback is available and Output (digital) is set, you cannot limit the output value. The digital outputs are reset with Actuator\_H = TRUE or Actuator\_L = TRUE, or after a travel time amounting to 110% of the motor transition time.



### 5.3.1.4 Advanced settings V1

#### Actual value monitoring V1

Configure a warning high and low limit for the process value in the "Process value monitoring" configuration window. If one of the warning limits is exceeded or undershot during operation, a warning will be displayed at the PID\_3Step instruction:

- At the InputWarning\_H output parameter if the warning high limit has been exceeded
- At the InputWarning\_L output parameter if the warning low limit has been undershot

The warning limits must be within the process value high and low limits.

The process value high and low limits will be used if you do not enter values.

#### Example

Process value high limit = 98° C; warning high limit = 90° C

Warning low limit = 10° C; process value low limit = 0° C

PID\_3Step will respond as follows:

Process value	InputWarning_H	InputWarning_L	Operating mode
> 98° C	TRUE	FALSE	Inactive
≤ 98° C and > 90° C	TRUE	FALSE	Automatic mode
≤ 90° C and ≥ 10° C	FALSE	FALSE	Automatic mode
< 10° C and ≥ 0° C	FALSE	TRUE	Automatic mode
< 0° C	FALSE	TRUE	Inactive

### PID parameters V1

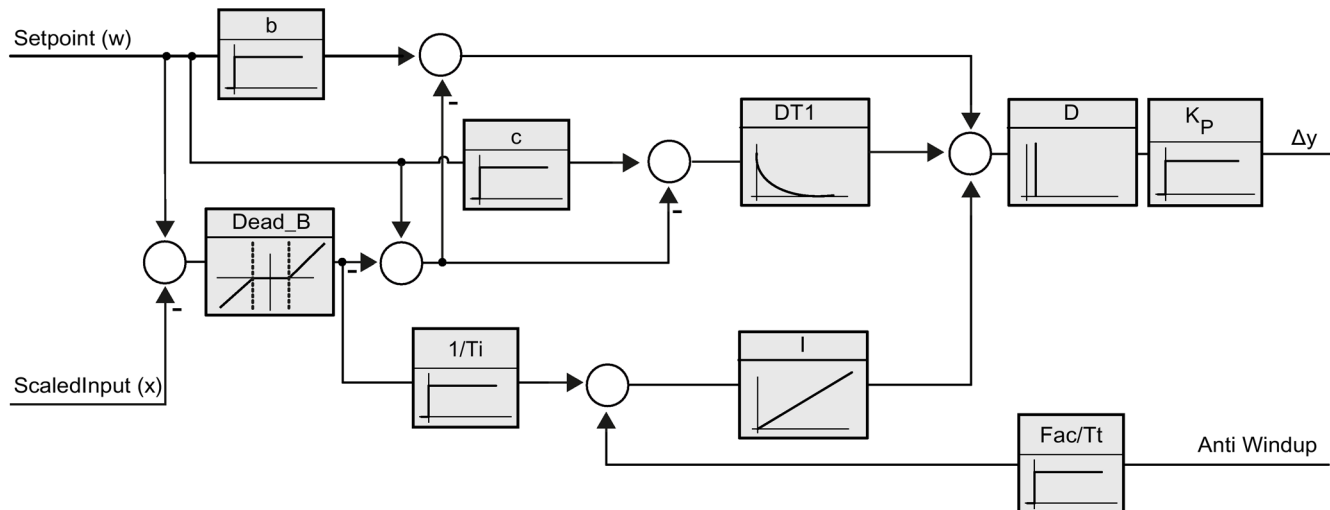
The PID parameters are displayed in the "PID Parameters" configuration window. The PID parameters will be adapted to your controlled system during controller tuning. You do not need to enter the PID parameters manually.

The PID algorithm operates according to the following equation:

$$\Delta y = K_p \cdot s \cdot \left[ (b \cdot w - x) + \frac{1}{T_i \cdot s} (w - x) + \frac{T_d \cdot s}{a \cdot T_d \cdot s + 1} (c \cdot w - x) \right]$$

Symbol	Description
$\Delta y$	Output value of the PID algorithm
$K_p$	Proportional gain
$s$	Laplace operator
$b$	Proportional action weighting
$w$	Setpoint
$x$	Process value
$T_i$	Integration time
$a$	Derivative delay coefficient (derivative delay $T_1 = a \times T_D$ )
$T_D$	Derivative action time
$c$	Derivative action weighting

The diagram below illustrates the integration of the parameters into the PID algorithm:



All PID parameters are retentive. If you enter the PID parameters manually, you must completely download PID\_3Step.

Downloading technology objects to device (Page 72)

### Proportional gain

The value specifies the proportional gain of the controller. PID\_3Step does not work with a negative proportional gain. Control logic is inverted under Basic settings > Controller type.

### Integration time

The integration time determines the time behavior of the integral action. The integral action is deactivated with integration time = 0.0.

### Derivative action time

The derivative action time determines the time behavior of the derivative action. Derivative action is deactivated with derivative action time = 0.0.

### Derivative delay coefficient

The derivative delay coefficient delays the effect of the derivative action.

Derivative delay = derivative action time × derivative delay coefficient

- 0.0: Derivative action is effective for one cycle only and therefore almost not effective.
- 0.5: This value has proved useful in practice for controlled systems with **one** dominant time constant.
- > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed.

### Proportional action weighting

The proportional action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Proportional action for setpoint change is fully effective
- 0.0: Proportional action for setpoint change is not effective

The proportional action is always fully effective when the process value is changed.

### Derivative action weighting

The derivative action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Derivative action is fully effective upon setpoint change
- 0.0: Derivative action is not effective upon setpoint change

The derivative action is always fully effective when the process value is changed.

### **PID algorithm sampling time**

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the PID\_3Step sampling time. All other functions of PID\_3Step are executed at every call.

### **Dead band width**

The deadband suppresses the noise component in the steady controller state. The dead band width specifies the size of the dead band. The dead band is off if the dead band width is 0.0.

If values not equal to 1.0 are configured for the proportional action weighting or the derivative action weighting, setpoint changes even within the dead zone affect the output value. Process value changes within the dead zone do not affect the output value, regardless of the weighting.

## 5.3.2 Commissioning PID\_3Step V1

### 5.3.2.1 Commissioning V1

You can monitor the setpoint, process value and output value over time in the "Tuning" working area. The following commissioning functions are supported in the curve plotter:

- Controller pretuning
- Controller fine tuning
- Monitoring the current closed-loop control in the trend view

All functions require an online connection to the CPU to have been established.

#### Basic handling

- Select the desired sampling time in the "Sampling time" drop-down list.  
All values in the tuning working area are updated in the selected update time.
- Click the "Start" icon in the measuring group if you want to use the commissioning functions.  
Value recording is started. The current values for the setpoint, process value and output value are entered in the trend view. Operation of the commissioning window is enabled.
- Click the "Stop" icon if you want to end the commissioning functions.  
The values recorded in the trend view can continue to be analyzed.
- Closing the commissioning window will terminate recording in the trend view and delete the recorded values.

### 5.3.2.2 Pretuning V1

The pretuning determines the process response to a pulse of the output value and searches for the point of inflection. The tuned PID parameters are calculated as a function of the maximum slope and dead time of the controlled system.

The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. The PID parameters are backed up before being recalculated.

The setpoint is frozen during pretuning.

#### Requirement

- The PID\_3Step instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- PID\_3Step is in "inactive" or "manual" mode.
- The setpoint and the process value lie within the configured limits (see "Process value settings" configuration).

#### Procedure

To perform pretuning, follow these steps:

1. Double-click the "PID\_3Step > Commissioning" entry in the project tree.
2. Select the entry "Pretuning" in the "Tuning mode" drop-down list in the working area "Tuning".
3. Click the "Start" icon.
  - An online connection will be established.
  - Value recording is started.
  - Pretuning is started.
  - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

---

#### Note

Click the "Stop" icon when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

---

#### Result

If pretuning was performed without an error message, the PID parameters have been tuned. PID\_3Step switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If pretuning is not possible, PID\_3Step changes to "Inactive" mode.

### 5.3.2.3 Fine tuning V1

Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are optimized for the operating point from the amplitude and frequency of this oscillation. All PID parameters are recalculated on the basis of the findings. PID parameters from fine tuning usually have better master control and disturbance behavior than PID parameters from pretuning.

PID\_3Step automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The PID parameters are backed up before being recalculated.

The setpoint is frozen during fine tuning.

#### Requirement

- The PID\_3Step instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- The motor transition time has been configured or measured.
- The setpoint and the process value lie within the configured limits (see "Process value settings" configuration).
- The control loop has stabilized at the operating point. The operating point is reached when the process value corresponds to the setpoint.
- No disturbances are expected.
- PID\_3Step is in inactive mode, automatic mode or manual mode.

#### Process depends on initial situation

Fine tuning proceeds as follows when started in:

- Automatic mode

Start fine tuning in automatic mode if you wish to improve the existing PID parameters using controller tuning.

PID\_3Step will regulate using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.

- Inactive or manual mode

Pretuning is always started first. The PID parameters established will be used for adjustment until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.

## Procedure

Proceed as follows to carry out "fine tuning":

1. Select the entry "Fine tuning" in the "Tuning mode" drop-down list.
2. Click the "Start" icon.
  - An online connection will be established.
  - Value recording is started.
  - The process of fine tuning is started.
  - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

---

### Note

Click the "Stop" icon in the "Tuning mode" group when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

---

## Result

The PID parameters will have been optimized if fine tuning has been executed without errors. PID\_3Step changes to automatic mode and uses the optimized parameters. The optimized PID parameters will be retained during power OFF and a restart of the CPU.

If errors occurred during fine tuning, PID\_3Step will change to "inactive" mode.

### 5.3.2.4 Commissioning with manual PID parameters V1

## Procedure

Proceed as follows to commission PID\_3Step with manual PID parameters:

1. Double-click on "PID\_3Step > Configuration" in the project tree.
2. Click on "Advanced settings > PID Parameters" in the configuration window.
3. Select the check box "Enable direct input".
4. Enter the PID parameters.
5. Double-click on "PID\_3Step > Commissioning" in the project tree.
6. Establish an online connection to the CPU.
7. Load the PID parameters to the CPU.
8. Click on the "Activate controller" icon.

## Result

PID\_3Step changes to automatic mode and controls using the current PID parameters.



### 5.3.2.5 Measuring the motor transition time V1

#### Introduction

PID\_3Step requires the motor transition time to be as accurate as possible for good controller results. The data in the actuator documentation contains average values for this type of actuator. The value for the specific actuator used may differ.


You can measure the motor transition time during commissioning if you are using actuators with position feedback or endstop signals. The output value limits are not taken into consideration during the motor transition time measurement. The actuator can travel to the high or the low endstop.

The motor transition time cannot be measured if neither position feedback nor endstop signals are available.

#### Actuators with analog position feedback

Proceed as follows to measure motor transition time with position feedback:

##### Requirement

- Feedback or Feedback\_PER has been selected in the basic settings and the signal has been connected.
  - An online connection to the CPU has been established.
1. Select the "Use position feedback" check box.
  2. Enter the position to which the actuator is to be moved in the "Target position" input field.  
The current position feedback (starting position) will be displayed. The difference between "Target position" and "Position feedback" must be at least 50% of the valid output value range.
  3. Click the  "Start transition time measurement" icon.

#### Result


The actuator is moved from the starting position to the target position. Time measurement starts immediately and ends when the actuator reaches the target position. The motor transition time is calculated according to the following equation:

Motor transition time = (output value high limit – output value low limit) × Measuring time / AMOUNT (target position – starting position).

The progress and status of transition time measurement are displayed. The transition time measured is saved in the instance data block on the CPU and displayed in the "Measured transition time" field. PID\_3Step will change to "Inactive" mode once transition time measurement is complete.

---

##### Note

Click on the icon  "Upload measured transition time" to load the motor transition time measured to the project.

---


## Actuators with endstop signals

Proceed as follows to measure the transition time of actuators with endstop signals:

### Requirement

- The "Endstop signals" check box in the basic settings has been selected and Actuator\_H and Actuator\_L are connected.
- An online connection to the CPU has been established.

Proceed as follows to measure motor transition time with endstop signals:

1. Select the "Use actuator endstop signals" check box.
2. Select the direction in which the actuator is to be moved.
  - Open - Close - Open  
The actuator is moved first to the high endstop, then to the low endstop and then back to the high endstop.
  - Close - Open - Close  
The actuator is moved first to the low endstop, then to the high endstop and then back to the low endstop.
3. Click the  "Start transition time measurement" icon.

## Result

The actuator is moved in the selected direction. Time measurement will start once the actuator has reached the first endstop and will end when the actuator reaches this endstop for the second time. The motor transition time is equal to the time measured divided by two.

The progress and status of transition time measurement are displayed. The transition time measured is saved in the instance data block on the CPU and displayed in the "Measured transition time" field. PID\_3Step will change to "Inactive" mode once transition time measurement is complete.

## Cancelling transition time measurement

PID\_3Step will change to "Inactive" mode immediately if you cancel transition time measurement. The actuator will stop being moved. You can reactive PID-3Step in the curve plotter.

### 5.3.3 Simulating PID\_3Step V1 with PLCSIM

---

#### Note

#### Simulation with PLCSIM

For the simulation with PLCSIM, the time behavior of the simulated PLC is not exactly identical to that of a "real" PLC. The actual cycle clock of a cyclic interrupt OB can have larger fluctuations with a simulated PLC than with "real" PLCs.

In the standard configuration, PID\_3Step determines the time between calls automatically and monitors them for fluctuations.

For a simulation of PID\_3Step with PLCSIM, for example, a sampling time error (ErrorBits = DW#16#00000800) can therefore be detected.

This results in ongoing tuning being aborted.

The response in automatic mode depends on the value of the ActivateRecoverMode tag.

To prevent this from happening, you should configure PID\_3Step for simulation with PLCSIM as follows:

- CycleTime.EnEstimation = FALSE
  - CycleTime.EnMonitoring = FALSE
  - CycleTime.Value: Assign the cycle clock of the calling cyclic interrupt OB in seconds to this tag.
-

## Using PID\_Temp

### 6.1 Technology object PID\_Temp

The PID\_Temp technology object provides a continuous PID controller with integrated tuning. PID\_Temp is especially designed for temperature control and is suited for heating or heating/cooling applications. Two outputs are available for this purpose, one each for heating and cooling. PID\_Temp can also be used for other control tasks. PID\_Temp is cascadable and can be used in manual or automatic mode.

PID\_Temp continuously acquires the measured process value within a control loop and compares it with the set setpoint. From the resulting control deviations, the instruction PID\_Temp calculates the output value for heating and/or cooling which is used to adjust the process value to the setpoint. The output values for the PID controller consist of three actions:

- Proportional action

The proportional action of the output value increases in proportion to the control deviation.

- Integral action

The integral action of the output value increases until the control deviation has been balanced.

- Derivative action

The derivative action increases with the rate of change of control deviation. The process value is corrected to the setpoint as quickly as possible. The derivative action will be reduced again if the rate of change of control deviation drops.

The instruction PID\_Temp calculates the proportional, integral and derivative parameters for your controlled system during "pretuning". "Fine tuning" can be used to tune the parameters further. You do not need to manually determine the parameters.

Either a fixed cooling factor or two PID parameter sets can be used for heating-and-cooling applications.

#### Additional information

- Overview of software controller (Page 38)
- Add technology objects (Page 40)
- Configure technology objects (Page 45)
- Configuring PID\_Temp (Page 165)

## 6.2 Configuring PID\_Temp

### 6.2.1 Basic settings

#### 6.2.1.1 Introduction

Configure the following properties of the "PID\_Temp" technology object under "Basic settings" in the Inspector window or in the configuration window:

- Physical quantity
- Start-up behavior after reset
- Source and input of the setpoint (only in the Inspector window)
- Selection of the process value
- Source and input of the process value (only in the Inspector window)
- Selection of the heating output value
- Source and input of the heating output value (only in the Inspector window)
- Activation and selection of the cooling output value
- Source and input of the cooling output value (only in the Inspector window)
- Activation of PID\_Temp as master or slave of a cascade
- Number of slaves
- Selection of the master (only in the Inspector window)

#### **Setpoint, process value, heating output value and cooling output value**

You can select the source and enter values or tags for the setpoint, process value, heating output value and cooling output value in the Inspector window of the programming editor.

Select the source for each value:

- Instance DB:  
The value saved in the instance DB is used. The value must be updated by the user program in the instance DB. There should be no value at the instruction. Can be changed using HMI.
- Instruction:  
The value connected to the instruction is used. The value is written to the instance DB each time the instruction is called. Cannot be changed using HMI.

### 6.2.1.2 Controller type

#### Physical quantity

Select the unit of measurement and physical quantity for the setpoint and the process value in the "Controller type" group. The setpoint and the process value are displayed in this unit.

#### Startup characteristics

1. To switch to "Inactive" mode after CPU restart, clear the "Activate Mode after CPU restart" check box.

To switch to the operating mode saved in the Mode parameter after CPU restart, select the "Activate Mode after CPU restart" check box.

2. In the "Set Mode to" drop-down list, select the mode that is to be enabled after a complete download to the device.

After a complete "Download to device", PID\_Temp starts in the selected operating mode. With each additional restart, PID\_Temp starts in the mode that was last saved in Mode.

When selecting pretuning or fine tuning, you also have to set or reset the Heat.EnableTuning and Cool.EnableTuning tags in order to choose between tuning for heating and tuning for cooling.

Example:

You have selected the "Activate Mode after CPU restart" check box and the "Pretuning" entry in the "Set Mode to" list. After a complete "Download to device", PID\_Temp starts in the "Pretuning" mode. If pretuning is still active, PID\_Temp starts in "Pretuning" mode again after restart of the CPU (heating/cooling depends on the tags Heat.EnableTuning and Cool.EnableCooling). If pretuning was successfully completed and automatic mode is active, PID\_Temp starts in "Automatic mode" after restart of the CPU.

### 6.2.1.3 Setpoint

#### Procedure

Proceed as follows to define a fixed setpoint:

1. Select "Instance DB".
2. Enter a setpoint, e.g. 80° C.
3. Delete any entry in the instruction.

Proceed as follows to define a variable setpoint:

1. Select "Instruction".
2. Enter the name of the REAL tag in which the setpoint is saved.

Program-controlled assignment of various values to the REAL tag is possible, for example for the time-controlled change of the setpoint.

#### 6.2.1.4 Process value

PID\_Temp will scale the value of the analog input to the physical quantity if you use the analog input value directly.

You will need to write a program for processing if you wish first to process the analog input value. The process value is, for example, not directly proportional to the value at the analog input. The processed process value must be in floating point format.

#### Procedure

Proceed as follows to use the analog input value without processing:

1. Select the entry "Input\_PER" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the address of the analog input.

Proceed as follows to use the processed process value in floating point format:

1. Select the entry "Input" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the name of the variable in which the processed process value is saved.

#### 6.2.1.5 Heating and cooling output value

The PID\_Temp instruction provides a PID controller with integrated tuning for temperature processes. PID\_Temp is suitable for heating or heating-and-cooling applications.

PID\_Temp provides the following output values. Your actuator will determine which output value you use.

- OutputHeat  
Heating output value (floating-point format): The output value for heating needs to be processed by the user program, for example, because of non-linear actuator response.
- OutputHeat\_PER  
Analog heating output value: The actuator for heating is triggered via an analog output and controlled with a continuous signal, e.g. 0...10 V, 4...20 mA.
- OutputHeat\_PWM  
Pulse-width modulated heating output value: The actuator for heating is controlled via a digital output. Pulse width modulation creates variable ON and OFF times.
- OutputCool  
Cooling output value (floating-point format): The output value for cooling needs to be processed by the user program, for example because of non-linear actuator response.
- OutputCool\_PER  
Analog cooling output value: The actuator for cooling is triggered via an analog output and controlled with a continuous signal, e.g. 0...10 V, 4...20 mA.

- OutputCool\_PWM

Pulse-width modulated cooling output value: The actuator for cooling is controlled via a digital output. Pulse width modulation creates variable ON and OFF times.

The cooling output is only available if it was activated via the "Activate cooling" check box.

- If the check box is cleared, the output value of the PID algorithm (PidOutputSum) is scaled and output at the outputs for heating.
- If the check box is selected, positive output values of the PID algorithm (PidOutputSum) are scaled and output at the outputs for heating. Negative output values of the PID algorithm are scaled and output at the outputs for cooling. You can choose between two methods for output value calculation at the output settings.

---

**Note**

**Note:**

- The OutputHeat\_PWM, OutputHeat\_PER, OutputCool\_PWM, OutputCool\_PER outputs are only calculated if you select these correspondingly from the drop-down list.
  - The OutputHeat output is always calculated.
  - The OutputCool output is calculated if the check box for cooling is selected.
  - The "Activate cooling" check box is only available if the controller is not configured as a master in a cascade.
- 

## Procedure

Proceed as follows to use the analog output value:

1. Select the entry "OutputHeat\_PER" or "OutputCool\_PER" in the drop-down list "OutputHeat" or "OutputCool".
2. Select "Instruction".
3. Enter the address of the analog output.

Proceed as follows to use the pulse-width modulated output value:

1. Select the entry "OutputHeat\_PWM" or "OutputCool\_PWM" in the drop-down list "OutputHeat" or "OutputCool".
2. Select "Instruction".
3. Enter the address of the digital output.

Proceed as follows to process the output value using the user program:

1. Select the entry "OutputHeat" or "OutputCool" in the drop-down list "OutputHeat" or "OutputCool".
2. Select "Instruction".
3. Enter the name of the variable you are using to process the output value.
4. Transfer the processed output value to the actuator by means of an analog or digital CPU output.



### **6.2.1.6 Cascade**

If a PID\_Temp instance receives its setpoint from a higher-level master controller and outputs its output value in turn to a subordinate slave controller, this PID\_Temp instance is both a master controller and a slave controller simultaneously. Both configurations listed below then have to be carried out for such a PID\_Temp instance. This is the case, for example, for the middle PID\_Temp instance in a cascade control system with three concatenated measured variables and three PID\_Temp instances.

#### **Configuring a controller as master in a cascade**

A master controller defines the setpoint of a slave controller with its output.

In order to use PID\_Temp as master in a cascade, you have to deactivate the cooling in the basic settings. In order to configure this PID\_Temp instance as a master controller in a cascade, activate the "Controller is master" check box. The selection of the output value for heating is set automatically to OutputHeat.

OutputHeat\_PWM and OutputHeat\_PER cannot be used at a master in a cascade.

Subsequently specify the number of directly subordinate slave controllers that receive their setpoint from this master controller.

If no own scaling function is used when assigning the OutputHeat parameter of the master to the Setpoint parameter of the slave, it may be necessary to adapt the output value limits and the output scaling of the master to the setpoint/process value range of the slave. This can be done in the output settings of the master in the "OutputHeat / OutputCool" section.

#### **Configuring a controller as a slave in a cascade**

A slave controller receives its setpoint (Setpoint parameter) from the output of its master controller (OutputHeat parameter).

In order to configure this PID\_Temp instance as a slave controller in a cascade, activate the "Controller is slave" check box in the basic settings.

Subsequently select the PID\_Temp instance that is to be used as the master controller for this slave controller in the Inspector window of the programming editor. The Master and Setpoint parameters of the slave controller are interconnected with the selected master controller through this selection (the existing interconnections at these parameters are overwritten). This interconnection allows the exchange of information and the setpoint specification between master and slave. If required, the interconnection can be changed subsequently at the Setpoint parameter of the slave controller in order, for example, to insert an additional filter. The interconnection at the parameter Master may not be changed subsequently.

The "Controller is master" check box has to be selected and the number of slaves has to be configured correctly at the selected master controller. The master controller has to be called before the slave controller in the same cyclic interrupt OB.

#### **Additional information**

Additional information about program creation, configuration and commissioning when PID\_Temp is used in cascade control systems is available under Cascade control with PID\_Temp (Page 198).

## 6.2.2 Process value settings

### 6.2.2.1 Process value limits

You must specify an appropriate absolute high limit and low limit for the process value as limit values for your controlled system. As soon as the process value violates these limits, an error occurs (ErrorBits = 0001h). Tuning is canceled when the process value limits are violated. You can specify how PID\_Temp responds to errors in automatic mode in the output settings.

### 6.2.2.2 Process value scaling

If you have configured the use of Input\_PER in the basic settings, you will need to convert the value of the analog input into the physical quantity of the process value. The current configuration is displayed in the Input\_PER display.

Input\_PER is scaled using a low and high value pair if the process value is directly proportional to the value of the analog input.

## Procedure

To scale the process value, follow these steps:

1. Enter the low pair of values in the "Scaled low process value" and "Low" input fields.
2. Enter the high pair of values in the "Scaled high process value" and "High" input fields.

Default settings for the value pairs are saved in the hardware configuration. Proceed as follows to use the value pairs from the hardware configuration:

1. Select the instruction PID\_Temp in the programming editor.
2. Interconnect Input\_PER with an analog input in the basic settings.
3. Click on the "Automatic setting" button in the process value settings.

The existing values are overwritten with the values from the hardware configuration.

## 6.2.3 Output settings

### 6.2.3.1 Basic settings of output

#### Method for heating and cooling

If cooling is activated in the basic settings, two methods are available for calculating the PID output value:

- PID parameter switching (Config.AdvancedCooling = TRUE):

The output value calculation for cooling takes place by means of a separate PID parameter set. Based on the calculated output value and the control deviation, the PID algorithm decides whether the PID parameter for heating or cooling is used. This method is suitable if the heating and cooling actuators have different time responses and different gains.

Pretuning and fine tuning for cooling are only available if this method is selected.

- Cooling factor (Config.AdvancedCooling = FALSE):

Output value calculation for cooling is effected with the PID parameters for heating under consideration of the configurable cooling factor Config.CoolFactor. This method is suitable if the heating and cooling actuators have a similar time response but different gains. If this method is selected, pretuning and fine tuning for cooling as well as the PID parameter set for cooling are not available. You can only execute the tuning for heating.

#### Cooling factor

If the cooling factor is selected as the method for heating/cooling, this factor is used in the calculation of the output value for cooling. This allows different gains of heating and cooling actuators to be taken into account.

The cooling factor is not set automatically or adjusted during tuning. You have to configure the correct cooling factor manually by using the ratio "Heating actuator gain/Cooling actuator gain".

Example: Cooling factor = 2.0 means that the heating actuator gain is twice as high as the cooling actuator gain.

The cooling factor is only effective and can only be changed if "Cooling factor" is selected as the method for heating/cooling.

### Reaction to error

<b>NOTICE</b>
<b>Your system may be damaged.</b>
If you output "Current value while error is pending " or "Substitute output value while error is pending" in the event of an error, PID_Temp remains in automatic mode or in manual mode. This may cause a violation of the process value limits and damage your system.
It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

PID\_Temp is preset so that the controller stays active in most cases in the event of an error.

If errors occur frequently in controller mode, this default reaction has a negative effect on the control response. In this case, check the ErrorBits parameter and eliminate the cause of the error.

PID\_Temp generates a programmable output value in response to an error:

- Zero (inactive)  
At all errors, PID\_Temp switches to the "Inactive" operating mode and outputs the following:
  - 0.0 as PID output value (PidOutputSum)
  - 0.0 as output value for heating (OutputHeat) and output value for cooling (OutputCool)
  - 0 as analog output value for heating (OutputHeat\_PER) and analog output value for cooling (OutputCool\_PER)
  - FALSE as PWM output value for heating (OutputHeat\_PWM) and PWM output value for cooling (OutputCool\_PWM)This is independent of the configured output value limits and the scaling. The controller is only reactivated by a falling edge at Reset or a rising edge at ModeActivate.
- Current value while error is pending  
The error response depends on the error occurring and the operating mode.  
If one or more of the following errors occur in automatic mode, PID\_Temp stays in automatic mode:
  - 0000001h: The Input parameter is outside the process value limits.
  - 0000800h: Sampling time error
  - 0040000h: Invalid value at Disturbance parameter.
  - 8000000h: Error during the calculation of the PID parameters.

If one or more of the following errors occur in automatic mode, PID\_Temp switches to "Substitute output value with error monitoring" mode and outputs the last valid PID output value (PidOutputSum):

- 0000002h: Invalid value at Input\_PER parameter.
- 0000200h: Invalid value at Input parameter.
- 0000400h: Calculation of output value failed.
- 0001000h: Invalid value at Setpoint or SubstituteSetpoint parameter.

The values at the outputs for heating and cooling resulting from the PID output value are produced by the configured output scaling.

As soon as the errors are no longer pending, PID\_Temp switches back to automatic mode.

If an error occurs during manual mode, PID\_Temp remains in manual mode and continues to use the manual value as the PID output value.

If the manual value is invalid, the configured substitute output value is used.

If the manual value and substitute output value are invalid, the low limit of the PID output value for heating (Config.Output.Heat.PidLowerLimit) is used.

If the following error occurs during pretuning or fine tuning, PID\_Temp remains in active mode:

- 0000020h: Pretuning is not permitted during fine tuning.

When any other error occurs, PID\_Temp cancels the tuning and switches to the mode from which tuning was started.

- Substitute output value while error is pending

PID\_Temp behaves as described at "Current value while error is pending", but outputs the configured substitute output value (SubstituteOutput) as a PID output value (PidOutputSum) in "Substitute output value with error monitoring" operating mode.

The values at the outputs for heating and cooling resulting from the PID output value are produced by the configured output scaling.

In the case of controllers with activated cooling output (Config.ActivateCooling = TRUE), enter:

- A positive substitute output value to output the value at the outputs for heating.
- A negative substitute output value to output the value at the outputs for cooling.

If the following error occurs, PID\_Temp stays in "Substitute output value with error monitoring" mode and outputs the low limit of the PID output value for heating (Config.Output.Heat.PidLowerLimit):

- 0020000h: Invalid value at SubstituteOutput tag.

### 6.2.3.2 Output value limits and scaling

Depending on the operating mode, the PID output value (PidOutputSum) is calculated automatically by the PID algorithm or by the manual value (ManualValue) or the configured substitute output value (SubstituteOutput).

The PID output value is limited depending on the configuration:

- If the cooling is deactivated in the basic settings (Config.ActivateCooling = FALSE), the value is limited to the high limit of the PID output value (heating) (Config.Output.Heat.PidUpperLimit) and the low limit of the PID output value (heating) (Config.Output.Heat.PidLowerLimit).

You can configure both limits at the horizontal axis of the scaling characteristic line in the "OutputHeat / OutputCool" section. These are displayed in the "OutputHeat\_PWM / OutputCool\_PWM" and "OutputHeat\_PER / OutputCool\_PER" sections, but cannot be changed.

- If the cooling is activated in the basic settings (Config.ActivateCooling = TRUE), the value is limited to the high limit of the PID output value (Config.Output.Heat.PidUpperLimit) and the low limit of the PID output value (cooling) (Config.Output.Cool.PidLowerLimit).

You can configure both limits at the horizontal axis of the scaling characteristic line in the "OutputHeat / OutputCool" section. These are displayed in the "OutputHeat\_PWM / OutputCool\_PWM" and "OutputHeat\_PER / OutputCool\_PER" sections, but cannot be changed.

The low limit of the PID output value (heating) (Config.Output.Heat.PidLowerLimit) and the high limit of the PID output value (cooling) (Config.Output.Cool.PidUpperLimit) cannot be changed and have to be assigned the value 0.0.

The PID output value is scaled and output at the outputs for heating and cooling. Scaling can be specified separately for each output and is specified across 2 value pairs each, consisting of a limit value of the PID output value and a scaling value:

Output	Value pair	Parameter
OutputHeat	Value pair 1	PID output value high limit (heating) Config.Output.Heat.PidUpperLimit, Scaled high output value (heating) Con- fig.Output.Heat.UpperScaling
	Value pair 2	PID output value low limit (heating) Config.Output.Heat.PidLowerLimit, Scaled low output value (heating) Con- fig.Output.Heat.LowerScaling
OutputHeat_PWM	Value pair 1	PID output value high limit (heating) Config.Output.Heat.PidUpperLimit, Scaled high PWM output value (heating) Config.Output.Heat.PwmUpperScaling
	Value pair 2	PID output value low limit (heating) Config.Output.Heat.PidLowerLimit, Scaled low PWM output value (heating) Config.Output.Heat.PwmLowerScaling

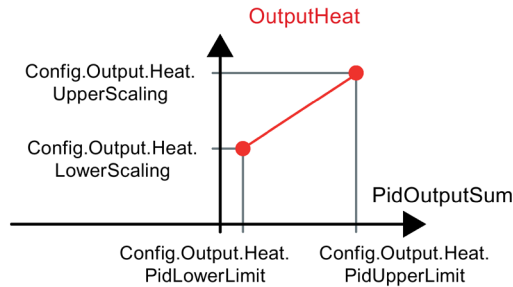
Output	Value pair	Parameter
OutputHeat_PER	Value pair 1	PID output value high limit (heating) Config.Output.Heat.PidUpperLimit, Scaled high analog output value (heating) Config.Output.Heat.PerUpperScaling
	Value pair 2	PID output value low limit (heating) Config.Output.Heat.PidLowerLimit, Scaled low analog output value (heating) Config.Output.Heat.PerLowerScaling
OutputCool	Value pair 1	PID output value low limit (cooling) Config.Output.Cool.PidLowerLimit, Scaled high output value (cooling) Config.Output.Cool.UpperScaling
	Value pair 2	PID output value high limit (cooling) Config.Output.Cool.PidUpperLimit, Scaled low output value (cooling) Config.Output.Cool.LowerScaling
OutputCool_PWM	Value pair 1	PID output value low limit (cooling) Config.Output.Cool.PidLowerLimit, Scaled high PWM output value (cooling) Config.Output.Cool.PwmUpperScaling
	Value pair 2	PID output value high limit (cooling) Config.Output.Cool.PidUpperLimit, Scaled low PWM output value (cooling) Config.Output.Cool.PwmLowerScaling
OutputCool_PER	Value pair 1	PID output value low limit (cooling) Config.Output.Cool.PidLowerLimit, Scaled high analog output value (cooling) Config.Output.Cool.PerUpperScaling
	Value pair 2	PID output value high limit (cooling) Config.Output.Cool.PidUpperLimit, Scaled low analog output value (cooling) Config.Output.Cool.PerLowerScaling

The low limit of PID output value (heating) (Config.Output.Heat.PidLowerLimit) has to have the value 0.0, if the cooling is activated (Config.ActivateCooling = TRUE).

The high limit of PID output value (cooling) (Config.Output.Cool.PidUpperLimit) must always have the value 0.0.

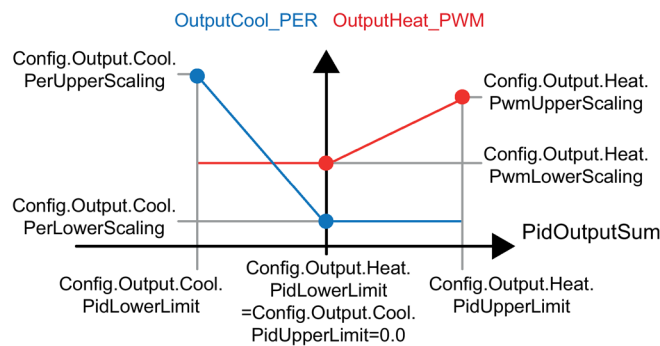
**Example:**

Output scaling when the OutputHeat output is used (cooling deactivated. The low limit of PID output value (heating) (Config.Output.Heat.PidLowerLimit) may be unequal to 0.0):



**Example:**

Output scaling when the OutputHeat\_PWM and OutputCool\_PER outputs are used (cooling activated. The low limit of PID output value (heating) (Config.Output.Heat.PidLowerLimit) must be 0.0):



With the exception of the "Inactive" operating mode, the value at an output always lies between its scaled high output value and the scaled low output value, for example for OutputHeat always between the scaled high output value (heating) (Config.Output.Heat.UpperScaling) and the scaled low output value (heating) (Config.Output.Heat.LowerScaling).

If you want to limit the value at the associated output, you therefore have to adapt these scaling values as well.

You can configure the scaling values of an output at the vertical axes of the scaling characteristic line. Each output has two separate scaling values. These can only be changed for OutputHeat\_PWM, OutputCool\_PWM, OutputHeat\_PER and OutputCool\_PER if the corresponding output is selected in the basic settings. The cooling has to be activated additionally in the basic settings at all the outputs for cooling.

The trend view in the commissioning dialog box only records the values of OutputHeat and OutputCool, irrespective of the selected output in the basic settings. Therefore, if necessary, adapt the scaling values for OutputHeat or OutputCool if you use OutputHeat\_PWM or OutputHeat\_PER or OutputCool\_PWM or OutputCool\_PER and want to use the trend view in the commissioning dialog.



## 6.2.4 Advanced settings

### 6.2.4.1 Process value monitoring

Configure a warning high and low limit for the process value in the "Process value monitoring" configuration window. If one of the warning limits is exceeded or undershot during operation, a warning is displayed at the PID\_Temp instruction:

- At the InputWarning\_H output parameter if the warning high limit has been exceeded
- At the InputWarning\_L output parameter if the warning low limit has been undershot

The warning limits must be within the process value high and low limits.

The process value high and low limits are used if you do not enter values.

### Example

Process value high limit = 98° C; warning high limit = 90° C

Warning low limit = 10° C; process value low limit = 0° C

PID\_Temp will respond as follows:

Process value	InputWarning_H	InputWarning_L	ErrorBits
> 98 °C	TRUE	FALSE	0001h
≤ 98° C and > 90° C	TRUE	FALSE	0000h
≤ 90° C and ≥ 10° C	FALSE	FALSE	0000h
< 10° C and ≥ 0° C	FALSE	TRUE	0000h
< 0° C	FALSE	TRUE	0001h

You can configure the response of PID\_Temp when the process value high limit or low limit is violated in the output settings.

### 6.2.4.2 PWM limits

The PID output value `PidOutputSum` is scaled and transformed via a pulse width modulation into a pulse train that is output at the output parameter `OutputHeat_PWM` or `OutputCool_PWM`. The "Sampling time of PID algorithm" represents the time between two calculations of the PID output value. The sampling time is used as time period of the pulse width modulation.

During heating, the PID output value is always calculated in the "Sampling time of PID algorithm for heating".

Calculation of the PID output value during cooling depends on the type of cooling selected in "Basic settings Output":

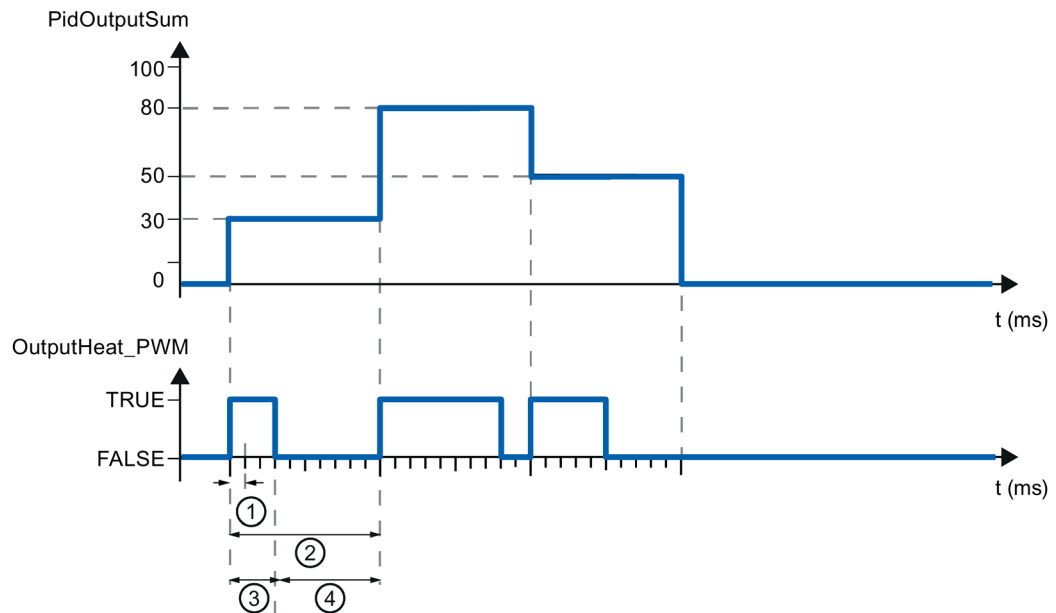
- If the cooling factor is used, the "Sampling time of PID algorithm for heating" applies.
- If the PID parameter switching is used, the "Sampling time of PID algorithm for cooling" applies.

`OutputHeat_PWM` and `OutputCool_PWM` are output in the sampling time `PID_Temp` (corresponds to the cycle time of the calling OB).

The PID algorithm sampling time for heating or cooling is determined during pretuning or fine tuning. If you set the PID parameters manually, you will also need to configure the PID algorithm sampling time for heating or cooling. The `PID_Temp` sampling time is equivalent to the cycle time of the calling OB.

The pulse duration is proportional to the PID output value and is always an integer multiple of the `PID_Temp` sampling time.

#### Example for `OutputHeat_PWM`



- ① `PID_Temp` sampling time
- ② PID algorithm sampling time for heating
- ③ Pulse duration
- ④ Break time

The "Minimum ON time" and the "Minimum OFF time" can be set separately for heating and cooling, rounded to an integer multiple of the PID\_Temp sampling time.

A pulse or a break is never shorter than the minimum ON or OFF time. The inaccuracies this causes are added up and compensated in the next cycle.

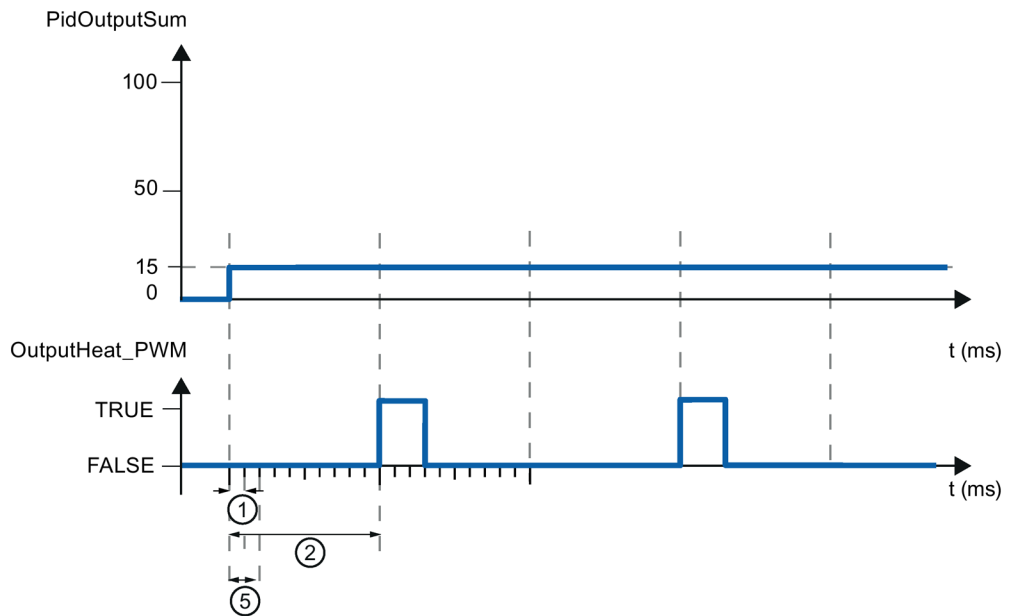
#### Example for OutputHeat\_PWM

PID\_Temp sampling time = 100 ms

PID algorithm sampling time = 1000 ms

Minimum ON time = 200 ms

The PID output value PidOutputSum amounts to 15% constantly. The smallest pulse that PID\_Temp can output corresponds to 20%. In the first cycle, no pulse is output. In the second cycle, the pulse not output in the first cycle is added to the pulse of the second cycle.



- ① PID\_Temp sampling time
- ② PID algorithm sampling time for heating
- ⑤ Minimum ON time

In order to minimize operation frequency and conserve the actuator, extend the minimum ON and OFF times.

If you have selected OutputHeat/OutputCool or OutputHeat\_PER/OutputCool\_PER as the output in the basic settings, the minimum ON time and the minimum OFF time are not evaluated and cannot be changed.

If the "Sampling time of PID algorithm" (Retain.CtrlParams.Heat.Cycle or Retain.CtrlParams.Cool.Cycle) and thus the period duration of the pulse width modulation is very high when OutputHeat\_PWM or OutputCool\_PWM is used, you can specify a deviating shorter period duration at the parameters Config.Output.Heat.PwmPeriode or Config.Output.Cool.PwmPeriode in order to improve smoothness of the process value (see also PwmPeriode tag (Page 456)).

---

**Note**

The minimum ON and OFF times only affect the output parameters OutputHeat\_PWM or OutputCool\_PWM and are not used for any pulse generators integrated in the CPU.

---

### 6.2.4.3 PID parameters

The PID parameters are displayed in the "PID Parameters" configuration window.

If cooling is activated in the basic settings and PID parameter switching is selected as the method for heating/cooling in the output settings, two parameter sets are available: One for heating and one for cooling.

In this case, the PID algorithm decides on the basis of the calculated output value and the control deviation whether the PID parameters for heating or cooling are used.

If cooling is deactivated or the cooling factor is selected as the method for heating/cooling, the parameter set for heating is always used.

During tuning, the PID parameters are adapted to the controlled system with the exception of the dead zone width that has to be configured manually.

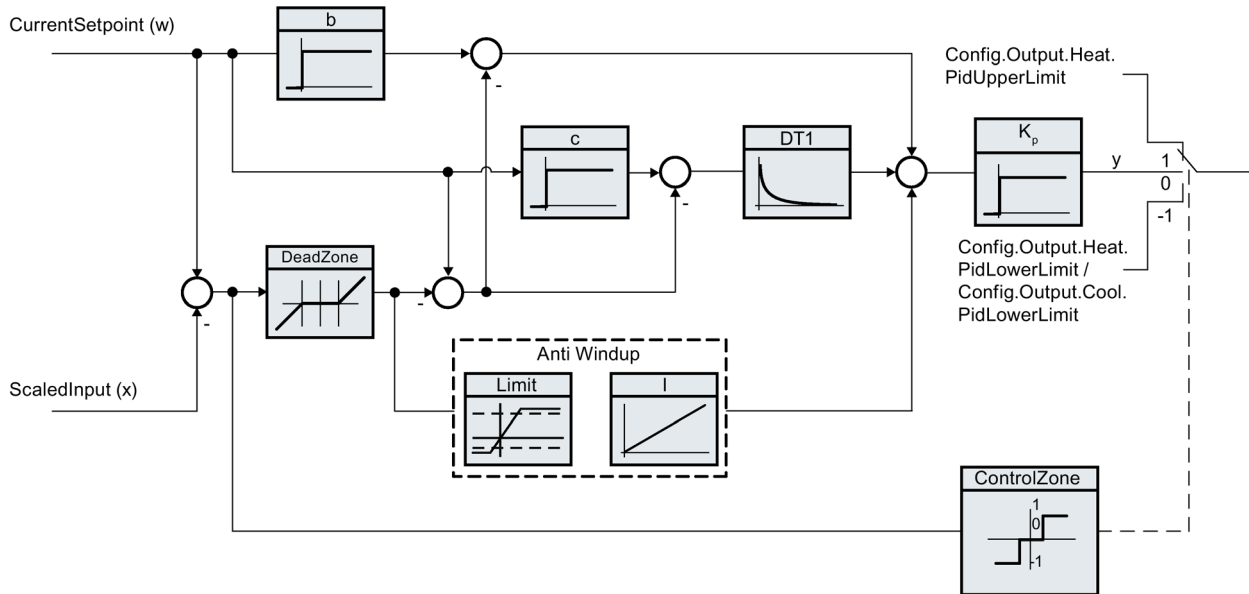
PID\_Temp is a PIDT1 controller with anti-windup and weighting of the proportional and derivative actions.

The PID algorithm operates according to the following equation (control zone and dead zone deactivated):

$$y = K_p \left[ (b \cdot w - x) + \frac{1}{T_i \cdot s} (w - x) + \frac{T_d \cdot s}{a \cdot T_d \cdot s + 1} (c \cdot w - x) \right]$$

Symbol	Description	Associated parameters of the PID_Temp instruction
y	Output value of the PID algorithm	-
K <sub>p</sub>	Proportional gain	Retain.CtrlParams.Heat.Gain Retain.CtrlParams.Cool.Gain CoolFactor
s	Laplace operator	-
b	Proportional action weighting	Retain.CtrlParams.Heat.PWeighting Retain.CtrlParams.Cool.PWeighting
w	Setpoint	CurrentSetpoint
x	Process value	ScaledInput
T <sub>i</sub>	Integral action time	Retain.CtrlParams.Heat.Ti Retain.CtrlParams.Cool.Ti
T <sub>D</sub>	Derivative action time	Retain.CtrlParams.Heat.Td Retain.CtrlParams.Cool.Td
a	Coefficient for derivative-action delay (Derivative delay T1 = a × T <sub>D</sub> )	Retain.CtrlParams.Heat.TdFiltRatio Retain.CtrlParams.Cool.TdFiltRatio
c	Derivative action weighting	Retain.CtrlParams.Heat.DWeighting Retain.CtrlParams.Cool.DWeighting
DeadZone	Dead zone width	Retain.CtrlParams.Heat.DeadZone Retain.CtrlParams.Cool.DeadZone
ControlZone	Control zone width	Retain.CtrlParams.Heat.ControlZone Retain.CtrlParams.Cool.ControlZone

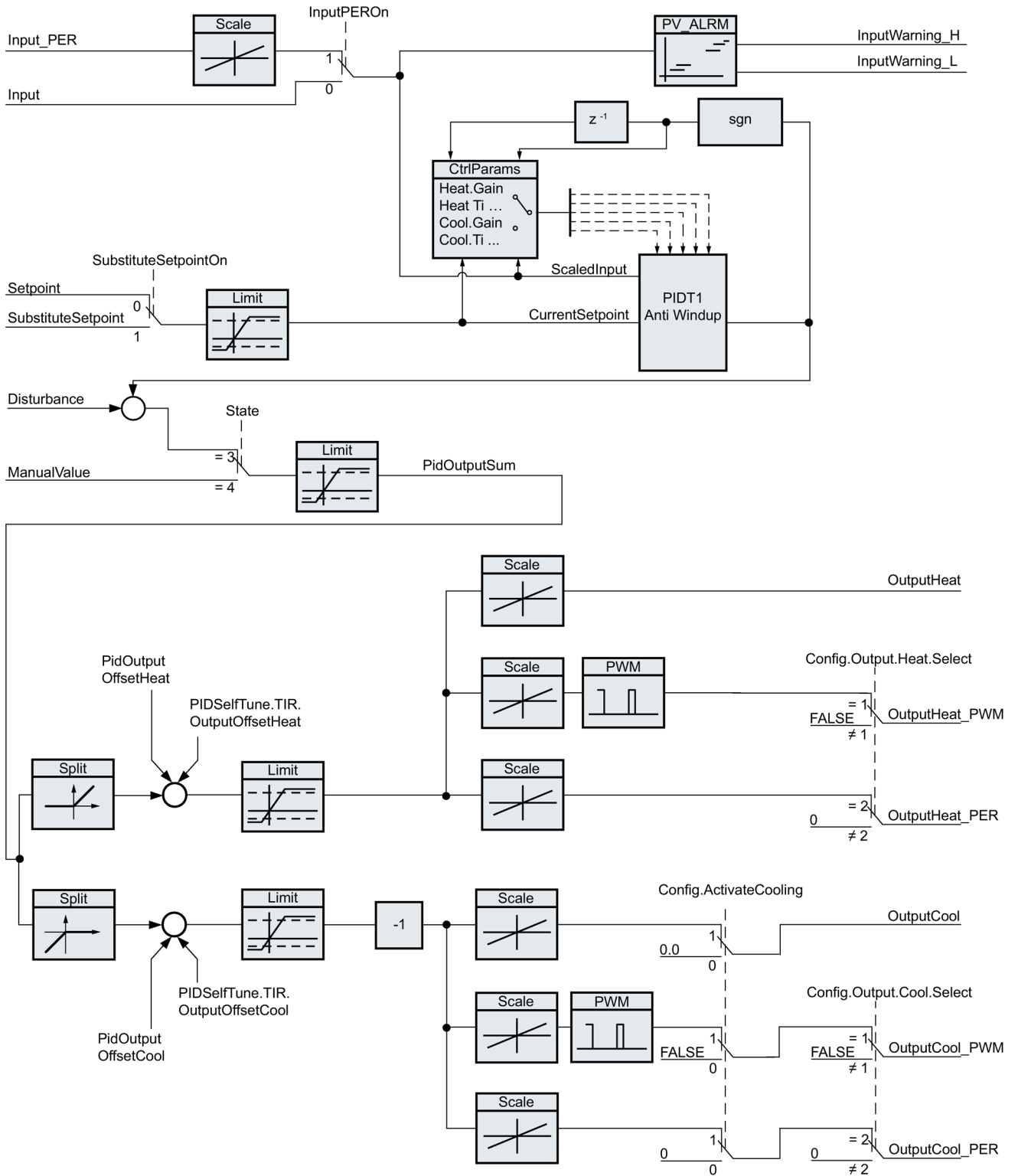
The diagram below illustrates the integration of the parameters into the PID algorithm:



All PID parameters are retentive. If you enter the PID parameters manually, you must completely download PID\_Temp (Downloading technology objects to device (Page 72)).

### PID\_Temp block diagram

The following block diagram shows how the PID algorithm is integrated in the PID\_Temp.



### Proportional gain

The value specifies the proportional gain of the controller. PID\_Temp does not operate with a negative proportional gain and only supports the normal control direction, meaning that an increase in the process value is achieved by an increase in the PID output value (PidOutputSum).

### Integral action time

The integral action time determines the time behavior of the integral action. The integral action is deactivated with integral action time = 0.0.

### Derivative action time

The derivative action time determines the time behavior of the derivative action. Derivative action is deactivated with derivative action time = 0.0.

### Derivative delay coefficient

The derivative delay coefficient delays the effect of the derivative action.

Derivative delay = derivative action time × derivative delay coefficient

- 0.0: Derivative action is effective for one cycle only and therefore almost not effective.
- 0.5: This value has proved useful in practice for controlled systems with one dominant time constant.
- > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed.

### Proportional action weighting

The proportional action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Proportional action for setpoint change is fully effective
- 0.0: Proportional action for setpoint change is not effective

The proportional action is always fully effective when the process value is changed.

### Derivative action weighting

The derivative action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Derivative action is fully effective upon setpoint change
- 0.0: Derivative action is not effective upon setpoint change

The derivative action is always fully effective when the process value is changed.



## PID algorithm sampling time

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of "PID algorithm" represents the time between two calculations of the PID output value. It is calculated during tuning and rounded to a multiple of the PID\_Temp sampling time (cycle time of the cyclic interrupt OB). All other functions of PID\_Temp are executed at every call.

If you use OutputHeat\_PWM or OutputCool\_PWM, the sampling time of the PID algorithm is used as the period duration of the pulse width modulation. The accuracy of the output signal is determined by the ratio of the PID algorithm sampling time to the cycle time of the OB. The cycle time should be no more than a tenth of the PID algorithm sampling time.

The sampling time of the PID algorithm that is used as the period duration of the pulse width modulation at OutputCool\_PWM depends on the method for heating/cooling selected in "Basic settings Output":

- If the cooling factor is used, the "sampling time of the PID algorithm for heating" also applies to OutputCool\_PWM.
- If PID parameter switching is used, the "sampling time PID algorithm for cooling" applies as the period duration for OutputCool\_PWM.

If the sampling time of the PID algorithm and thus the period duration of the pulse width modulation is very high when OutputHeat\_PWM or OutputCool\_PWM is used, you can specify a deviating shorter period duration at the parameters Config.Output.Heat.PwmPeriode or Config.Output.Cool.PwmPeriode in order to improve smoothness of the process value.

## Dead zone width

If the process value is affected by noise, the noise can also have an effect on the output value. The output value may fluctuate considerably when controller gain is high and the derivative action is activated. If the process value lies within the dead zone around the setpoint, the control deviation is suppressed so that the PID algorithm does not react and unnecessary fluctuations of the output value are reduced.

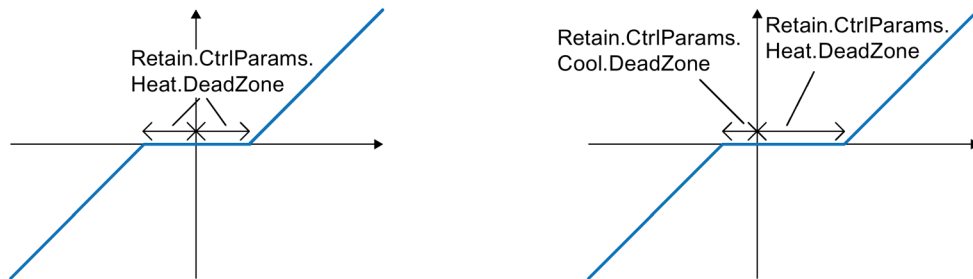
The dead zone width for heating or cooling is not set automatically during tuning. You have to correctly configure the dead zone width manually. The dead zone is deactivated by setting the dead zone width = 0.0.

When the dead zone is switched on, the result can be a permanent control deviation (deviation between setpoint and actual value). This can have a negative effect on fine tuning.

If cooling is activated in the basic settings and PID parameter switching is selected as the method for heating/cooling in the output settings, the dead zone lies between "Setpoint - dead zone width (heating)" and "Setpoint + dead zone width (cooling)".

If cooling is deactivated in the basic settings or the cooling factor is used, the dead zone lies symmetrically between "Setpoint - dead zone width (heating)" and "Setpoint + dead zone width (heating)".

If values not equal to 1.0 are configured for the proportional action weighting or the derivative action weighting, setpoint changes even within the dead zone affect the output value. Process value changes within the dead zone do not affect the output value, regardless of the weighting.



Dead zone with deactivated cooling or cooling factor (left) or activated cooling and PID parameter switching (right). The x / horizontal axis displays the control deviation = setpoint - process value. The y / vertical axis shows the output signal of the dead zone that is passed to the PID algorithm.

**Control zone width**

If the process value exits the control zone around the setpoint, the minimum or maximum output value is output. This means that the process value reaches the setpoint faster.

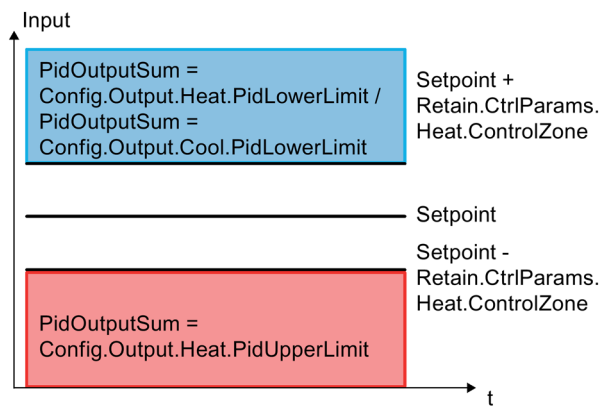
If the process value lies within the control zone around the setpoint, the output value is calculated by the PID algorithm.

The control zone width for heating or cooling is only set automatically during the pretuning, if "PID (temperature)" is selected as the controller structure for cooling or heating.

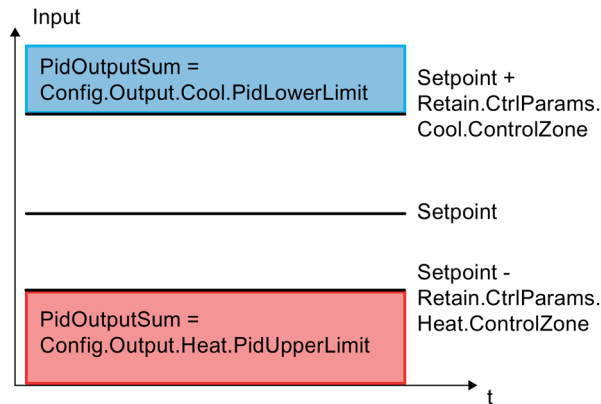
The control zone is deactivated by setting the control zone width = 3.402822e+38.

If cooling is deactivated in the basic settings or the cooling factor is used, the control zone lies symmetrically between "Setpoint - control zone width (heating)" and "Setpoint + control zone width (heating)".

If cooling is activated in the basic settings and PID parameter switching is selected as the method for heating/cooling in the output settings, the control zone lies between "Setpoint - control zone width (heating)" and "Setpoint + control zone width (cooling)".



Control zone with deactivated cooling or cooling factor.



Control zone with activated cooling and PID parameter switching.

## Rule for tuning

Select whether PI or PID parameters are to be calculated in the "Controller structure" drop-down list. You can specify the rules for tuning for heating and for tuning for cooling separately.

- PID (temperature)

Calculates PID parameters during pretuning and fine tuning.

Pretuning is designed for temperature processes and results in a slower and rather asymptotic control response with lower overshoot than with the "PID" option. Fine tuning is identical to the "PID" option.

The control zone width is determined automatically during pretuning only if this option is selected.

- PID

Calculates PID parameters during pretuning and fine tuning.

- PI

Calculates PI parameters during pretuning and fine tuning.

- User-defined

The drop-down list displays "User-defined" if you have configured different controller structures for pretuning and fine tuning via a user program or the parameter view.


## 6.3 Commissioning PID\_Temp

### 6.3.1 Commissioning

The commissioning window helps you commission the PID controller. You can monitor the values for the setpoint, process value and the output values for heating and cooling along the time axis in the trend view. The following functions are supported in the commissioning window:

- Controller pretuning
- Controller fine tuning  
Use fine tuning for fine adjustments to the PID parameters.
- Monitoring the current closed-loop control in the trend view
- Testing the controlled system by specifying a manual PID output value and a substitute setpoint
- Saving the actual values of the PID parameters to an offline project.

All functions require an online connection to the CPU.

The online connection to the CPU is established, if it does not exist already, and operation of the commissioning window is enabled by means of the "Monitor all"  or "Start" buttons of the trend view.

#### Operation of the trend view

- Select the desired sampling time in the "Sampling time" drop-down list.  
All the values of the trend view are updated in the selected sampling time.
- Click the "Start" icon in the Measurement group if you want to use the trend view.  
Value recording is started. The current values for the setpoint, process value and output values for heating and cooling are entered in the trend view.
- Click the "Stop" icon if you want to end the trend view.  
The values recorded in the trend view can continue to be analyzed.

Closing the commissioning window will terminate recording in the trend view and delete the recorded values.

## 6.3.2 Pretuning

The pretuning determines the process response to a jump change of the output value and searches for the point of inflection. The tuned PID parameters are calculated as a function of the maximum slope and dead time of the controlled system. You obtain the best PID parameters when you perform pretuning and fine tuning.

The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. This is most likely the case in operating modes "Inactive" or "Manual mode". The PID parameters are backed up before being recalculated.

PID\_Temp offers different pretuning types depending on the configuration:

- Pretuning heating

A jump is output at the output value heating, the PID parameters for heating are calculated and then the setpoint is used as the control variable in automatic mode.

- Pretuning heating and cooling

A jump is output at the output value heating.

As soon as the process value is close to the setpoint, a jump change is output at the output value cooling.

The PID parameters for heating (Retain.CtrlParams.Heat structure) and cooling (Retain.CtrlParams.Cool structure) are calculated and then the setpoint is used as the control variable in automatic mode.

- Pretuning cooling

A jump is output at the output value cooling.

The PID parameters for cooling are calculated and then the setpoint is used as the control variable in automatic mode.

If you want to tune the PID parameters for heating and cooling, you can expect a better control response with "Pretuning heating" followed by "Pretuning cooling" rather than with "Pretuning heating and cooling". However, carrying out pretuning in two steps takes more time.

### General requirements

- The PID\_Temp instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- Reset = FALSE
- PID\_Temp is in one of the following modes: "Inactive", "Manual mode", or "Automatic mode".
- The setpoint and the process value lie within the configured limits (see Process value monitoring (Page 177) configuration).

### Requirements for pretuning heating

- The difference between setpoint and process value is greater than 30% of the difference between process value high limit and process value low limit.
- The distance between the setpoint and the process value is greater than 50% of the setpoint.
- The setpoint is greater than the process value.

### Requirements for pretuning heating and cooling


- The cooling output in the "Basic settings" is activated (Config.ActivateCooling = TRUE).
- The PID parameter switching in the "Basic settings of output value" is activated (Config.AdvancedCooling = TRUE).
- The difference between setpoint and process value is greater than 30% of the difference between process value high limit and process value low limit.
- The distance between the setpoint and the process value is greater than 50% of the setpoint.
- The setpoint is greater than the process value.

### Requirements for pretuning cooling

- The cooling output in the "Basic settings" is activated (Config.ActivateCooling = TRUE).
- The PID parameter switching in the "Basic settings of output value" is activated (Config.AdvancedCooling = TRUE).
- "Pretuning heating" or "Pretuning heating and cooling" has been carried out successfully (PIDSelfTune.SUT.ProcParHeatOk = TRUE). The same setpoint should be used for all tunings.
- The difference between setpoint and process value is smaller than 5% of the difference between process value high limit and process value low limit.

### Procedure

To perform pretuning, follow these steps:

1. Double-click the "PID\_Temp > Commissioning" entry in the project tree.
2. Activate the "Monitor all"  button or start the trend view.

An online connection will be established.

3. Select the desired pretuning entry from the "Tuning mode" drop-down list.
4. Click the "Start" icon.
  - Pretuning is started.
  - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

---

**Note**

Click the "Stop" icon when the progress bar ("Progress" tag) has not changed for a long period and it is to be assumed that the tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

---

## Result

If pretuning was performed without an error message, the PID parameters have been tuned. PID\_Temp switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If pretuning is not possible, PID\_Temp responds with the configured reaction to errors.

### 6.3.3 Fine tuning

Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are tuned for the operating point from the amplitude and frequency of this oscillation. The PID parameters are recalculated from the results. PID parameters from fine tuning usually have better master control and disturbance characteristics than PID parameters from pretuning. You obtain the best PID parameters when you perform pretuning and fine tuning.

PID\_Temp automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The PID parameters are backed up before being recalculated.

PID\_Temp offers different fine tuning types depending on the configuration:

- Fine tuning heating:

PID\_Temp generates an oscillation of the process value with periodic changes at the output value heating and calculates the PID parameters for heating.

- Fine tuning cooling:

PID\_Temp generates an oscillation of the process value with periodic changes at the output value cooling and calculates the PID parameters for cooling.

#### Temporary tuning offset for heating/cooling controllers

If PID\_Temp is used as a heating/cooling controller (Config.ActivateCooling = TRUE), the PID output value (PidOutputSum) at the setpoint has to fulfill the following requirements so that process value oscillation can be generated and fine tuning can be carried out successfully:

- Positive PID output value for fine tuning heating
- Negative PID output value for fine tuning cooling

If this condition is not fulfilled, you can specify a temporary offset for fine tuning that is output at the opposing output.

- Offset for cooling output (PIDSelfTune.TIR.OutputOffsetCool) with fine tuning heating.

Before starting tuning, enter a negative tuning offset cooling that is smaller than the PID output value (PidOutputSum) at the setpoint in the stationary state.

- Offset for heating output (PIDSelfTune.TIR.OutputOffsetHeat) at fine tuning cooling

Before starting tuning, enter a positive tuning offset heating that is greater than the PID output value (PidOutputSum) at the setpoint in the stationary state.

The defined offset is balanced by the PID algorithm so that the process value remains at the setpoint. The height of the offset allows the PID output value to be adapted correspondingly so that it fulfills the requirement mentioned above.

To avoid larger overshoots of the process value when defining the offset, it can also be increased in several steps.

If PID\_Temp exits the fine tuning mode, the tuning offset is reset.



### Example: Specification of an offset for fine tuning cooling

- Without offset
  - Setpoint = Process value (ScaledInput) = 80 °C
  - PID output value (PidOutputSum) = 30.0
  - Output value heating (OutputHeat) = 30.0
  - Output value cooling (OutputCool) = 0.0

Oscillation of the process value around the setpoint cannot be generated with the cooling output alone. Fine tuning would fail here.

- With offset for heating output (PIDSelfTune.TIR.OutputOffsetHeat) = 80.0
  - Setpoint = Process value (ScaledInput) = 80 °C
  - PID output value (PidOutputSum) = -50.0
  - Output value heating (OutputHeat) = 80.0
  - Output value cooling (OutputCool) = -50.0

Thanks to the specification of an offset for the heating output, the cooling output can now generate oscillation of the process value around the setpoint. Fine tuning can now be carried out successfully.

### General requirements

- The PID\_Temp instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- Reset = FALSE
- The setpoint and the process value lie within the configured limits (see "Process value settings" configuration).
- The control loop has stabilized at the operating point. The operating point is reached when the process value corresponds to the setpoint.  

When the dead zone is switched on, the result can be a permanent control deviation (deviation between setpoint and actual value). This can have a negative effect on fine tuning.
- No disturbances are expected.
- PID\_Temp is in inactive mode, automatic mode or manual mode.

### Requirements for fine tuning heating

- Heat.EnableTuning = TRUE
- Cool.EnableTuning = FALSE
- If PID\_Temp is configured as a heating-and-cooling controller (Config.ActivateCooling = TRUE), the heating output has to be active at the operating point where tuning is to be carried out.  

PidOutputSum > 0.0 (see tuning offset)

### Requirements for fine tuning cooling

- Heat.EnableTuning = FALSE
- Cool.EnableTuning = TRUE
- The cooling output is activated (Config.ActivateCooling = TRUE).
- The PID parameter switching is activated (Config.AdvancedCooling = TRUE).
- The cooling output has to be active at the operating point where tuning is to be carried out.  
PidOutputSum < 0.0 (see tuning offset)

### Process depends on initial situation


Fine tuning can be started from the following operating modes: "Inactive", "automatic mode", or "manual mode".

Fine tuning proceeds as follows when started from:

- Automatic mode with PIDSelfTune.TIR.RunIn = FALSE (default)  
Start fine tuning from automatic mode if you wish to improve the existing PID parameters through tuning.  
PID\_Temp controls the system using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.
- Inactive, manual mode or automatic mode with PIDSelfTune.TIR.RunIn = TRUE  
An attempt is made to reach the setpoint with the minimum or maximum output value (two-point control):
  - With minimum or maximum output value heating at fine tuning heating.
  - With minimum or maximum output value cooling for fine tuning cooling.This can produce increased overshoot. Fine tuning starts when the setpoint is reached.  
If the setpoint cannot be reached, PID\_Temp does not automatically abort tuning.

### Procedure

To perform fine tuning, follow these steps:

1. Double-click the "PID\_Temp > Commissioning" entry in the project tree.
2. Activate the "Monitor all"  button or start the trend view.  
An online connection will be established.
3. Select the desired fine tuning entry from the "Tuning mode" drop-down list.

4. If required (see tuning offset), specify a tuning offset and wait until the stationary state is reached again.
5. Click the "Start" icon.
  - The process of fine tuning is started.
  - The "Status" field displays the current steps and any errors that may have occurred.  
The progress bar indicates the progress of the current step.

---

**Note**

Click the "Stop" icon in the "Tuning mode" group if the progress bar ("Progress" tag) has not changed for a long period and it is to be assumed that the tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

In the following phases in particular, tuning is not aborted automatically if the setpoint cannot be reached.

- "Attempting to reach setpoint for heating with two-point control."
  - "Attempting to reach setpoint for cooling with two-point control."
- 

## Result

If fine tuning was performed without errors, the PID parameters have been tuned. PID\_Temp switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If errors occurred during fine tuning, PID\_Temp responds with the configured response to errors.

### 6.3.4 "Manual" mode

The following section describes how you can use "Manual mode" in the commissioning window of the "PID\_Temp" technology object.

Manual mode is also possible when an error is pending.


#### Requirement

- The "PID\_Temp" instruction is called in a cyclic interrupt OB.
- An online connection to the CPU has been established.
- The CPU is in "RUN" mode.

#### Procedure

If you want to test the controlled system by specifying a manual value, use "Manual mode" in the commissioning window.

To define a manual value, follow these steps:

1. Double-click the "PID\_Temp > Commissioning" entry in the project tree.
2. Activate the "Monitor all"  button or start the trend view.

An online connection will be established.


3. Select the "Manual mode" check box in the "Online status of controller" area.

PID\_Temp operates in manual mode. The most recent current output value remains in effect.

4. Enter the manual value in the editable field as a % value.

If cooling is activated in the basic settings, enter the manual value as follows:

- Enter a positive manual value to output the value at the outputs for heating.
- Enter a negative manual value to output the value at the outputs for cooling.

5. Click the  icon.

#### Result

The manual value is written to the CPU and immediately goes into effect.

Clear the "Manual mode" check box if the output value is to be specified again by the PID controller.

The switchover to automatic mode is bumpless.

### 6.3.5 Substitute setpoint

The following section describes how you can use the substitute setpoint in the commissioning window of the "PID\_Temp" technology object.



#### Requirement

- The "PID\_Temp" instruction is called in a cyclic interrupt OB.
- An online connection to the CPU has been established.
- The CPU is in "RUN" mode.

#### Procedure

If you want to use a different value as the setpoint than that specified at the "Setpoint" parameter (for example to tune a slave in a cascade), use the substitute setpoint in the commissioning window.

Proceed as follows to specify a substitute setpoint:

1. Double-click the "PID\_Temp > Commissioning" entry in the project tree.
2. Activate the "Monitor all"  button or start the trend view.  
An online connection will be established.
3. Select the "Subst.Setpoint" check box in the "Online status of controller" section.  
The substitute setpoint (SubstituteSetpoint tag) is initialized with the most recently updated setpoint and now used.
4. Enter the substitute setpoint in the editable field.
5. Click the  icon.

#### Result

The substitute setpoint is written to the CPU and immediately goes into effect.

Clear the "Subst.Setpoint" check box if the value at the "Setpoint" parameter is to be used again as setpoint.

The switchover is not bumpless.

### 6.3.6 Cascade commissioning

Information about cascade commissioning with PID\_Temp is available under Commissioning (Page 204).

## 6.4 Cascade control with PID\_Temp

### 6.4.1 Introduction

In cascade control, several control loops are nested within each other. In the process, slaves receive their setpoint (Setpoint) from the output value (OutputHeat) of the respective higher-level master.

A prerequisite for establishing a cascade control system is that the controlled system can be divided into subsystems, each with its own measured variable.

Setpoint specification for the controlled variable is carried out at the outmost master.

The output value of the innermost slave is applied to the actuator and thus acts on the controlled system.

The following major advantages result from the use of a cascade control system in comparison with a single-loop control system:

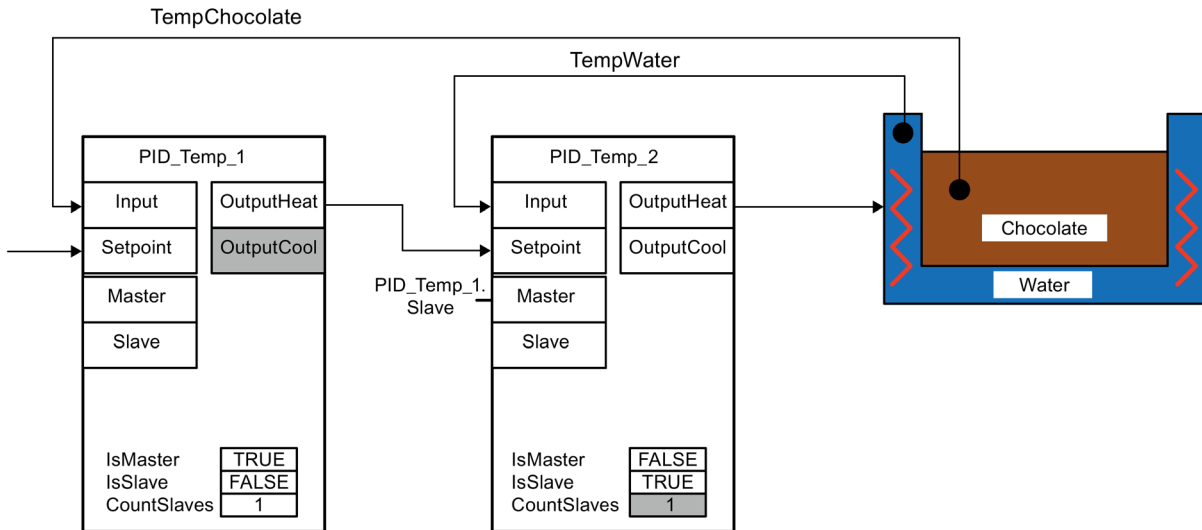
- Thanks to the additional subordinate control loops, disturbances which occur there are corrected quickly. Their influence on the controlled variable is reduced considerably. The disturbance behavior is thus improved.
- The subordinate control loops act in linearizing form. The negative effects of such non-linearities on the controlled variable are thus moderated.

PID\_Temp offers the following functionality especially for use in cascade control systems:

- Specification of a substitute setpoint
- Exchange of status information between master and slave (for example, current operating mode)
- Different Anti-Wind-Up modes (response of the master to limitation of its slave)

## Example

The following block diagram shows a cascade control system with PID\_Temp using the simplified example of a chocolate melting unit:



The PID\_Temp\_1 master compares the process value of the chocolate temperature (TempChocolate) with the setpoint specification by the user at the Setpoint parameter. Its output value OutputHeat forms the setpoint of the slave PID\_Temp\_2.

PID\_Temp\_2 attempts to regulate the process value of the water-bath temperature (TempWater) to this setpoint. The output value of PID\_Temp\_2 acts directly on the actuator of the controlled system (heating of the water bath) and thus influences the water-bath temperature. The water-bath temperature in turn has an effect on the chocolate temperature.

## FAQ

For more information, see the following FAQs in the Siemens Industry Online Support:

- Entry ID 103526819 (<https://support.industry.siemens.com/cs/ww/en/view/103526819>)

## See also

Program creation (Page 200)

## 6.4.2 Program creation

Observe the following points during program creation:

- Number of PID\_Temp instances

The number of different PID\_Temp instances called up in a cyclic interrupt OB has to agree with the number of concatenated measured variables in the process.

There are two concatenated measured variables in the example: TempChocolate and TempWater. Therefore two PID\_Temp instances are required.

- Call sequence

A master has to be called before its slaves in the same cyclic interrupt OB.

The outermost master at which the user setpoint is specified is called first.

The slave whose setpoint is specified by the outermost master is called next, etc.

The innermost slave that acts on the actuator of the process with its output value is called last.

In the example, PID\_Temp\_1 is called before PID\_Temp\_2.

- Interconnection of the measured variables

The outermost master is interconnected with the outermost measured variable that is to be regulated to the user setpoint.

The innermost slave is interconnected with the innermost measured variable that is influenced directly by the actuator.

Interconnection of the measured variables with PID\_Temp is carried out with the parameters Input or Input\_PER.

In the example, the outermost measured variable TempChocolate is interconnected with PID\_Temp\_1 and the innermost measured variable TempWater with PID\_Temp\_2.

- Interconnection of the output value of the master to the setpoint of the slave

The output value (OutputHeat) of a master has to be assigned to the setpoint (Setpoint) of its slave.

This interconnection can be carried out in the programming editor or automatically in the Inspector window of the slave in the basic settings via the selection of the master.

If required, you can insert your own filter or scaling functions, for example in order to adapt the output value range of the master to the setpoint/process value range of the slave.

In the example, OutputHeat of PID\_Temp\_1 is assigned to Setpoint of PID\_Temp\_2.



- Interconnection of the interface for information exchange between master and slave

The "Slave" parameter of a master has to be assigned to the "Master" parameter of all its directly subordinate slaves (which receive their setpoint from this master). The assignment should be carried out via the interface of the slave in order to allow the interconnection of a master with multiple slaves and the display of the interconnection in the Inspector window of the slave in the basic settings.

This interconnection can be carried out in the programming editor or automatically in the Inspector window of the slave in the basic settings via the selection of the master.

The Anti-Wind-Up functionality and the evaluation of the slave operating modes at the master can only function correctly if this interconnection is carried out.

In the example, the "Slave" parameter of PID\_Temp\_1 is assigned to the "Master" parameter of PID\_Temp\_2.

Program code of the example using SCL (without assignment of the output value of the slave to the actuator):

```
"PID_Temp_1" (Input:="TempChocolate");  
"PID_Temp_2" (Input:="TempWater", Master := "PID_Temp_1".Slave, Setpoint :=  
"PID_Temp_1".OutputHeat);
```

## See also

PID\_Temp ActivateRecoverMode tag (Page 453)

### 6.4.3 Configuration

You can carry out the configuration via your user program, the configuration editor or the Inspector window of the PID\_Temp call.

When using PID\_Temp in a cascade control system, ensure the correct configuration of the settings specified below.

If a PID\_Temp instance receives its setpoint from a superior master controller and outputs its output value in turn to a subordinate slave controller, this PID\_Temp instance is both a master controller and a slave controller simultaneously. Both configurations listed below have to be carried out for such a PID\_Temp instance. This is the case, for example, for the middle PID\_Temp instance in a cascade control system with three concatenated measured variables and three PID\_Temp instances.

#### Configuration of a master

Setting in the configuration editor or Inspector window	DB parameter	Explanation
Basic settings → Cascade: Activate "Controller is master" check box	Config.Cascade.IsMaster = TRUE	Activates this controller as a master in a cascade
Basic settings → Cascade: Number of slaves	Config.Cascade.CountSlaves	Number of directly subordinate slaves that receive their setpoint directly from this master
Basic settings → Input/output parameters: Selection of the output value (heating) = OutputHeat	Config.Output.Heat.Select = 0	The master only uses the output parameter OutputHeat. OutputHeat_PWM and OutputHeat_PER are deactivated.
Basic settings → Input/output parameters: Clear "Activate cooling" check box	Config.ActivateCooling = FALSE	The cooling has to be deactivated at a master.
Output settings → Output limits and scaling → OutputHeat / OutputCool: PID output value low limit (heating), PID output value high limit (heating), Scaled low output value (heating), Scaled high output value (heating)	Config.Output.Heat.PidLowerLimit, Config.Output.Heat.PidUpperLimit, Config.Output.Heat.LowerScaling, Config.Output.Heat.UpperScaling	If no own scaling function is used when assigning OutputHeat of the master to Setpoint of the slave, it may be necessary to adapt the output value limits and the output scaling of the master to the setpoint/process value range of the slave.

Setting in the configuration editor or Inspector window	DB parameter	Explanation
<p>This tag is not available in the Inspector window or in the functional view of the configuration editor.</p> <p>You can change it via the parameter view of the configuration editor.</p>	Config.Cascade.AntiWindUpMode	<p>The Anti-Wind-Up mode determines how the integral action of this master is treated if directly subordinate slaves reach their output value limits.</p> <p>Options are:</p> <ul style="list-style-type: none"> <li>• AntiWindUpMode = 0: The AntiWindUp functionality is deactivated. The master does not react to the limitation of its slaves.</li> <li>• AntiWindUpMode = 1 (default): The integral action of the master is reduced in the relationship "Slaves in limitation/Number of slaves". This reduces the effects of the limitation on the control behavior.</li> <li>• AntiWindUpMode = 2: The integral action of the master is held as soon as a slave is in limitation.</li> </ul>

### Configuration of a slave

Setting in the configuration editor or Inspector window	DB parameter	Explanation
<p>Basic settings → Cascade: Select the "Controller is slave" check box</p>	Config.Cascade.IsSlave = TRUE	Activates this controller as a slave in a cascade

### 6.4.4 Commissioning

After compiling and loading of the program, you can start commissioning of the cascade control system.

Begin with the innermost slave at commissioning (implementation of tuning or change to automatic mode with existing PID parameters) and continue outwards until the outermost master has been reached.

In the above example, commissioning starts with PID\_Temp\_2 and is continued with PID\_Temp\_1.

#### Tuning the slave

Tuning of PID\_Temp requires a constant setpoint. Therefore, activate the substitute setpoint of a slave (SubstituteSetpoint and SubstituteSetpointOn tags) to tune the slave or set the associated master to manual mode with a corresponding manual value. This ensures that the setpoint of the slave remains constant during tuning.

#### Tuning the master

In order for a master to influence the process or to carry out tuning, all the downstream slaves have to be in automatic mode and their substitute setpoint has to be deactivated. A master evaluates these conditions through the interface for information exchange between master and slave (Master parameter and Slave parameter) and displays the current state at the AllSlaveAutomaticState and NoSlaveSubstituteSetpoint tags. Corresponding status messages are output in the commissioning editor.

Status message in the commissioning editor of the master	DB parameter of the master	Correction
One or more slaves are not in automatic mode.	AllSlaveAutomaticState = FALSE, NoSlaveSubstituteSetpoint = TRUE	First, carry out commissioning of all downstream slaves.
One or more slaves have activated the substitute setpoint.	AllSlaveAutomaticState = TRUE, NoSlaveSubstituteSetpoint = FALSE	Ensure that the following conditions are fulfilled before carrying out tuning or activating manual mode or automatic mode of the master:
One or more slaves are not in automatic mode and have activated the substitute setpoint.	AllSlaveAutomaticState = FALSE, NoSlaveSubstituteSetpoint = FALSE	<ul style="list-style-type: none"> <li>• All downstream slaves are in automatic mode (state = 3).</li> <li>• All downstream slaves have deactivated the substitute setpoint (SubstituteSetpointOn = FALSE).</li> </ul>

If pretuning or fine tuning is started for a master, PID\_Temp aborts tuning in the following cases and displays an error with ErrorBits = DW#16#0200000:

- One or more slaves are not in automatic mode (AllSlaveAutomaticState = FALSE)
- One or more slaves have activated the substitute setpoint (NoSlaveSubstituteSetpoint = FALSE).

The subsequent operating mode changeover depends on ActivateRecoverMode.

### 6.4.5 Substitute setpoint

In order to specify a setpoint, PID\_Temp offers a substitute setpoint at the SubstituteSetpoint tag in addition to the Setpoint parameter. This can be activated by setting SubstituteSetpointOn = TRUE or by selecting the corresponding check box in the commissioning editor.

The substitute setpoint allows you to specify the setpoint temporarily directly at the slave, for example during commissioning or tuning.

In this case, the interconnection of the output value of the master with the setpoint of the slave that is required for normal operation of the cascade control system does not have to be changed in the program

In order for a master to influence the process or to carry out tuning, the substitute setpoint has to be deactivated at all downstream slaves.

You can monitor the currently effective setpoint as it is used by the PID algorithm for calculation at the CurrentSetpoint tag.

### 6.4.6 Operating modes and fault response

The master or slave of a PID\_Temp instance does not change the operating mode of this PID\_Temp instance.

If a fault occurs at one of its slaves, the master remains in its current operating mode.

If a fault occurs at its master, the slave remains in its current operating mode. However, further operation of the slave then depends on the fault and the configured fault response of the master since the output value of the master is used as the setpoint of the slave:

- If ActivateRecoverMode = TRUE is configured at the master. and the fault does not prevent the calculation of OutputHeat, the fault does not have any effect on the slave.
- If ActivateRecoverMode = TRUE is configured at the master and the fault prevents the calculation of OutputHeat, the master outputs the last output value or the configured substitute output value SubstituteOutput, depending on SetSubstituteOutput. This is then used by the slave as the setpoint.

PID\_Temp is preconfigured so that the substitute output value 0.0 is output in this case (ActivateRecoverMode = TRUE, SetSubstituteOutput = TRUE, SubstituteOutput = 0.0). Configure a suitable substitute output value for your application or activate the use of the last valid PID output value (SetSubstituteOutput = FALSE).

- If ActivateRecoverMode = FALSE is configured at the master, the master changes to the "Inactive" mode when a fault occurs and outputs OutputHeat = 0.0. The slave then uses 0.0 as the setpoint.

The fault response is located in the output settings in the configuration editor.

## 6.5 Multi-zone controlling with PID\_Temp

### Introduction

In a multi-zone control system, several sections, so-called zones, of a plant are controlled simultaneously to different temperatures. A multi-zone control system is characterized by the mutual influence of the temperature zones through thermal coupling, i.e. the process value of one zone can influence the process value of a different zone through thermal coupling. The strength that this influence has depends on the structure of the plant and the selected operating points of the zones.

Example: Extrusion plant as it is used, for example, in plastics processing.

The substance mixture that passes through the extruder has to be controlled to different temperatures for optimal processing. For example, different temperatures can be required at the filling point of the extruder than at the outlet nozzle. The individual temperature zones mutually influence each other through thermal coupling.

When PID\_Temp is used in multi-zone control systems, each temperature zone is controlled by a separate PID\_Temp instance.

Observe the following explanations if you want to use the PID\_Temp in a multi-zone control system.

### Separate pretuning for heating and cooling

Initial commissioning of a plant as a rule begins with the carrying out of pretuning in order to carry out initial setting of the PID parameters and control to the operating point. The pretuning for multi-zone control systems is often carried out simultaneously for all zones.

PID\_Temp offers the possibility of carrying out pretuning for heating and cooling in one step (Mode = 1, Heat.EnableTuning = TRUE, Cool.EnableTuning = TRUE) for controllers with activated cooling and PID parameter switching as the method for heating/cooling (Config.ActivateCooling = TRUE, Config.AdvancedCooling = TRUE).

However, it is advisable not to use this tuning for simultaneous pretuning of several PID\_Temp instances in a multi-zone control system. Instead, first carry out the pretuning for heating (Mode = 1, Heat.EnableTuning = TRUE, Cool.EnableTuning = FALSE) and the pretuning for cooling (Mode = 1, Heat.EnableTuning = FALSE, Cool.EnableTuning = TRUE) separately.

Pretuning for cooling should not be started until all zones have completed pretuning for heating and have reached their operating points.

This reduces mutual influencing through thermal coupling between the zones during tuning.

### Adapting the delay time

If PID\_Temp is used in a multi-zone control system with strong thermal couplings between the zones, you should ensure that the adaption of the delay time is deactivated for pretuning with PIDSelfTune.SUT.AdaptDelayTime = 0. Otherwise, the determination of the delay time can be incorrect if the cooling of a zone is prevented by the thermal influence of other zones during the adapting of the delay time (heating is deactivated in this phase).

## Temporary deactivation of cooling

PID\_Temp offers the possibility of deactivating cooling temporarily in automatic mode for controllers with active cooling (Config.ActivateCooling = TRUE) by setting DisableCooling = TRUE.

This ensures that this controller does not cool in automatic mode during commissioning while the controllers of other zones have not yet completed tuning of heating. The tuning could otherwise be influenced negatively by the thermal coupling between the zones.

## Procedure

You can proceed as follows during the commissioning of multi-zone control systems with relevant thermal couplings:

1. Set DisableCooling = TRUE for all controllers with activated cooling.
2. Set PIDSelfTune.SUT.AdaptDelayTime = 0 for all controllers.
3. Specify the desired setpoints (Setpoint parameter) and start pretuning for heating (Mode = 1, Heat.EnableTuning = TRUE, Cool.EnableTuning = FALSE) simultaneously for all controllers.
4. Wait until all the controllers have completed pretuning for heating.
5. Set DisableCooling = FALSE for all controllers with activated cooling.
6. Wait until the process values of all the zones are steady and close to the respective setpoint.  
  
If the setpoint cannot be reached permanently for a zone, the heating or cooling actuator is too weak.
7. Start pretuning for cooling (Mode = 1, Heat.EnableTuning = FALSE, Cool.EnableTuning = TRUE) for all controllers with activated cooling.

---

### Note

#### Limit violation of the process value

If the cooling is deactivated in automatic mode with DisableCooling = TRUE, this can cause the process value to exceed the setpoint and the process value limits while DisableCooling = TRUE. Observe the process values and intervene, if appropriate, if you use DisableCooling.

---

### Note

#### Multi-zone control systems

For multi-zone control systems, the thermal couplings between the zones can result in increased overshoots, permanent or temporary violation of limits and permanent or temporary control deviations during commissioning or operation. Observe the process values and be ready to intervene. Depending on the system, it can be necessary to deviate from the procedure described above.

---

### Synchronization of several fine tuning processes

If fine tuning is started from automatic mode with `PIDSelfTune.TIR.RunIn = FALSE`, `PID_Temp` tries to reach the setpoint with PID controlling and the current PID parameters. The actual tuning does not start until the setpoint is reached. The time required to reach the setpoint can be different for the individual zones of a multi-zone control system.

If you want to carry out fine tuning for several zones simultaneously, `PID_Temp` offers the possibility to synchronize these by waiting with the further tuning steps after the setpoint has been reached.

### Procedure

This ensures that all the controllers have reached their setpoint when the actual tuning steps start. This reduces mutual influencing through thermal coupling between the zones during tuning.

Proceed as follows for controllers for whose zones you want to carry out fine tuning simultaneously:

1. Set `PIDSelfTune.TIR.WaitForControlIn = TRUE` for all controllers.  
These controllers have to be in automatic mode with `PIDSelfTune.TIR.RunIn = FALSE`.
2. Specify the desired setpoints (Setpoint parameters) and start fine tuning for all controllers.
3. Wait until `PIDSelfTune.TIR.ControlInReady = TRUE` at all controllers.
4. Set `PIDSelfTune.TIR.FinishControlIn = TRUE` for all controllers.

All controllers then start the actual tuning simultaneously.



## 6.6 Override control with PID\_Temp

### Override control

In case of override control, two or more controllers share one actuator. Only one controller has access to the actuator at any time and influences the process.

A logic operation decides which controller has access to the actuator. This decision is often made based on a comparison of the output values of all controllers, for example, in case of a maximum selection, the controller with the largest output value gets access to the actuator.

The selection based on the output value requires that all controllers operate in automatic mode. The controllers that do not have an effect on the actuator are updated. This is necessary to prevent windup effects and their negative impacts on the control response and the switchover between the controllers.

PID\_Temp supports override controls as of version 1.1 by offering a simple process for updating the controllers that are not active: By using the tags `OverwriteInitialOutputValue` and `PIDCtrl.PIDInit`, you can pre-assign the integral action of the controller in automatic mode as though the PID algorithm had calculated `PidOutputSum = OverwriteInitialOutputValue` for the PID output value in the last cycle. To do this, `OverwriteInitialOutputValue` is interconnected with the PID output value of the controller that currently has access to the actuator. By setting the bit `PIDCtrl.PIDInit`, you trigger the pre-assignment of the integral action as well as the restart of the controller cycle and the PWM period. The subsequent calculation of the PID output value in the current cycle takes place based on the pre-assigned (and synchronized for all controllers) integral action as well as the proportional action and integral action from the current control deviation. The derivative action is not active during the call with `PIDCtrl.PIDInit = TRUE` and therefore does not contribute to the output value.

This procedure ensures that the calculation of the current PID output value and thus the decision on which controller is to have access to the actuator is only based on the current process state and the PI parameters. Windup effects for controllers that are not active and thus incorrect decisions of the switchover logic are prevented.

### Requirement

- `PIDCtrl.PIDInit` is only effective if the integral action is activated (tags `Retain.CtrlParams.Heat.Ti` and `Retain.CtrlParams.Cool.Ti > 0.0`).
- You must assign `PIDCtrl.PIDInit` and `OverwriteInitialOutputValue` in your user program yourself (see example below). `PID_Temp` does not automatically change these tags.
- `PIDCtrl.PIDInit` is only effective when `PID_Temp` is in automatic mode (parameter `State = 3`).

- If possible, select the sampling time of the PID algorithm ( Retain.CtrlParams.Heat.Cycle and Retain.CtrlParams.Cool.Cycle tags) so that it is identical for all controllers, and call all controllers in the same cyclic interrupt OB. In this way, you ensure that the switchover does not take place within a controller cycle or a PWM period.

---

**Note**

**Constant adaptation of the output value limits**

Instead of the active updating of the controllers without access to the actuator described here, this is implemented alternatively by constant adaptation of the output value limits in other controller systems.

This is not possible with PID\_Temp, because a change of the output value limits is not supported in automatic mode.

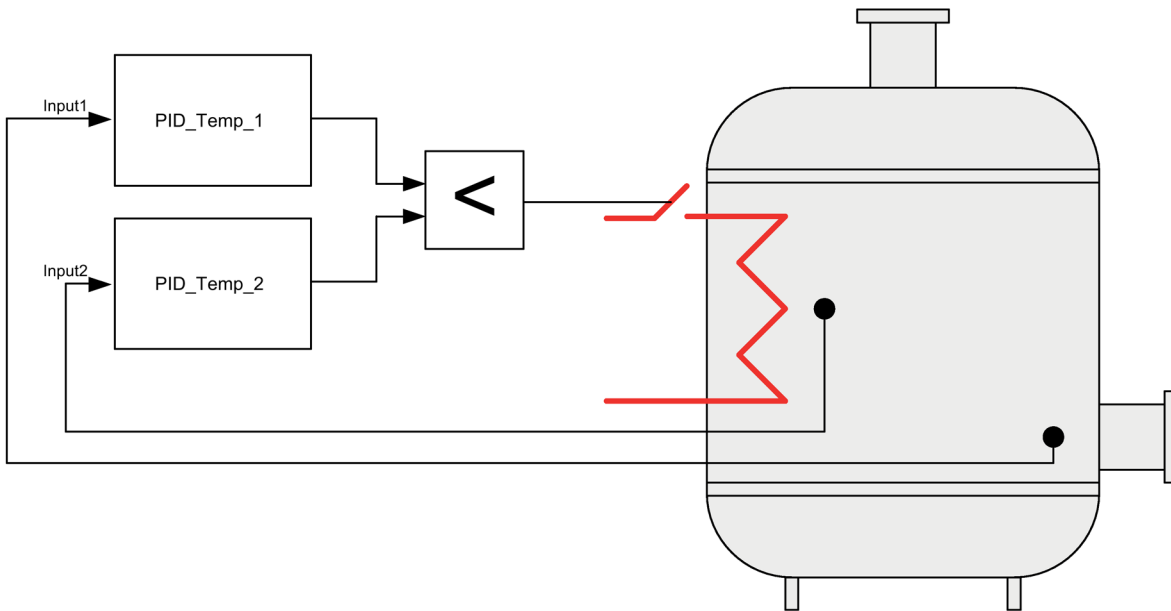
---

**Example: Control of a large boiler**

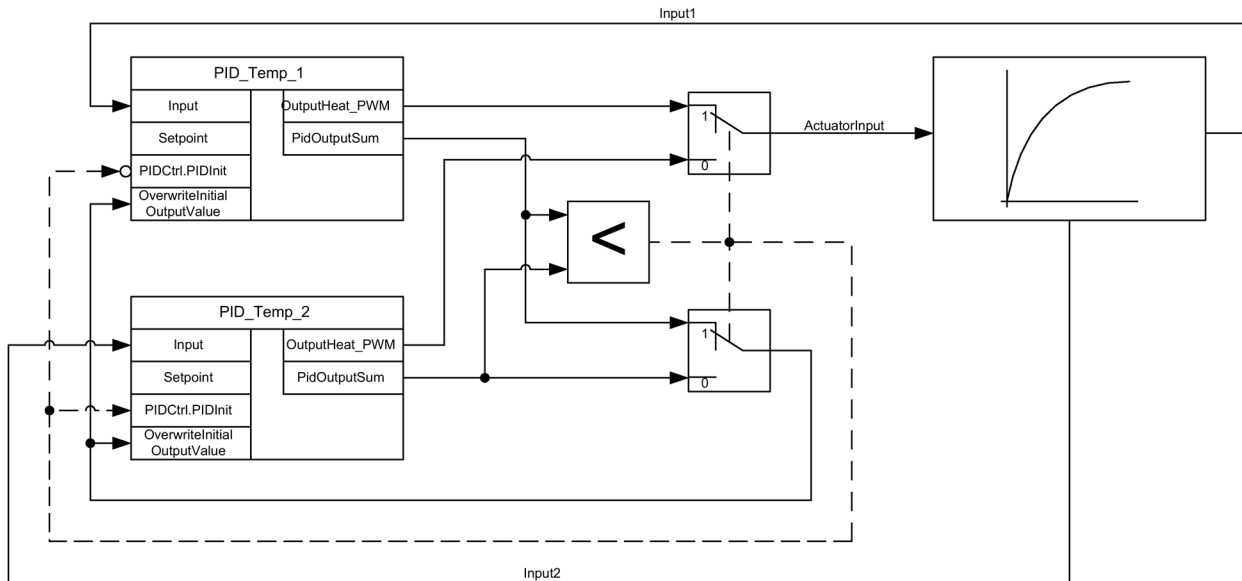
PID\_Temp is used for control of a large boiler.

The main goal is to control the temperature Input1. The controller PID\_Temp\_1 is used for this purpose. In addition, the temperature Input2 is to be kept below a high limit at an additional measuring point with the limiting controller PID\_Temp\_2.

Both temperatures are influenced by only one heater. The output value of the controller corresponds to the heating power.



The heater is controlled with the pulse-width modulated output value of PID\_Temp (parameter OutputHeat\_PWM) by writing the program tag ActuatorInput. The setpoint for the temperature Input1 is specified at the parameter PID\_Temp\_1.Setpoint. The temperature high limit for the additional measuring point is specified as setpoint at the parameter PID\_Temp\_2.Setpoint.



Both controllers must share one heater as shared actuator. The logic that decides which controller gets access to the actuator is implemented by a minimum selection of the PID output value (in Real format, parameter PidOutputSum) in this case. Because the PID output value corresponds to the heating power, the controller that requires lower heating power gets the control.

In normal operation of the plant, the process value of the main controlled variable corresponds to the setpoint. The main controller PID\_Temp\_1 has settled on a stationary PID output value PID\_Temp\_1.PidOutputSum. The process value of the limiting controller Input2 in normal operation is significantly below the high limit that is specified as setpoint for PID\_Temp\_2. The limiting controller therefore wants to increase the heating power to increase its process value, which means it will calculate a PID output value PID\_Temp\_2.PidOutputSum that is greater than the main controller PID\_Temp\_1.PidOutputSum. The minimum selection of the switchover logic therefore gives the main controller PID\_Temp\_1 continued access to the actuator. In addition, it is ensured that PID\_Temp\_2 is updated by means of the assignments PID\_Temp\_2.OverwriteInitialOutputValue = PID\_Temp\_1.PidOutputSum and PID\_Temp\_2.PIDCtrl.PIDInit = TRUE.

If Input2 now approaches the high limit or exceeds it, for example due to a fault, the limiting controller PID\_Temp\_2 calculates a smaller PID output value to restrict the heating power and thus reduce Input2. If PID\_Temp\_2.PidOutputSum is smaller than PID\_Temp\_1.PidOutputSum, the limiting controller PID\_Temp\_2 receives access to the actuator through the minimum selection and reduces the heating power. It is ensured that PID\_Temp\_1 is updated by means of the assignments PID\_Temp\_1.OverwriteInitialOutputValue = PID\_Temp\_2.PidOutputSum and PID\_Temp\_1.PIDCtrl.PIDInit = TRUE.

6.6 Override control with PID\_Temp

---

The temperature at the additional measuring point Input2 drops. The temperature of the main controlled variable Input1 drops as well and cannot be held at the setpoint any longer.

Once the fault has been remedied, the Input2 will continue to drop and the heating power is further increased by the limiting controller. As soon as the main controller has calculated a lower heating power as output value, the plant returns to normal operation so that the main controller PID\_Temp\_1 once again has access to the actuator. This example can be implemented with the following SCL program code:

```
"PID Temp 1"(Input := "Input1");  
"PID Temp 2"(Input := "Input2");  
IF "PID Temp 1".PidOutputSum <= "PID Temp 2".PidOutputSum THEN  
  "ActuatorInput" := "PID_Temp_1".OutputHeat_PWM;  
  
  "PID_Temp_1".PIDCtrl.PIDInit := FALSE;  
  
  "PID_Temp_2".PIDCtrl.PIDInit := TRUE;  
  
  "PID_Temp_2".OverwriteInitialOutputValue := "PID_Temp_1".PidOutputSum;  
ELSE  
  "ActuatorInput" := "PID_Temp_2".OutputHeat_PWM;  
  
  "PID_Temp_1".PIDCtrl.PIDInit := TRUE;  
  
  "PID_Temp_2".PIDCtrl.PIDInit := FALSE;  
  
  "PID_Temp_1".OverwriteInitialOutputValue := "PID_Temp_2".PidOutputSum;  
END IF;
```

## 6.7 Simulating PID\_Temp with PLCSIM

---

### Note

#### Simulation with PLCSIM

The simulation of PID\_Temp with PLCSIM for CPU S7-1200 is not supported.

PID\_TEMP can be simulated only for CPU S7-1500 with PLCSIM.

For the simulation with PLCSIM, the time behavior of the simulated PLC is not exactly identical to that of a "real" PLC. The actual cycle clock of a cyclic interrupt OB can have larger fluctuations with a simulated PLC than with "real" PLCs.

In the standard configuration, PID\_Temp determines the time between calls automatically and monitors them for fluctuations.

For the simulation of PID\_Temp with PLCSIM, for example, a sampling time error (ErrorBits = DW#16#00000800) can therefore be detected.

This results in ongoing tuning being aborted.

The response in automatic mode depends on the value of the ActivateRecoverMode tag.

To prevent this from happening, you should configure PID\_Temp for simulation with PLCSIM as follows:

- CycleTime.EnEstimation = FALSE
  - CycleTime.EnMonitoring = FALSE
  - CycleTime.Value: Assign the cycle clock of the calling cyclic interrupt OB in seconds to this tag.
-

## Using PID basic functions

### 7.1 CONT\_C

#### 7.1.1 Technology object CONT\_C

The technology object CONT\_C provides a continual PID-controller for automatic and manual mode. It corresponds to the instance data block of the instruction CONT\_C. You can configure a pulse controller using the PULSEGEN instruction.

The proportional, integral (INT) and differential components (DIF) are switched parallel to each other and can be turned on and off individually. With this, P-, I, PI-, PD- and PID-controller can be set.

S7-1500

All parameters and tags of the technology object are retentive and can only be changed during download to the device if you completely download CONT\_C.

#### See also

[Overview of software controller \(Page 38\)](#)

[Add technology objects \(Page 40\)](#)

[Configure technology objects \(Page 45\)](#)

[CONT\\_C \(Page 461\)](#)

[Downloading technology objects to device \(Page 72\)](#)

## 7.1.2 Configure controller difference CONT\_C

### Use process value periphery

To use the process value in the periphery format at the PV\_PER input parameter, follow these steps:

1. Select the "Enable I/O" check box.
2. If the process value is available as a physical size, enter the factor and offset for the scaling in percent.

The process value is then determined according to the following formula:

$$PV = PV\_PER \times PV\_FAC + PV\_OFF$$

### Use internal process values

To use the process value in the floating-point format at the PV\_IN input parameter, follow these steps:

1. Clear the "Enable I/O" check box.

### Control deviation

Set a dead zone range under the following requirement:

- The process value signal is noisy.
- The controller gain is high.
- The derivative action is activated.

The noise component of the process value causes strong deviations of the output value in this case. The dead zone suppresses the noise component in the steady controller state. The dead zone range specifies the size of the dead zone. With a dead zone range of 0.0, the dead zone is turned off.

### See also

How CONT\_C works (Page 462)

### 7.1.3 Configure the controller algorithm CONT\_C

#### General

To determine which components of the control algorithm are activated, proceed as follows:

1. Select an entry from the "Controller structure" list.  
You can only specify required parameters for the selected controller structure.

#### Proportional action

1. If the controller structure contains a proportional action, enter the "proportional gain".

#### Integral action

1. If the controller structure contains an integral action, enter the integral action time.
2. To give the integral action an initialization value, select the check box "Initialize integral action" and enter the initialization value.
3. In order to permanently set the integral action to this initialization value, select the "Integral action hold" check box.

#### Derivative action

1. If the controller structure contains a derivative action, enter the derivative action time, the derivative action weighting and the delay time.

#### See also

How CONT\_C works (Page 462)



## 7.1.4 Configure the output value CONT\_C

### General

You can set CONT\_C in the manual or automatic mode.

1. To set a manual manipulated value, activate the option "Activate manual mode" option check box.

You can specify a manual manipulated value on the input parameter MAN.

### Manipulated value limits

The manipulated value is limited at the top and bottom so that it can only accept valid values. You cannot turn off the limitation. Exceeding the limits is displayed through the output parameters QLMN\_HLM and QLMN\_LLM.

1. Enter a value for the high and low manipulated value limits.  
If the manipulated value is a physical size, the units for the high and low manipulated value limits must match.

### Scaling

The manipulated value can be scaled for output as a floating point and periphery value through a factor and an offset according to the following formula.

Scaled manipulated value = manipulated value x factor + offset

Default is a factor of 1.0 and an offset of 0.0.

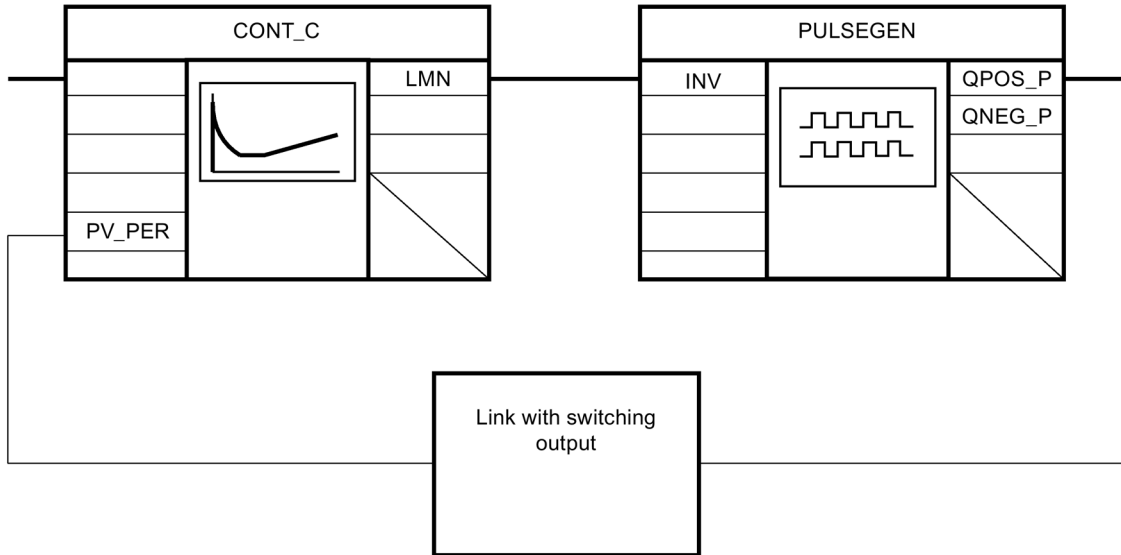
1. Enter a value for the factor and offset.

### See also

How CONT\_C works (Page 462)

### 7.1.5 Programming a pulse controller

With the continuous controller CONT\_C and the pulse shaper PULSEGEN, you can implement a fixed setpoint controller with a switching output for proportional actuators. The following figure shows the signal flow of the control loop.



The continuous controller CONT\_C forms the output value LMN that is converted by the pulse shaper PULSEGEN into pulse/break signals QPOS\_P or QNEG\_P.

#### See also

PULSEGEN (Page 473)

## 7.1.6 Commissioning CONT\_C

### Requirements



- The instruction and the technology object are loaded on the CPU.

### Procedure

In order to manually determine the optimal PID parameter, proceed as follows:

1. Click the "Start" icon.

If there is no online connection, this will be established. The current values for the setpoint, process value and output value are recorded.

2. Enter new PID parameters in the "P", "I", "D" and "Delay time" fields.
3. Click on the icon  "Send parameter to CPU" in the "Tuning" group.
4. Select the "Change setpoint" check box in the "Current values" group.
5. Enter a new setpoint and click in the "Current Values" group on the icon .
6. Clear the "Manual mode" check box.

The controller works with the new PID parameters and controls the new setpoint.

7. Check the quality of the PID parameter to check the curve points.
8. Repeat steps 2 to 6 until you are satisfied with the controller results.

## 7.2 CONT\_S

### 7.2.1 Technology object CONT\_S

The technology object CONT\_S provides a step controller for actuators with integrating behavior and is used to control technical temperature processes with binary output value output signals. The technology object corresponds to the instance data block of the CONT\_S instruction. The operating principle is based on the PI control algorithm of the sampling controller. The step controller operates without a position feedback signal. Both manual and automatic mode are possible.

S7-1500

All parameters and tags of the technology object are retentive and can only be changed during download to the device if you completely download CONT\_S.

#### See also

[Overview of software controller \(Page 38\)](#)

[Add technology objects \(Page 40\)](#)

[Configure technology objects \(Page 45\)](#)

[CONT\\_S \(Page 468\)](#)

[Downloading technology objects to device \(Page 72\)](#)

## 7.2.2 Configure controller difference CONT\_S

### Use process value periphery

To use the process value in the periphery format at the PV\_PER input parameter, follow these steps:

1. Select the "Enable I/O" check box.
2. If the process value is available as a physical quantity, enter the factor and offset for the scaling in percent.

The process value is then determined according to the following formula:

$$PV = PV\_PER \times PV\_FAC + PV\_OFF$$

### Use internal process values

To use the process value in the floating-point format at the PV\_IN input parameter, follow these steps:

1. Clear the "Enable I/O" check box.

### Control deviation

Set a deadband range under the following requirement:

- The process value signal is noisy.
- The controller gain is high.
- The derivative action is activated.

The noise component of the process value causes strong deviations of the manipulated variable in this case. The deadband suppresses the noise component in the steady controller state. The deadband range specifies the size of the deadband. With a deadband range of 0.0, the deadband is turned off.

### See also

Mode of operation CONT\_S (Page 469)

## 7.2.3 Configuring control algorithm CONT\_S

### PID algorithm

1. Enter the "proportional amplification" for the P-component.
2. Enter the integration time for the time behavior of the I-component.  
With an integration time of 0.0, the I-component is switched off.

### See also

Mode of operation CONT\_S (Page 469)

## 7.2.4 Configure manipulated value CONT\_S

### General

You can set CONT\_S in the manual or automatic mode.

1. To set a manual manipulated value, activate the "Activate manual mode" option check box.  
Enter a manual manipulated value for the input parameters LMNUP and LMNDN.

### Pulse generator

1. Enter the minimum impulse duration and minimum pause duration.  
The values must be greater than or equal to the cycle time for the input parameter CYCLE. The frequency of operation is reduced through this.
2. Enter the motor setting time.  
The value must be greater than or equal to the cycle time of the input parameter CYCLE.

### See also

Mode of operation CONT\_S (Page 469)



## 7.2.5 Commissioning CONT\_S

### Requirements

- The instruction and the technology object have been loaded to the CPU.

### Procedure

To manually determine the optimal PID parameters, proceed as follows:

1. Click the "Start" icon.  
If there is no online connection, this will be established. The current values for the setpoint, process value and output value are recorded.
2. In the fields "P" and "I", enter a new proportional value and a new integration time.
3. Click on the icon  "Send parameter to CPU" in the "Tuning" group.
4. Select the "Change setpoint" check box in the "Current values" group.
5. Enter a new setpoint and click in the "Current Values" group on the icon .
6. Clear the "Manual mode" check box.  
The controller works with the new parameters and controls the new setpoint.
7. Check the quality of the PID parameter to check the curve points.
8. Repeat steps 2 to 6 until you are satisfied with the controller results.

## 7.3 TCONT\_CP

### 7.3.1 Technology object TCONT\_CP

The technology object TCONT\_CP provides a continual temperature controller with pulse generator. It corresponds to the instance data block of the instruction TCONT\_CP. The operation is based on the PID control algorithm of the sampling controller. Both manual and automatic mode are possible.

The instruction TCONT\_CP calculates the proportional, integral and derivative parameters for your controlled system during pretuning. "Fine tuning" can be used to tune the parameters further. You can also enter the PID parameters manually.

S7-1500

All parameters and tags of the technology object are retentive and can only be changed during download to the device if you completely download TCONT\_CP.

#### See also

Overview of software controller (Page 38)

Add technology objects (Page 40)

Configure technology objects (Page 45)

TCONT\_CP (Page 483)

Downloading technology objects to device (Page 72)

## 7.3.2 Configure TCONT\_CP

### 7.3.2.1 Controller difference

#### Use process value periphery

To use the input parameter PV\_PER, proceed as follows:

1. Select the entry "Periphery" from the "Source" list.
2. Select the "sensor type".  
Depending on the sensor type, the process value is scaled according to different formulas.
  - Standard  
Thermoelements; PT100/Ni100  
 $PV = 0.1 \times PV\_PER \times PV\_FAC + PV\_OFFS$
  - Cooling;  
PT100/Ni100  
 $PV = 0.01 \times PV\_PER \times PV\_FAC + PV\_OFFS$
  - Current/voltage  
 $PV = 100/27648 \times PV\_PER \times PV\_FAC + PV\_OFFS$
3. Enter the factor and offset for the scaling of the process value periphery.

#### Use internal process values

To use the input parameter PV\_IN, proceed as follows:

1. Select the entry "Internal" from the "Source" list.

#### Control deviation

Set a deadband range under the following requirement:

- The process value signal is noisy.
- The controller gain is high.
- The derivative action is activated.

The noise component of the process value causes strong deviations of the manipulated variable in this case. The deadband suppresses the noise component in the steady controller state. The deadband range specifies the size of the deadband. With a deadband range of 0.0, the deadband is turned off.

#### See also

Mode of operation TCONT\_CP (Page 484)



### 7.3.2.2 Controlling algorithm

#### General

1. Enter the "Sampling time PID algorithm".  
A controller sampling time should not exceed 10 % of the determined integratl action time of the controller (TI).
2. If the controller structure contains a proportional action, enter the "proportional gain".  
A negative proportional gain inverts the rule meaning.

#### Proportional action

For changes of the setpoint, it may lead to overshooting of the proportional action. Through the weighting of the proportional action, you can select how strongly the proportional action should react when setpoint changes are made. The weakening of the proportional action is reached through a compensation of the integral action.

1. To weaken the proportional action for setpoint changes, enter a "Proportional action weighting".
  - 1.0: Proportional action for setpoint change is fully effective
  - 0.0: Proportional action for setpoint change is not effective

#### Integral action

With a limitation of the manipulated value, the integral action is stopped. With a control deviation that moves the integral action in the direction of an internal setting range, the integral action is released again.

1. If the controller structure contains an integral action, enter the "integral action time".  
With an integral action time of 0.0, the integral action is switched off.
2. To give the integral action an initialization value, select the "Initialize integral action" check box and enter the "Initialization value".  
Upon restart or COM\_RST = TRUE, the integral action is set to this value.

#### Derivative action

1. If the controller structure contains a derivative action, enter the derivative action time (TD) and the coefficients DT1 (D\_F).  
With switched derivative action, the following equation should be maintained:  
 $TD = 0.5 \times CYCLE \times D\_F$   
The delay time is calculated from this according to the formula:  
delay time = TD/D\_F

#### Set PD-controller with operating point

1. Enter the integral action time 0.0.
2. Activate the "Initialize integral action" check box.
3. Enter the operating point as the initialization value.

### Set P-controller with operating point

1. Set a PD-controller with an operating point.
2. Enter the derivative action time 0.0.  
The derivative action is disabled.

### Control zone

The control zone limits the value range of the control deviation. If the control deviation is outside of this value range, the manipulated value limits are used.

With an occurrence in the control zone, the derivative action leads to a very quick reduction of the manipulated variable. Thus, the control zone only makes sense for switched on derivative actions. Without control zone, only the reducing proportional action would reduce the manipulated value. The control zone leads to a quick oscillation without over/under shooting if the emitted minimum or maximum manipulated values are removed from the manipulated value required for the new operating point.

1. Activate the "Activate" check box in the "control zone" group.
2. Enter a setpoint value in the "Width" input field from which the process value may deviate above or below.

### See also

Mode of operation TCONT\_CP (Page 484)

### 7.3.2.3 Manipulated value continual controller

#### Manipulated value limits

The manipulated value is limited at the top and bottom so that it can only accept valid values. You cannot turn off the limitation. Exceeding the limits is displayed through the output parameters QLMN\_HLM and QLMN\_LLM.

1. Enter a value for the high and low manipulated value limits.

#### Scaling

The manipulated value can be scaled for output as a floating point and periphery value through a factor and an offset according to the following formula.

Scaled manipulated value = manipulated value x factor + offset

Default is a factor of 1.0 and an offset of 0.0.

1. Enter a value for the factor and offset.

#### Pulse generator

The pulse generator must be turned on for a continual controller.

1. Disable the "Activate" option check box in the "Pulse generator" group.

#### See also

Mode of operation TCONT\_CP (Page 484)

### 7.3.2.4 Manipulated value pulse controller

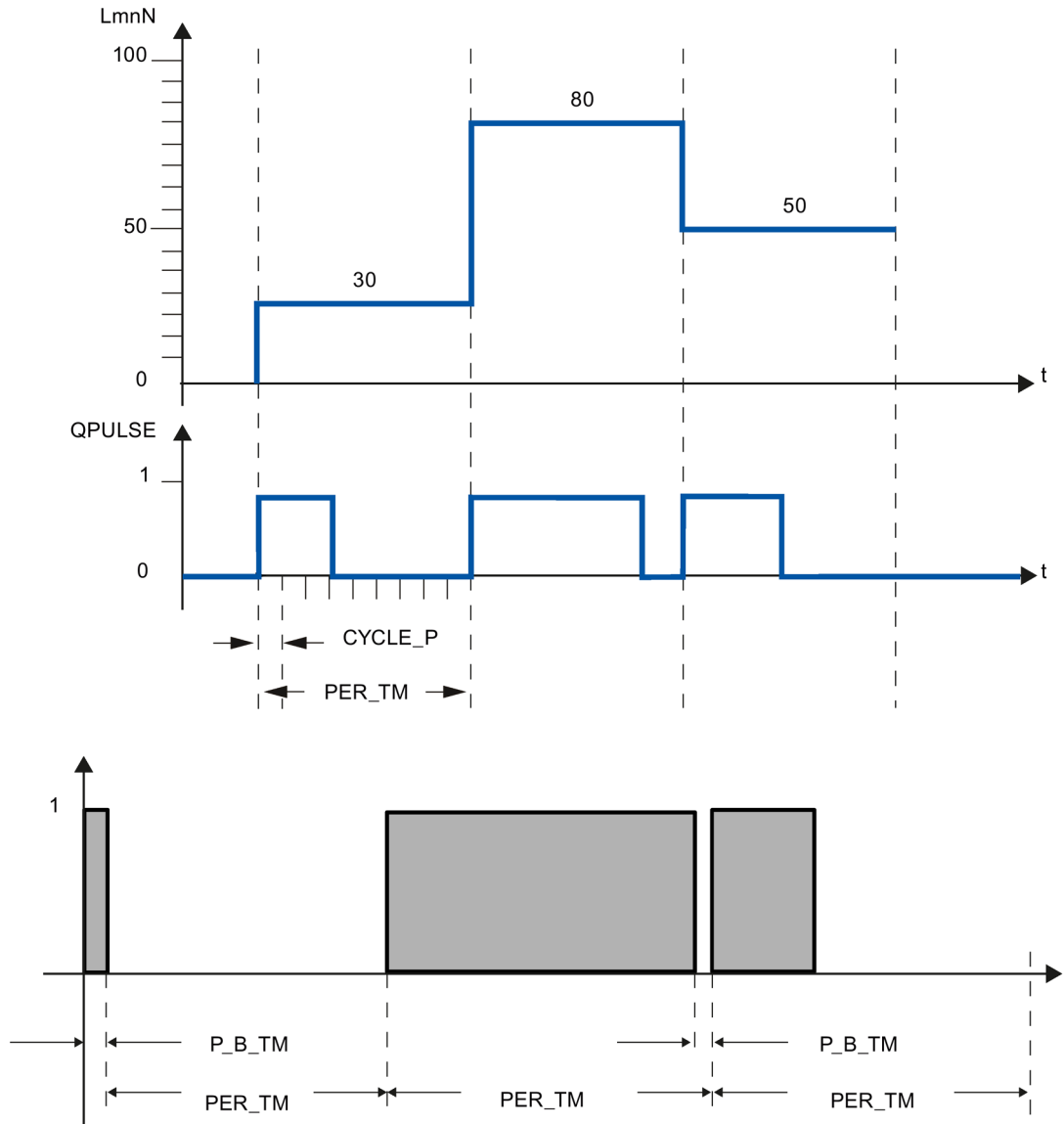
#### Pulse generator

The analog manipulated value (LmnN) can be emitted through pulse-duration modulation on the output parameter QPULSE as an impulse sequence.

To use the pulse generator, proceed as follows:

1. Activate the "Activate" option check box in the "pulse generator" group.
2. Enter the "sampling time pulse generator", the "minimum impulse/break duration" and the "period duration".

The following graphics clarify the connection between the "sampling pulse generator" (CYCLE\_P), the "minimum impulse/break duration" (P\_B\_TM) and the "period duration" (PER\_TM):



### Sampling time pulse generator

The sampling time pulse generator must agree with the time tact of the cyclic interrupt OB being called. The duration of the created impulse is always a whole number factor of this value. For an adequately precise manipulated value resolution, the following relationship should apply:

$$\text{CYCLE\_P} \leq \text{PER\_TM}/50$$

### Minimum impulse/break duration

Through the minimum impulse/break duration, short on or off times on the actuator are avoided. An impulse smaller than P\_B\_TM is suppressed.

Recommended are values  $P\_B\_TM \leq 0.1 \times PER\_TM$ .

### Period duration

The period duration should not exceed 20% of the determined integration time of the controller (TI):

$$PER\_TM \leq TI/5$$

### Example for the effect of the parameter CYCLE\_P, CYCLE and PER\_TM:

Period duration PER\_TM = 10 s

Sampling time PID-algorithm CYCLE = 1 s

Sampling time pulse generator CYCLE\_P = 100 ms.

Every second, a new manipulated value, every 100 ms the comparison of the manipulated value occurs with the previously emitted impulse length and break length.

- If an impulse is emitted, there are 2 possibilities:
  - The calculated manipulated value is larger than the previous impulse length/PER\_TM. Then the impulse is extended.
  - The calculated manipulated value is less than or equal to the previous impulse length/PER\_TM. Then no impulse signal will be emitted.
- If no impulse is emitted, there are also 2 possibilities:
  - The value (100 % - calculated manipulated value) is greater than the previous break length / PER\_TM. Then the break is extended.
  - The value (100 % - calculated manipulated value) is less than or equal to the previous break length / PER\_TM. Then an impulse signal will be emitted.

### See also

Mode of operation TCONT\_CP (Page 484)

Operating principle of the pulse generator (Page 493)

### 7.3.3 Commissioning TCONT\_CP

#### 7.3.3.1 Optimization of TCONT\_CP

##### Application possibilities

The controller optimization for heating or cooling processes from process type I is applicable. But you can use the block for processes with higher levels like process type II or III.

The PI/PID parameters are automatically determined and set. The controller draft is designed for an optimal disruption behavior. The "precise" parameters resulting from this lead to overshooting of 10% to 40% of the jump height for setpoint jump heights.

##### Phases of controller optimization

For the controller optimization, individual phases are run through, which you can read on the parameter PHASE .

##### PHASE = 0

No tuning is running. TCONT\_CP works in automatic or manual mode.

During PHASE = 0, you can make sure that the controlled system fulfills the requirements for an optimization.

At the end of the optimization, TCONT\_CP changes back into PHASE = 0.

##### PHASE = 1

TCONT\_CP is prepared for optimization. PHASE = 1 may only be started if the requirements for an optimization are fulfilled.

During PHASE = 1, the following values are determined:

- Process value noise NOISE\_PV
- Initial slope PVDT0
- Average of the manipulated variable
- Sampling time PID algorithm CYCLE
- Sampling time pulse generator CYCLE\_P

## PHASE = 2

In phase 2, the process value attempts to detect the point of inflection with a constant manipulated variable. This method prevents the point of inflection from being found too early as a result of process variable noise.

With the pulse controller, the process variable is averaged over N pulse cycles and then made available to the controller stage. There is a further averaging of the process variable in the controller stage: Initially, this averaging is inactive; in other words, averaging always takes place over 1 cycle. As long as the noise exceeds a certain level, the number of cycles is doubled.

The period and amplitude of the noise are calculated. The search for the point of inflection is canceled and phase 2 is exited only when the gradient is always smaller than the maximum rise during the estimated period. TU and T\_P\_INF are, however, calculated at the actual point of inflection.

Tuning, however, is only ended when the following two conditions are met:

1. The process value is more than  $2 \cdot \text{NOISE\_PV}$  away from the point of inflection.
2. The process value has exceeded the point of inflection by 20%.

---

### Note

When exciting the process using a setpoint step change, tuning is ended at the latest when the process value exceeds 75% of the setpoint step change (SP\_INT-PV0) (see below).

---

## PHASE = 3, 4, 5

The phases 3, 4 and 5 last 1 cycle each.

In Phase 3, the valid PI/PID parameters are saved before the optimization and the process parameter is calculated.

In Phase 4, the new PI/PID parameters are calculated.

In Phase 5, the new manipulated variable is calculated and the controlled system is given.

## PHASE = 7

The process type is inspected in Phase 7, because TCONT\_CP always changes to automatic mode after optimization. The automatic mode starts with  $\text{LMN} = \text{LMN0} + 0.75 \cdot \text{TUN\_DLMN}$  as a manipulated variable. The testing of the process type occurs **in the automatic mode** with the recently recalculated controller parameters and ends at the latest  $0.35 \cdot \text{TA}$  (equilibrium time) after the point of inflection. If the process order deviates strongly from the estimated value, the controller parameters are newly calculated and STATUS\_D is counted up by 1, otherwise, the controller parameters remain unchanged.

Then the optimization mode is complete and TCONT\_CP is back in PHASE = 0. At the STATUS\_H parameter, you can identify whether the tuning was successfully completed.

### Premature cancellation of the optimization

In Phase 1, 2 or 3, you can cancel the optimization by resetting TUN\_ON = FALSE without calculating new parameters. The controller starts in the automatic mode with  $\text{LMN} = \text{LMN0} + \text{TUN\_DLMN}$ . If the controller was in manual mode before the tuning, the old manual manipulated variable is output.

If the tuning is canceled in Phase 4, 5 or 7 with TUN\_ON = FALSE, the determined controlled parameters are contained until then.

### 7.3.3.2 Requirements for an optimization

#### Transient response

The process must have a stable, asymptotic transient response with time lag.

The process value must settle to steady state after a step change of the manipulated variable. This therefore excludes processes that already show an oscillating response without control, as well as processes with no recovery (integrator in the control system).

 **WARNING**

This may result in death, severe injury or considerable property damage.

During an tuning, the parameter MAN\_ON is ineffective. Through this, the output value or process value may take on undesired - even extreme - values.

The output value is defined through the tuning. To cancel the tuning, you first have to set TUN\_ON = FALSE. This makes MAN\_ON effective again.

#### Guaranteeing a stationary initial state (phase 0)

With lower-frequency oscillations of the process value, for example, due to incorrect controller parameters, the controller must be put in manual mode before the tuning is started and wait for the oscillation to stop. Alternatively, you could switch to a "soft" set PI controller (small loop gain large integration time).

Now you have to wait until the stationary state is reached, this means, until the process value and output value have a steady state. It is also permissible to have an asymptotic transient oscillation or slow drifting of the process value (stationary state, see the following image). The output value must be constant or fluctuate by a constant average.

---

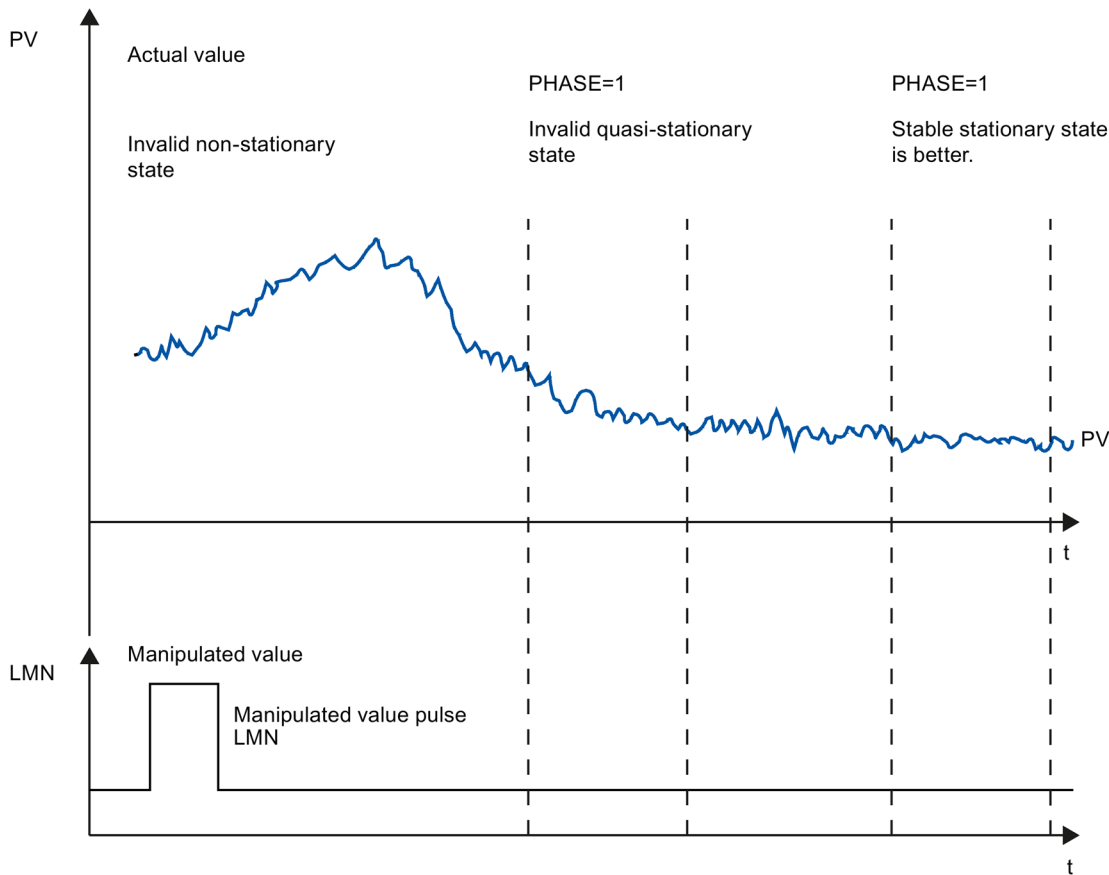
#### Note

Avoid changing the manipulated variable shortly before starting the tuning. A change of the manipulated variable can occur in an unintended manner through the establishment of the test conditions (for example, closing an oven door)! If this does happen, you have to at least wait until the process value has an asymptotic transient oscillation in a stationary state again. Better controller parameters can be reached if you wait until the transient effect has completely subsided.

---



In the following image, the transient oscillation is illustrated in the stationary state:



### Linearity and operating range

The process response must be linear across the operating range. Non-linear response occurs, for example, when an aggregation state changes. Tuning must take place in a linear part of the operating range.

This means, during tuning and normal control operation non-linear effects within the operating range must be insignificant. It is, however possible to retune the process when the operating point changes, providing tuning is repeated in the close vicinity of the new operating point and non-linearity does not occur during tuning.

If a specific static non-linearity (e.g., valve characteristics) is known, it is always advisable to compensate this with a polyline to linearize the process response.

### Disturbance in temperature processes

Disturbances such as the transfer of heat to neighboring zones must not affect the overall temperature process too much. For example, when optimizing the zones of an extruder, all zones must be heated simultaneously.

### 7.3.3.3 Possibilities for optimization

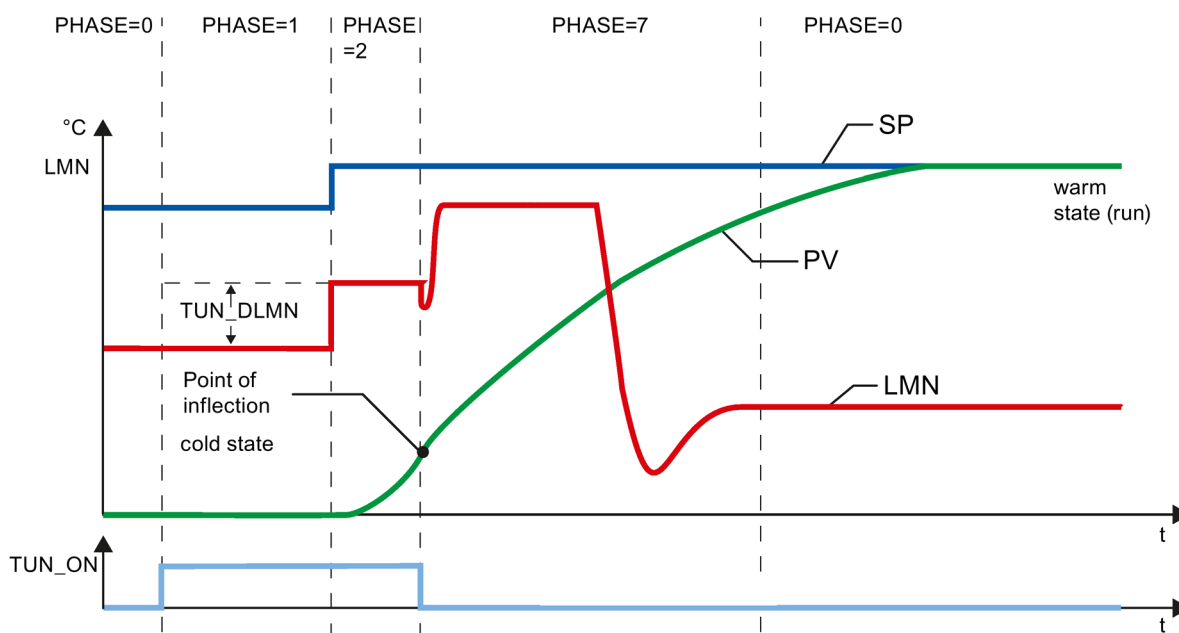
The following possibilities for tuning exist:

- Pretuning
- Fine tuning
- Manual fine-tuning in control mode

#### Pretuning

During this tuning, the working point is approached from the cold state through a setpoint jump.

With TUN\_ON = TRUE, you can establish the tuning readiness. The controller switches from PHASE = 0 to PHASE = 1.



The tuning manipulated variable ( $LMN0 + TUN\_DLMN$ ) is activated by a setpoint change (transition phase 1 → 2). The setpoint is not effective until the inflection point has been reached (automatic mode is not enabled until this point is reached).

The user is responsible for defining the output excitation delta ( $TUN\_DLMN$ ) according to the permitted process value change. The sign of  $TUN\_DLMN$  must be set depending on the intended process value change (take into account the direction in which the control is operating).

The setpoint step change and  $TUN\_DLMN$  must be suitably matched. If the value of  $TUN\_DLMN$  is too high, there is a risk that the point of inflection will not be found before 75% of the setpoint step change is reached.

$TUN\_DLMN$  must nonetheless be high enough to ensure that the process value reaches at least 22% of the setpoint step change. Otherwise, the process will remain in tuning mode (phase 2).

Remedy: Reduce the setpoint value during the inflection point search.

### Note

If processes are extremely sluggish, it is advisable during tuning to specify a target setpoint that is somewhat lower than the desired operating point and to monitor the status bits and PV closely (risk of overshooting).

### Tuning only in the linear range:

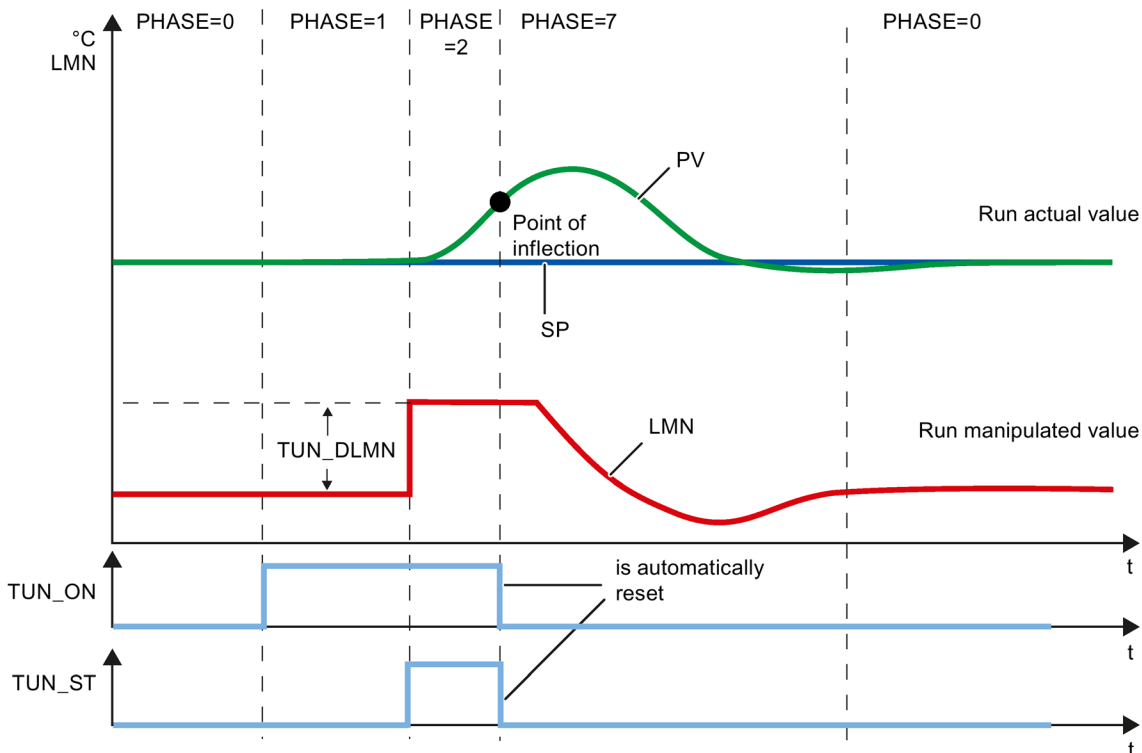
The signals of certain processes (e.g., zinc or magnesium smelters) will pass a non-linear area at the approach of the operating range (change in the state of aggregation).

By selecting a suitable setpoint step change, tuning can be limited to the linear range. When the process value has passed 75% of the setpoint step change (SP\_INT-PV0), tuning is ended.

At the same time, TUN\_DLMN should be reduced to the extent that the point of inflection is guaranteed to be found before 75% of the setpoint step change is reached.

## Fine tuning

During this tuning, the process with a constant setpoint is activated through a output value jump.



The tuning manipulated variable ( $LMN0 + TUN\_DLMN$ ) is activated by setting the start bit TUN\_ST (transition from phase 1 -> 2). When you modify the setpoint value, the new value will not take effect until the point of inflection has been reached (automatic mode will not be enabled until this point has been reached).

The user is responsible for defining the output excitation delta (TUN\_DLMN) according to the permitted process value change. The sign of TUN\_DLMN must be set depending on the intended process value change (take into account the direction in which the control is operating).

<b>NOTICE</b>
---------------

Safety off at 75% is not available when you excite the process via TUN_ST. Tuning is ended when the point of inflection is reached. However, in noisy processes the point of inflection may be significantly exceeded.
--

### Manual fine-tuning in control mode

The following measures can be employed to achieve an overshoot-free setpoint response:

- Adapting the control zone
- Optimize command action
- Attenuation of control parameters
- Modifying control parameters

### 7.3.3.4 Tuning result

The left cipher of STATUS\_H displays the tuning status

STATUS_H	Result
0	Default, i.e., new controller parameters have not (yet) been found.
10000	Suitable control parameters found.
2xxxx	Control parameters have been found via estimated values; check the control response or check the STATUS_H diagnostic message and repeat controller tuning.
3xxxx	An operator error has occurred; check the STATUS_H diagnostic message and repeat controller tuning.

The CYCLE and CYCLE\_P sampling times were already checked in phase 1.

The following controller parameters are updated on TCONT\_CP:

- P (proportional GAIN)
- I (integration time TI)
- D (derivative time TD)
- Weighting of the proportional action PFAC\_SP
- Coefficient DT1 (D\_F)
- Control zone on/off CONZ\_ON
- Control zone width CON\_ZONE

The control zone is only activated if the process type is suitable (process type I and II) and a PID controller is used (CONZ\_ON = TRUE).

Depending on PID\_ON, control is implemented either with a PI or a PID controller. The old controller parameters are saved and can be retrieved with UNDO\_PAR. A PI parameter record and a PID parameter record are saved additionally in the PI\_CON and PID\_CON structures. Using LOAD\_PID and making a suitable setting for PID\_ON, it is also possible to switch later between the tuned PI or PID parameters.

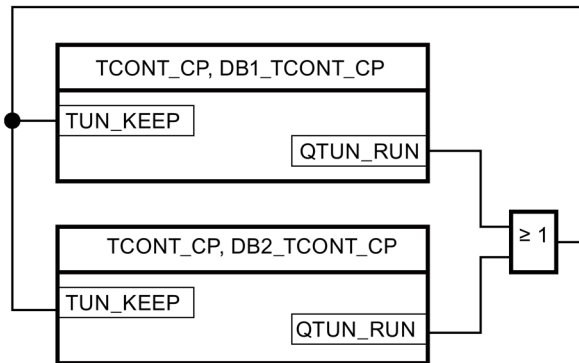
### 7.3.3.5 Parallel tuning of controller channels

#### Adjacent zones (strong heat coupling)

If two or more controllers are controlling the temperature, on a plate, for example (in other words, there are two heaters and two measured process values with strong heat coupling), proceed as follows:

1. OR the two outputs QTUN\_RUN.
2. Interconnect each TUN\_KEEP input with the output of the OR element.
3. Start both controllers by specifying a setpoint step change at the same time or by setting TUN\_ST at the same time.

The following schematic illustrates the parallel tuning of controller channels.



#### Advantage:

Both controllers output LMN0 + TUN\_DLMN until both controllers have left phase 2. This prevents the controller that completes tuning first from falsifying the tuning result of the other controller due to the change in its manipulated variable.

#### NOTICE

Reaching 75% of the setpoint step change causes an exiting of phase 2 and resetting of output QTUN\_RUN. However, automatic mode does not start until TUN\_KEEP is also 0.

#### Adjacent zones (weak heat coupling)

In general terms, tuning should be carried out to reflect the way in which the controller will operate subsequently. If zones are operated together during production such that the temperature differences between the zones remain the same, the temperature of the adjacent zones ought to be increased accordingly during tuning.

Differences in temperature at the beginning of the tuning are irrelevant since they will be compensated by the initial heating (-> initial rise = 0).

### 7.3.3.6 Fault descriptions and corrective measures

#### Compensating operator errors

Operator error	STATUS and action	Comment
TUN_ON and setpoint step change or TUN_ST are set simultaneously	Transition to phase 1; however, tuning is not started. <ul style="list-style-type: none"> <li>• SP_INT = SP<sub>old</sub> or</li> <li>• TUN_ST = FALSE</li> </ul>	The setpoint change is canceled. This prevents the controller from settling to the new setpoint value and from leaving the stationary operating point unnecessarily.
Effective TUN_DLMN < 5% (end of phase 1)	STATUS_H = 30002 <ul style="list-style-type: none"> <li>• Transition to phase 0</li> <li>• TUN_ON = FALSE</li> <li>• SP = SP<sub>old</sub></li> </ul>	Tuning is canceled. The setpoint change is canceled. This prevents the controller from settling to the new setpoint value and from leaving the stationary operating point unnecessarily.

#### Point of inflection not reached (only if excited by setpoint step change)

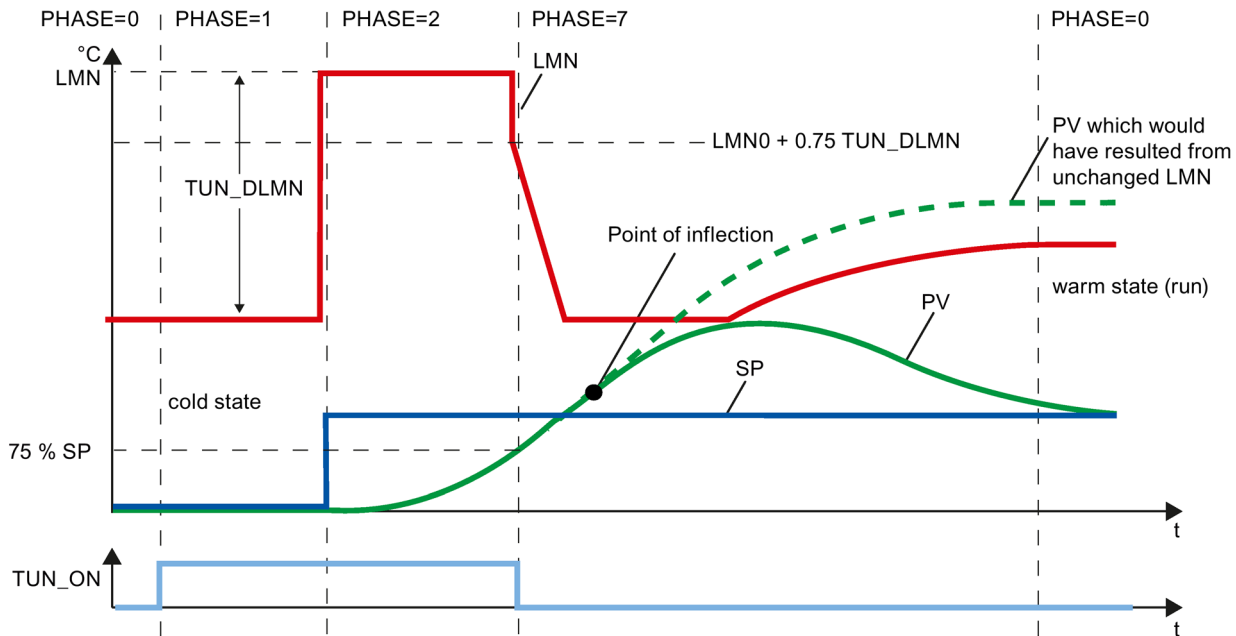
At the latest, tuning is ended when the process value has passed 75% of the setpoint step change (SP\_INT-PV0). This is signaled by "inflection point not reached" in STATUS\_H (2xx2x).

The currently valid setpoint always applies. By reducing the setpoint, it is possible to achieve an earlier end of the tuning function.

In typical temperature processes, cancelation of tuning at 75% of the setpoint step change is normally adequate to prevent overshoot. However, **caution** is advised, particularly in processes with a greater delay ( $TU/TA > 0.1$ , process type III). If manipulated variable excitation is too strong compared to the setpoint step change, the process value can overshoot heavily (up to a factor of 3).

In higher-order processes, if the point of inflection is still a long way off after reaching 75% of the setpoint step change, there will be significant overshoot. In addition, the controller parameters are too stringent. In this case, you should reduce the controller parameters or repeat the attempt.

The following schematic illustrates the overshoot of the process variable when the excitation is too strong (process type III):



In typical temperature processes, cancellation shortly before reaching the point of inflection is not critical in terms of the controller parameters.

If you repeat the attempt, reduce TUN\_DLMN or increase the setpoint step change.

Principle: The value of the manipulated variable used for tuning must be suitable for the setpoint step change.

### Error estimating the delay time or order

The delay time (STATUS\_H = 2x1xx or 2x3xx) or order (STATUS\_H = 21xxx or 22xxx) were not acquired correctly. Operation continues with an estimate that can lead to non-optimum controller parameters.

Repeat the tuning procedure and ensure that disturbances do not occur at the process value.

---

#### Note

The special case of a PT1-only process is also indicated by STATUS\_H = 2x1xx (TU <= 3\*CYCLE). In this case, it is not necessary to repeat the attempt. Reduce the controller parameters if the control oscillates.

---

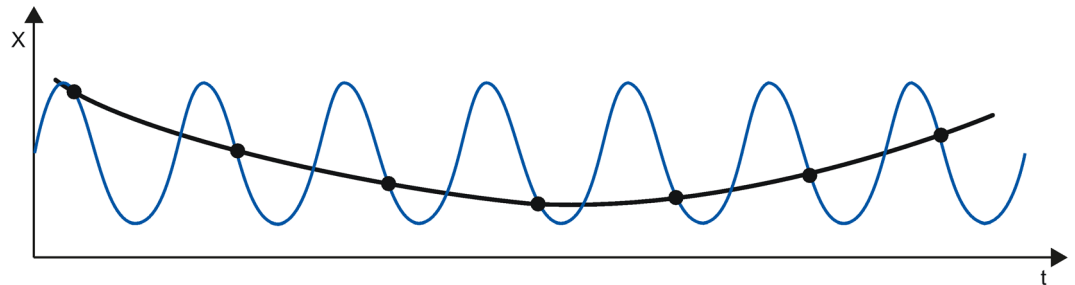


## Quality of measuring signals (measurement noise, low-frequency interference)

The results of tuning can be distorted by measurement noise or by low-frequency interference. Note the following:

- If you encounter measurement noise, set the sampling frequency higher rather than lower. During one noise period, the process value should be sampled at least twice. In pulse mode, integrated mean value filtering can be helpful. This assumes, however, that the process variable PV is transferred to the instruction in the fast pulse cycle. The degree of noise should not exceed 5% of the useful signal change.
- High-frequency interference cannot be filtered out by TCONT\_CP. This should be filtered earlier in the measuring sensor to prevent the aliasing effect.

The following schematic illustrates the aliasing effect when the sampling time is too long:



- With low-frequency interference, it is relatively easy to ensure an adequately high sampling rate. However, the TCONT\_CP must then generate a uniform measuring signal by having a large interval in the mean value filtering. Mean value filtering must extend over at least two noise periods. Internally in the block, this soon results in higher sampling times such that the accuracy of the tuning is adversely affected. Adequate accuracy is guaranteed with at least 40 noise periods up to the point of inflection.

Possible remedy when repeating the attempt:

Increase TUN\_DLMN.

## Overshoot

Overshoot can occur in the following situations:

Situation	Cause	Remedy
End of tuning	<ul style="list-style-type: none"> <li>• Excitation by a too high manipulated value change compared with the setpoint step change (see above).</li> <li>• PI controller activated by PID_ON = FALSE.</li> </ul>	<ul style="list-style-type: none"> <li>• Increase the setpoint step change or reduce the manipulated value step change.</li> <li>• If the process permits a PID controller, start tuning with PID_ON = TRUE.</li> </ul>
Tuning in phase 7	Initially, less aggressive controller parameters were determined (process type III); these can lead to an overshoot in phase 7.	-
Control mode	PI controller with PFAC_SP = 1.0 for process type I.	If the process permits a PID controller, start tuning with PID_ON = TRUE.

### 7.3.3.7 Performing pretuning

#### Requirements

- The instruction and the technology object are loaded on the CPU.

#### Procedure

To manually determine the optimum PID parameters for initial commissioning, follow these steps:

1. Click the "Start" icon.


If there is no online connection, this will be established. The current values for the setpoint, process value and output value are recorded.

2. Select "Pretuning" from the "Mode" drop-down list.

TCONT\_CP is ready for tuning.

3. In the "Output value jump" field, specify how much the output value should be increased.

4. Enter a setpoint in the "Setpoint" field. The output value jump only takes effect when another setpoint is entered.

5. Click the  "Start tuning" icon.

The pretuning starts. The status of the tuning is displayed.

### 7.3.3.8 Performing fine tuning

#### Requirements

- The instruction and the technology object are loaded on the CPU.

#### Procedure

To determine the optimal PID parameters at the operating point, follow these steps:


1. Click the "Start" icon.

If there is no online connection, this will be established. The current values for the setpoint, process value and output value are recorded.

2. Select "Fine tuning" from the "Mode" drop-down list.


TCONT\_CP is ready for tuning.

3. In the "Output value jump" field, specify how much the output value should be increased.

4. Click the  "Start tuning" icon.

Fine tuning starts. The status of the tuning is displayed.

### 7.3.3.9 Cancelling pretuning or fine tuning

To cancel pretuning or fine tuning, click on the  icon, "Stop tuning".

If the PID parameters have not yet been calculated and stored, TCONT\_CP starts in automatic mode  $LMN = LMN0 + TUN\_DLMN$ . If the controller was in manual mode before the tuning, the old manual manipulated variable is output.

If the calculated PID parameters have already been saved, TCONT\_CP starts in automatic mode and works with the previously determined PID parameters.

### 7.3.3.10 Manual fine-tuning in control mode

The following measures can be employed to achieve an overshoot-free setpoint response:

#### Adapting the control zone

During tuning, "TCONT\_CP" determines a control zone CON\_ZONE and activated if the process type is suitable (process type I and II) and a PID controller is used (CONZ\_ON = TRUE). In control mode, you can modify the control zone or switch it off completely (with CONZ\_ON = FALSE).

---

#### Note

Activating the control zone with higher-order processes (process type III) does not normally provide any benefit since the control zone is then larger than the control range that can be achieved with a 100% manipulated variable. There is also no advantage in activating the control zone for PI controllers.

Before you switch on the control zone manually, make sure that the control zone is not too narrow. If the control zone is set too narrow, oscillations occur in the manipulated variable and the process value.

---

#### Continuous attenuation of the control response with PFAC\_SP

The control response can be attenuated with the PFAC\_SP parameter. This parameter specifies the percentage of proportional component that is effective for setpoint step changes.

Regardless of the process type, PFAC\_SP is set to a default value of 0.8 by the tuning function; you can later modify this value if required. To limit overshoot during setpoint step changes (with otherwise correct controller parameters) to approximately 2%, the following values are adequate for PFAC\_SP:

	Process type I	Process type II	Process type III
	Typical temperature process	Intermediate range	Higher-order temperature process
PI	0.8	0.82	0.8
PID	0.6	0.75	0.96

Adjust the default factor (0.8) in the following situations, in particular:

- Process type I with PID (0.8 →0.6): Setpoint step changes within the control zone still lead to approximately 18% overshoot with PFAC\_SP = 0.8.
- Process type III with PID (0.8 →0.96): Setpoint step changes with PFAC\_SP = 0.8 are attenuated too strongly. This leads to a significantly slower response time.

### Attenuation of control parameters

When a closed-loop control circuit oscillates or if overshoot occurs after setpoint step changes, you can reduce the controller GAIN (e.g., to 80% of the original value) and increase integral time TI (e.g., to 150% of the original value). If the analog output value of the continuous controller is converted to binary actuating signals by a pulse shaper, quantization noise may cause minor permanent oscillation. You can eliminate this by increasing the controller deadband DEADB\_W.

### Modifying control parameters

Proceed as follows to modify control parameters:

1. Save the current parameters with SAVE\_PAR.
2. Modify the parameters.
3. Test the control response.

If the new parameter settings are worse than the old ones, retrieve the old parameters with UNDO\_PAR.



### 7.3.3.11 Performing fine tuning manually

#### Requirements

- The instruction and the technology object have been loaded to the CPU.

#### Procedure

To manually determine the optimal PID parameters, proceed as follows:

1. Click the "Start" icon.  
If there is no online connection, this will be established. The current values for the setpoint, process value and output value are recorded.
2. Select "Manual" from the "Mode" drop-down list.
3. Enter the new PID parameters.
4. Click on the icon  "Send parameter to CPU" in the "Tuning" group.
5. Select the "Change setpoint" check box in the "Current values" group.
6. Enter a new setpoint and click in the "Current Values" group on the icon .
7. Clear the "Manual mode" check box.  
The controller works with the new PID parameters and controls the new setpoint.
8. Check the quality of the PID parameter to check the curve points.
9. Repeat steps 3 to 8 until you are satisfied with the controller results.

## 7.4 TCONT\_S

### 7.4.1 Technology object TCONT\_S

The technology object TCONT\_S provides a step controller for actuators with integrating behavior and is used to control technical temperature processes with binary output value output signals. The technology object corresponds to the instance data block of the TCONT\_S instruction. The operating principle is based on the PI control algorithm of the sampling controller. The step controller operates without a position feedback signal. Both manual and automatic mode are possible.

S7-1500

All parameters and tags of the technology object are retentive and can only be changed during download to the device if you completely download TCONT\_S.

#### See also

Overview of software controller (Page 38)

Add technology objects (Page 40)

Configure technology objects (Page 45)

TCONT\_S (Page 508)

Downloading technology objects to device (Page 72)

### 7.4.2 Configure controller difference TCONT\_S

#### Use process value periphery

To use the input parameter PV\_PER, proceed as follows:

1. Select the entry "Periphery" from the "Source" list.
2. Select the "sensor type".  
Depending on the sensor type, the process value is scaled according to different formulas.
  - Standard  
Thermoelements; PT100/Ni100  
 $PV = 0.1 \times PV\_PER \times PV\_FAC + PV\_OFFS$
  - Cooling;  
PT100/Ni100  
 $PV = 0.01 \times PV\_PER \times PV\_FAC + PV\_OFFS$
  - Current/voltage  
 $PV = 100/27648 \times PV\_PER \times PV\_FAC + PV\_OFFS$
3. Enter the factor and offset for the scaling of the process value periphery.

### Use internal process values

To use the input parameter PV\_IN, proceed as follows:

1. Select the entry "Internal" from the "Source" list.

### Control deviation

Set a dead zone range under the following requirement:

- The process value signal is noisy.
- The controller gain is high.
- The derivative action is activated.

The noise component of the process value causes strong deviations of the output value in this case. The dead zone suppresses the noise component in the steady controller state.

The dead zone range specifies the size of the dead zone. With a dead zone range of 0.0, the dead zone is turned off.

### See also

Mode of operation TCONT\_S (Page 509)

### 7.4.3 Configure controller algorithm TCONT\_S

#### General

1. Enter the "Sampling time PID algorithm".  
A controller sampling time should not exceed 10 % of the determined integral action time of the controller (TI).
2. If the controller structure contains a proportional action, enter the "proportional gain".  
A negative proportional gain inverts the rule meaning.

#### Proportional action

For changes of the setpoint, it may lead to overshooting of the proportional action. Through the weighting of the proportional action, you can select how strongly the proportional action should react when setpoint changes are made. The weakening of the proportional action is reached through a compensation of the integral action.

1. To weaken the proportional action for setpoint changes, enter a "Proportional action weighting".
  - 1.0: Proportional action for setpoint change is fully effective
  - 0.0: Proportional action for setpoint change is not effective

#### Integral action

1. If the controller structure contains an integral action, enter the "integral action time".  
With an integral action time of 0.0, the integral action is switched off.

#### See also

Mode of operation TCONT\_S (Page 509)

### 7.4.4 Configure manipulated value TCONT\_S

#### Pulse generator

1. Enter the minimum impulse duration and minimum pause duration.  
The values must be greater than or equal to the cycle time for the input parameter CYCLE. The frequency of operation is reduced through this.
2. Enter the motor setting time.  
The value must be greater than or equal to the cycle time of the input parameter CYCLE.

#### See also

Mode of operation TCONT\_S (Page 509)



## 7.4.5 Commissioning TCONT\_S

### Requirements



- The instruction and the technology object have been loaded to the CPU.

### Procedure

To manually determine the optimal PID parameters, proceed as follows:

1. Click the "Start" icon.

If there is no online connection, this will be established. The current values for the setpoint, process value and output value are recorded.

2. Enter new PID parameters in the "P", "I" and weighting proportional action fields.
3. Click on the icon  "Send parameter to CPU" in the "Tuning" group.
4. Select the "Change setpoint" check box in the "Current values" group.
5. Enter a new setpoint and click in the "Current Values" group on the icon .
6. Clear the "Manual mode" check box.

The controller works with the new parameters and controls the new setpoint.

7. Check the quality of the PID parameter to check the curve points.
8. Repeat steps 2 to 6 until you are satisfied with the controller results.

# Instructions

## 8.1 PID\_Compact

### 8.1.1 New features of PID\_Compact

#### PID\_Compact V2.3

- **Response of the output value when switching from "Inactive" operating mode to "Automatic mode"**

The new option `IntegralResetMode = 4` was added and defined as default. With `IntegralResetMode = 4`, the integral action is automatically pre-assigned when switching from "Inactive" operating mode to "Automatic mode" so that a control deviation results in a jump of the output value with identical sign.

- **Initialization of the integral action in automatic mode**

The integral action can be initialized in automatic mode with the tags `OverwriteInitialOutputValue` and `PIDCtrl.PIDInit`. This simplifies the use of `PID_Compact` for override controls.

#### PID\_Compact V2.2

- **Use with S7-1200**

As of `PID_Compact V2.2`, the instruction with V2 functionality can also be used on S7-1200 with firmware version 4.0 or higher.

## PID\_Compact V2.0

- **Reaction to error**

The reaction to error has been completely overhauled. PID\_Compact now reacts in a more fault-tolerant manner in the default setting. This reaction is set when copying PID\_Compact V1.X from an S7-1200 CPU to an S7-1500 CPU.

<b>NOTICE</b>
<b>Your system may be damaged.</b>
If you use the default setting, PID_Compact remains in automatic mode when the process value limits are exceeded. This may damage your system.
It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

The Error parameter indicates if an error is pending. When the error is no longer pending, Error = FALSE. The ErrorBits parameter shows which errors have occurred. Use ErrorAck to acknowledge the errors and warnings without restarting the controller or clearing the integral action. Switching operating modes no longer clears errors that are no longer pending.

You can configure the reaction to error with SetSubstituteOutput and ActivateRecoverMode.

- **Substitute output value**

You can configure a substitute output value that is to be output if an error occurs.

- **Switching the operating mode**

You specify the operating mode at the Mode in/out parameter and use a rising edge at ModeActivate to start the operating mode. The sRet.i\_Mode tag has been omitted.

- **Multi-instance capability**

You can call up PID\_Compact as multi-instance DB. No technology object is created in this case and no parameter assignment interface or commissioning interface is available. You must assign parameters for PID\_Compact directly in the multi-instance DB and commission it via a watch table.

- **Startup characteristics**

The operating mode specified at the Mode parameter is also started on a falling edge at Reset and during a CPU cold restart, if RunModeByStartup = TRUE.

- **ENO characteristics**

ENO is set depending on the operating mode.

If State = 0, then ENO = FALSE.

If State ≠ 0, then ENO = TRUE.

- **Setpoint value specification during tuning**

You configure the permitted fluctuation of the setpoint during tuning at the CancelTuningLevel tag.

- **Value range for output value limits**  
The value 0.0 no longer has to fall within the output value limits.
- **Pre-assigning the integral action**  
Using the tags IntegralResetMode and OverwriteInitialOutputValue, you can determine the pre-assignment of the integral action when switching from "Inactive" operating mode to "Automatic mode".
- **Switching a disturbance variable on**  
You can switch a disturbance variable on at the Disturbance parameter.
- **Default value of PID parameters**  
The following default settings have been changed:
  - Proportional action weighting (PWeighting) from 0.0 to 1.0
  - Derivative action weighting (DWeighting) from 0.0 to 1.0
  - Coefficient for derivative delay (TdFiltRatio) from 0.0 to 0.2
- **Renaming tags**  
The static tags have been given new names that are compatible with PID\_3Step.

### PID\_Compact V1.2

- **Manual mode on CPU startup**  
If ManualEnable = TRUE when the CPU starts, PID\_Compact starts in manual mode. A rising edge at ManualEnable is not necessary.
- **Pretuning**  
If the CPU is switched off during pretuning, pretuning starts again when the CPU is switched back on.

### PID\_Compact V1.1

- **Manual mode on CPU startup**  
When the CPU starts up, PID\_Compact only switches to manual mode with a rising edge at ManualEnable. Without rising edge, PID\_Compact starts in the last operating mode in which ManualEnable was FALSE.
- **Reaction to reset**  
A rising edge at Reset resets the errors and warnings and clears the integral action. A falling edge at Reset triggers a switchover to the most recently active operating mode.
- **Default of process value high limit**  
The default value of r\_Pv\_Hlm has been changed to 120.0.

- **Monitoring the sampling time**
  - An error is no longer output when the current sampling time is  $\geq 1.5 \times$  current mean value or when the current sampling time is  $\leq 0.5 \times$  current mean value. The sampling time may deviate much more in automatic mode.
  - PID\_Compact is compatible with FW, V2.0 or higher.
- **Access to tags**

The following tags can now be used in the user program.

  - i\_Event\_SUT
  - i\_Event\_TIR
  - r\_Ctrl\_loutv
- **Troubleshooting**

PID\_Compact now outputs the correct pulses when the shortest ON time is not equal to the shortest OFF time.

## 8.1.2 Compatibility with CPU and FW

The following table shows which version of PID\_Compact can be used on which CPU.

CPU	FW	PID_Compact
S7-1200	V4.2 or higher	V2.3 V2.2 V1.2
	V4.0 to V4.1	V2.2 V1.2
	V3.x	V1.2 V1.1
	V2.x	V1.2 V1.1
	V1.x	V1.0
S7-1500	V2.0 or higher	V2.3 V2.2 V2.1 V2.0
	V1.5 to V1.8	V2.2 V2.1 V2.0
	V1.1	V2.1 V2.0
	V1.0	V2.0

### 8.1.3 PID\_Compact V2

#### 8.1.3.1 Description of PID\_Compact V2

##### Description

The PID\_Compact instruction provides a PID controller with integrated tuning for actuators with proportional action.

The following operating modes are possible:

- Inactive
- Pretuning
- Fine tuning
- Automatic mode
- Manual mode
- Substitute output value with error monitoring

For a more detailed description of the operating modes, see the State parameter.

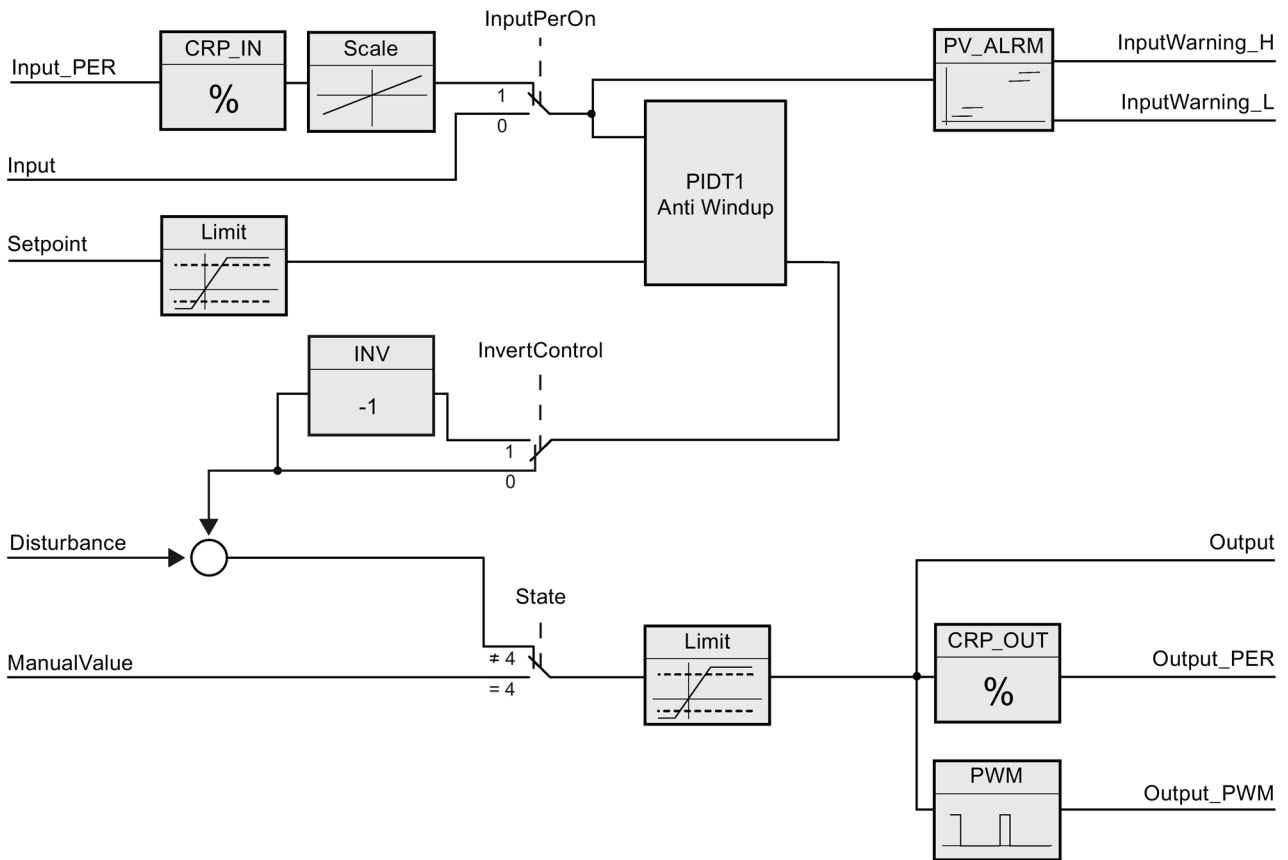
##### PID algorithm

PID\_Compact is a PIDT1 controller with anti-windup and weighting of the proportional and derivative actions. The PID algorithm operates according to the following equation:

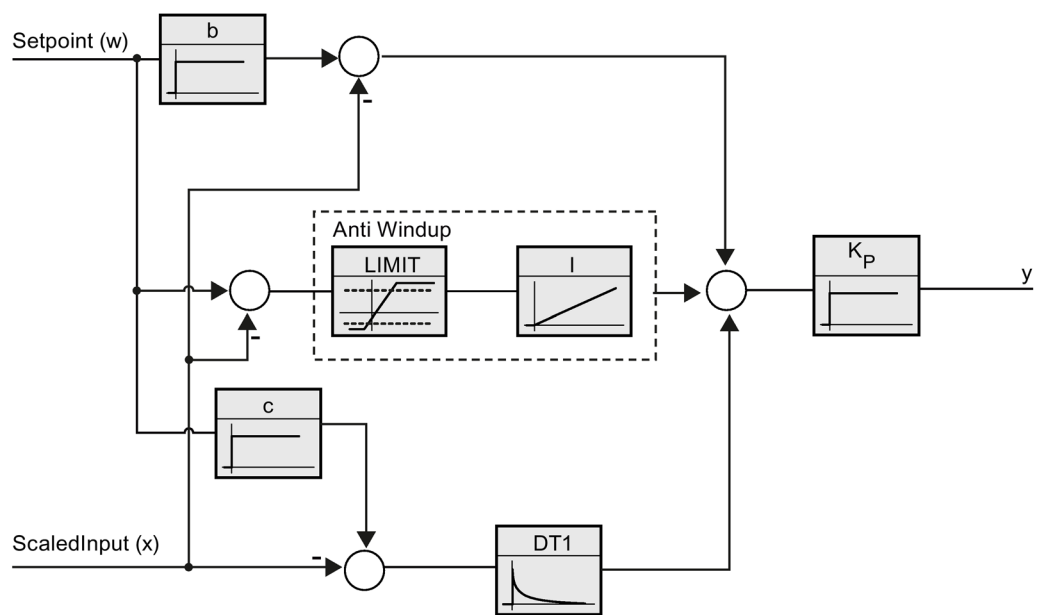
$$y = K_p \left[ (b \cdot w - x) + \frac{1}{T_i \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

Symbol	Description
y	Output value of the PID algorithm
K <sub>p</sub>	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T <sub>i</sub>	Integral action time
T <sub>D</sub>	Derivative action time
a	Derivative delay coefficient (derivative delay T1 = a × T <sub>D</sub> )
c	Derivative action weighting

Block diagram of PID\_Compact



Block diagram of PIDT1 with anti-windup



**Call**

PID\_Compact is called in the constant time scale of a cycle interrupt OB.

If you call PID\_Compact as a multi-instance DB, no technology object is created. No parameter assignment interface or commissioning interface is available. You must assign parameters for PID\_Compact directly in the multi-instance DB and commission it via a watch table.

**Download to device**

The actual values of retentive variables are only updated when you download PID\_Compact completely.

Downloading technology objects to device (Page 72)

**Startup**

When the CPU starts up, PID\_Compact starts in the operating mode that is saved in the Mode in/out parameter. To switch to "Inactive" operating mode during startup, set RunModeByStartup = FALSE.

**Reaction to error**

In automatic mode and during commissioning, the reaction to error depends on the SetSubstituteOutput and ActivateRecoverMode variables. In manual mode, the reaction is independent of SetSubstituteOutput and ActivateRecoverMode. If ActivateRecoverMode = TRUE, the reaction additionally depends on the error that occurred.

SetSubstitute-Output	ActivateRecoverMode	Configuration editor > output value > Set Output to	Reaction
Not relevant	FALSE	Zero (inactive)	Switch to "Inactive" mode (State = 0) The value 0.0 0 is transferred to the actuator.
FALSE	TRUE	Current output value while error is pending	Switch to "Substitute output value with error monitoring" mode (State = 5) The current output value is transferred to the actuator while the error is pending.
TRUE	TRUE	Substitute output value while error is pending	Switch to "Substitute output value with error monitoring" mode (State = 5) The value at SubstituteOutput is transferred to the actuator while the error is pending.

In manual mode, PID\_Compact uses ManualValue as output value, unless ManualValue is invalid. If ManualValue is invalid, SubstituteOutput is used. If ManualValue and SubstituteOutput are invalid, Config.OutputLowerLimit is used.

The Error parameter indicates if an error is pending. When the error is no longer pending, Error = FALSE. The ErrorBits parameter shows which errors have occurred. ErrorBits is reset by a rising edge at Reset or ErrorAck.



### 8.1.3.2 Mode of operation of PID\_Compact V2

#### Monitoring process value limits

You specify the high limit and low limit of the process value in the Config.InputUpperLimit and Config.InputLowerLimit tags. If the process value is outside these limits, an error occurs (ErrorBits = 0001h).

You specify a high and low warning limit of the process value in the Config.InputUpperWarning and Config.InputLowerWarning tags. If the process value is outside these warning limits, a warning occurs (Warning = 0040h), and the InputWarning\_H or InputWarning\_L output parameter changes to TRUE.

#### Limiting the setpoint

You specify a high limit and low limit of the setpoint in the Config.SetpointUpperLimit and Config.SetpointLowerLimit tags. PID\_Compact automatically limits the setpoint to the process value limits. You can limit the setpoint to a smaller range. PID\_Compact checks whether this range falls within the process value limits. If the setpoint is outside these limits, the high or low limit is used as the setpoint, and output parameter SetpointLimit\_H or SetpointLimit\_L is set to TRUE.

The setpoint is limited in all operating modes.

#### Limiting the output value

You specify a high limit and low limit of the output value in the Config.OutputUpperLimit and Config.OutputLowerLimit tags. Output, ManualValue and SubstituteOutput are limited to these values. The output value limits must match the control logic.

The valid output value limit values depend on the Output used.

Output	-100.0 to 100.0%
Output_PER	-100.0 to 100.0%
Output_PWM	0.0 to 100.0%

Rule:

OutputUpperLimit > OutputLowerLimit

---

#### Note

##### Use with two or more actuators

PID\_Compact is not suitable for use with two or more actuators (for example, in heating/cooling applications), because different actuators need different PID parameters to achieve a good control response. Use PID\_Temp for applications with two actuators acting in opposite directions.

---

### Substitute output value

In the event of an error, PID\_Compact can output a substitute output value that you define at the SubstituteOutput tag. The substitute output value must be within the output value limits.

### Monitoring signal validity

The values of the following parameters are monitored for validity when used:

- Setpoint
- Input
- Input\_PER
- Disturbance
- ManualValue
- SubstituteOutput
- Output
- Output\_PER
- Output\_PWM

### Monitoring of the sampling time PID\_Compact

Ideally, the sampling time is equivalent to the cycle time of the calling OB. The PID\_Compact instruction measures the time interval between two calls. This is the current sampling time. On every switchover of operating mode and during the initial startup, the mean value is formed from the first 10 sampling times. Too great a difference between the current sampling time and this mean value triggers an error (Error = 0800h).

The error occurs during tuning if:

- New mean value  $\geq 1.1 \times$  old mean value
- New mean value  $\leq 0.9 \times$  old mean value

The error occurs in automatic mode if:

- New mean value  $\geq 1.5 \times$  old mean value
- New mean value  $\leq 0.5 \times$  old mean value

If you deactivate the sampling time monitoring (CycleTime.EnMonitoring = FALSE), you can also call PID\_Compact in OB1. You must then accept a lower control quality due to the deviating sampling time.

### **Sampling time of the PID algorithm**

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the cycle time. All other functions of PID\_Compact are executed at every call.

If you use Output\_PWM, the accuracy of the output signal is determined by the ratio of the PID algorithm sampling time to the cycle time of the OB. The cycle time should be at least 10 times the PID algorithm sampling time.

### **Control logic**

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic. For cooling and discharge control systems, it may be necessary to invert the control logic. PID\_Compact does not work with negative proportional gain. If InvertControl = TRUE, an increasing control deviation causes a reduction in the output value. The control logic is also taken into account during pretuning and fine tuning.

8.1.3.3 Input parameters of PID\_Compact V2

Table 8- 1

Parameter	Data type	Default	Description
Setpoint	REAL	0.0	Setpoint of the PID controller in automatic mode
Input	REAL	0.0	A tag of the user program is used as source for the process value. If you are using parameter Input, then Config.InputPerOn = FALSE must be set.
Input_PER	INT	0	An analog input is used as the source of the process value. If you are using parameter Input_PER, then Config.InputPerOn = TRUE must be set.
Disturbance	REAL	0.0	Disturbance variable or precontrol value
ManualEnable	BOOL	FALSE	<ul style="list-style-type: none"> <li>A FALSE -&gt; TRUE edge activates "manual mode", while State = 4, Mode remains unchanged.</li> </ul> As long as ManualEnable = TRUE, you cannot change the operating mode via a rising edge at ModeActivate or use the commissioning dialog. <ul style="list-style-type: none"> <li>A TRUE -&gt; FALSE edge activates the operating mode that is specified by Mode.</li> </ul> We recommend that you change the operating mode using ModeActivate only.
ManualValue	REAL	0.0	Manual value This value is used as the output value in manual mode. Values from Config.OutputLowerLimit to Config.OutputUpperLimit are permitted.
ErrorAck	BOOL	FALSE	<ul style="list-style-type: none"> <li>FALSE -&gt; TRUE edge</li> </ul> ErrorBits and Warning are reset.

Parameter	Data type	Default	Description
Reset	BOOL	FALSE	<p>Restarts the controller.</p> <ul style="list-style-type: none"> <li>• FALSE -&gt; TRUE edge <ul style="list-style-type: none"> <li>- Switch to "Inactive" mode</li> <li>- ErrorBits and Warnings are reset.</li> </ul> </li> <li>• As long as Reset = TRUE, <ul style="list-style-type: none"> <li>- PID_Compact remains in "Inactive" mode (State = 0).</li> <li>- You cannot change the operating mode with Mode and ModeActivate or ManualEnable.</li> <li>- You cannot use the commissioning dialog.</li> </ul> </li> <li>• TRUE -&gt; FALSE edge <ul style="list-style-type: none"> <li>- If ManualEnable = FALSE, PID_Compact switches to the operating mode that is saved in Mode.</li> <li>- If Mode = 3, the integral action is treated as configured with the tag IntegralResetMode.</li> </ul> </li> </ul>
ModeActivate	BOOL	FALSE	<ul style="list-style-type: none"> <li>• FALSE -&gt; TRUE edge</li> </ul> <p>PID_Compact switches to the operating mode that is saved in the Mode parameter.</p>

8.1.3.4 Output parameters of PID\_Compact V2

Table 8- 2

Parameter	Data type	Default	Description
ScaledInput	REAL	0.0	Scaled process value
The "Output", "Output_PER", and "Output_PWM" outputs can be used concurrently.			
Output	REAL	0.0	Output value in REAL format
Output_PER	INT	0	Analog output value
Output_PWM	BOOL	FALSE	Pulse-width-modulated output value The output value is formed by by variable On and Off times.
SetpointLimit_H	BOOL	FALSE	If SetpointLimit_H = TRUE, the absolute setpoint high limit is reached (Setpoint ≥ Config.SetpointUpperLimit). The setpoint is limited to Config.SetpointUpperLimit .
SetpointLimit_L	BOOL	FALSE	If SetpointLimit_L = TRUE, the absolute setpoint low limit has been reached (Setpoint ≤ Config.SetpointLowerLimit). The setpoint is limited to Config.SetpointLowerLimit .
InputWarning_H	BOOL	FALSE	If InputWarning_H = TRUE, the process value has reached or exceeded the warning high limit.
InputWarning_L	BOOL	FALSE	If InputWarning_L = TRUE, the process value has reached or fallen below the warning low limit.
State	INT	0	The State parameter (Page 274) shows the current operating mode of the PID controller. You can change the operating mode using the input parameter Mode and a rising edge at ModeActivate. <ul style="list-style-type: none"> <li>• State = 0: Inactive</li> <li>• State = 1: Pretuning</li> <li>• State = 2: Fine tuning</li> <li>• State = 3: Automatic mode</li> <li>• State = 4: Manual mode</li> <li>• State = 5: Substitute output value with error monitoring</li> </ul>
Error	BOOL	FALSE	If Error = TRUE, at least one error message is pending in this cycle.
ErrorBits	DWORD	DW#16#0	The ErrorBits parameter (Page 278) shows which error messages are pending. ErrorBits is retentive and is reset upon a rising edge at Reset or ErrorAck.

### 8.1.3.5 In/out parameters of PID\_Compact V2

Table 8- 3

Parameter	Data type	Default	Description
Mode	INT	4	<p>At Mode, specify the operating mode to which PID_Compact is to switch. Options are:</p> <ul style="list-style-type: none"> <li>• Mode = 0: Inactive</li> <li>• Mode = 1: Pretuning</li> <li>• Mode = 2: Fine tuning</li> <li>• Mode = 3: Automatic mode</li> <li>• Mode = 4: Manual mode</li> </ul> <p>The operating mode is activated by:</p> <ul style="list-style-type: none"> <li>• Rising edge at ModeActivate</li> <li>• Falling edge at Reset</li> <li>• Falling edge at ManualEnable</li> <li>• Cold restart of CPU if RunModeBy-Startup = TRUE</li> </ul> <p>Mode is retentive.</p> <p>A detailed description of the operating modes can be found in Parameters State and Mode V2 (Page 274).</p>

#### See also

Parameters State and Mode V2 (Page 274)

8.1.3.6 Static tags of PID\_Compact V2

You must not change tags that are not listed. These are used for internal purposes only.

Tag	Data type	Default	Description
IntegralResetMode	INT	Up to V2.2: 1, V2.3 or higher: 4	The IntegralResetMode V2 tag (Page 283) determines how the integral actionPIDCtrl.IntegralSum is pre-assigned when switching from "Inactive" operating mode to "Automatic mode". This setting only works for one cycle. Options are: <ul style="list-style-type: none"> <li>• IntegralResetMode = 0: Smooth</li> <li>• IntegralResetMode = 1: Delete</li> <li>• IntegralResetMode = 2: Hold</li> <li>• IntegralResetMode = 3: Pre-assign</li> <li>• IntegralResetMode = 4: Like setpoint change (only for PID_Compact with version ≥ 2.3)</li> </ul>
OverwriteInitialOutputValue	REAL	0.0	If one of the following conditions is met, the integral action PIDCtrl.IntegralSum is pre-assigned automatically as if Output = OverwriteInitialOutputValue in the previous cycle: <ul style="list-style-type: none"> <li>• IntegralResetMode = 3 when switching from "Inactive" operating mode to "Automatic mode".</li> <li>• TRUE -&gt; FALSE edge at parameter Reset and parameter Mode = 3</li> <li>• PIDCtrl.PIDInit = TRUE in "Automatic mode" (available as of PID_Compact version 2.3)</li> </ul>
RunModeByStartup	BOOL	TRUE	Activate operating mode at Mode parameter after CPU restart If RunModeByStartup = TRUE, PID_Compact starts in the operating mode saved in the Mode parameter after CPU startup. If RunModeByStartup = FALSE, PID_Compact remains in "Inactive" mode after CPU startup.
LoadBackUp	BOOL	FALSE	If LoadBackUp = TRUE, the last set of PID parameters is reloaded. The set was saved prior to the last tuning. LoadBackUp is automatically set back to FALSE.
PhysicalUnit	INT	0	Unit of measurement of the process value and setpoint, e.g., °C, or °F.
PhysicalQuantity	INT	0	Physical quantity of the process value and setpoint, e.g., temperature.
ActivateRecoverMode	BOOL	TRUE	The Tag ActivateRecoverMode V2 (Page 280) determines the reaction to error.
Warning	DWORD	0	Tag Warning V2 (Page 282) shows the warnings since Reset = TRUE or ErrorAck =TRUE. Warning is retentive.
Progress	REAL	0.0	Progress of tuning as a percentage (0.0 - 100.0)



Tag	Data type	Default	Description
CurrentSetpoint	REAL	0.0	CurrentSetpoint always displays the current setpoint. This value is frozen during tuning.
CancelTuningLevel	REAL	10.0	Permissible fluctuation of setpoint during tuning. Tuning is not canceled until: <ul style="list-style-type: none"> <li>Setpoint &gt; CurrentSetpoint + CancelTuningLevel</li> <li>or</li> <li>Setpoint &lt; CurrentSetpoint - CancelTuningLevel</li> </ul>
SubstituteOutput	REAL	0.0	Substitute output value When the following conditions are met, the substitute output value is used: <ul style="list-style-type: none"> <li>An error has occurred in automatic mode.</li> <li>SetSubstituteOutput = TRUE</li> <li>ActivateRecoverMode = TRUE</li> </ul>
SetSubstituteOutput	BOOL	TRUE	If SetSubstituteOutput = TRUE and ActivateRecoverMode = TRUE, the substitute output value configured is output as long as an error is pending. If SetSubstituteOutput = FALSE and ActivateRecoverMode = TRUE, the actuator remains at the current output value as long as an error is pending. If ActivateRecoverMode = FALSE, SetSubstituteOutput is not effective. If SubstituteOutput is invalid (ErrorBits = 20000h), the substitute output value cannot be output.
Config.InputPerOn	BOOL	TRUE	If InputPerOn = TRUE, the Input_PER parameter is used. If InputPerOn = FALSE, the Input parameter is used.
Config.InvertControl	BOOL	FALSE	Invert control logic If InvertControl = TRUE, an increasing control deviation causes a reduction in the output value.
Config.InputUpperLimit	REAL	120.0	High limit of the process value Input and Input_PER are monitored to ensure adherence to this limit. At the I/O input, the process value can be a maximum of 18% higher than the standard range (overrange). This pre-assignment ensures that an error is no longer signaled due to a violation of the "Process value high limit". Only a wire-break and a short-circuit are recognized and PID_Compact reacts according to the configured reaction to error. InputUpperLimit > InputLowerLimit
Config.InputLowerLimit	REAL	0.0	Low limit of the process value Input and Input_PER are monitored to ensure adherence to this limit. InputLowerLimit < InputUpperLimit

8.1 PID\_Compact

Tag	Data type	Default	Description
Config.InputUpperWarning	REAL	3.402822e+38	Warning high limit of the process value If you set InputUpperWarning outside the process value limits, the configured absolute process value high limit is used as the warning high limit. If you configure InputUpperWarning within the process value limits, this value is used as the warning high limit. InputUpperWarning > InputLowerWarning InputUpperWarning ≤ InputUpperLimit
Config.InputLowerWarning	REAL	-3.402822e+38	Warning low limit of the process value If you set InputLowerWarning outside the process value limits, the configured absolute process value low limit is used as the warning low limit. If you configure InputLowerWarning within the process value limits, this value is used as the warning low limit. InputLowerWarning < InputUpperWarning InputLowerWarning ≥ InputLowerLimit
Config.OutputUpperLimit	REAL	100.0	High limit of output value For details, see OutputLowerLimit OutputUpperLimit > OutputLowerLimit
Config.OutputLowerLimit	REAL	0.0	Low limit of output value For Output and Output_PER, the range of values from -100.0 to +100.0, including zero, is valid. At -100.0, Output_PER = -27648; at +100.0, Output_PER = 27648. For Output_PWM, the value range 0.0 to +100.0 applies. The output value limits must match the control logic. OutputLowerLimit < OutputUpperLimit
Config.SetpointUpperLimit	REAL	3.402822e+38	High limit of setpoint If you configure SetpointUpperLimit outside the process value limits, the configured process value absolute high limit is used as the setpoint high limit. If you configure SetpointUpperLimit within the process value limits, this value is used as the setpoint high limit.
Config.SetpointLowerLimit	REAL	-3.402822e+38	Low limit of the setpoint If you set SetpointLowerLimit outside the process value limits, the configured process value absolute low limit is used as the setpoint low limit. If you set SetpointLowerLimit within the process value limits, this value is used as the setpoint low limit.
Config.MinimumOnTime	REAL	0.0	The minimum ON time of the pulse width modulation in seconds is rounded to MinimumOnTime = n×CycleTime.Value

Tag	Data type	Default	Description
Config.MinimumOffTime	REAL	0.0	The minimum OFF time of the pulse width modulation in seconds is rounded to MinimumOffTime = n×CycleTime.Value
Config.InputScaling.UpperPointIn	REAL	27648.0	Scaling Input_PER high Input_PER is converted to percent based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn.
Config.InputScaling.LowerPointIn	REAL	0.0	Scaling Input_PER low Input_PER is converted to percent based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn.
Config.InputScaling.UpperPointOut	REAL	100.0	Scaled high process value Input_PER is converted to percent based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn.
Config.InputScaling.LowerPointOut	REAL	0.0	Scaled low process value Input_PER is converted to percent based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn.
CycleTime.StartEstimation	BOOL	TRUE	If CycleTime.StartEstimation = TRUE, the automatic determination of the cycle time is started. CycleTime.StartEstimation = FALSE once measurement is complete.
CycleTime.EnEstimation	BOOL	TRUE	If CycleTime.EnEstimation = TRUE, the PID_Compact sampling time is calculated. If CycleTime.EnEstimation = FALSE, the PID_Compact sampling time is not calculated and you need to correct the configuration of CycleTime.Value manually.
CycleTime.EnMonitoring	BOOL	TRUE	If CycleTime.EnMonitoring = FALSE, the PID_Compact sampling time is not monitored. If it is not possible to execute PID_Compact within the sampling time, no error (ErrorBits=0800h) is output and PID_Compact does not switch to "Inactive" mode.
CycleTime.Value	REAL	0.1	PID_Compact sampling time in seconds CycleTime.Value is determined automatically and is usually equivalent to the cycle time of the calling OB.
CtrlParamsBackUp.Gain	REAL	1.0	Saved proportional gain You can reload values from the CtrlParamsBackUp structure with LoadBackUp = TRUE.
CtrlParamsBackUp.Ti	REAL	20.0	Saved integral action time [s]
CtrlParamsBackUp.Td	REAL	0.0	Saved derivative action time [s]
CtrlParamsBackUp.TdFiltRatio	REAL	0.2	Saved derivative delay coefficient
CtrlParamsBackUp.PWeighting	REAL	1.0	Saved proportional action weighting factor
CtrlParamsBackUp.DWeighting	REAL	1.0	Saved derivative action weighting factor
CtrlParamsBackUp.Cycle	REAL	1.0	Saved sampling time of PID algorithm

8.1 PID\_Compact

Tag	Data type	Default	Description
PIDSelfTune.SUT.CalculateParams	BOOL	FALSE	The properties of the controlled system are saved during tuning. If SUT.CalculateParams = TRUE, the parameters for pretuning are recalculated according to these properties. This enables you to change the parameter calculation method without having to repeat controller tuning. SUT.CalculateParams is set to FALSE after the calculation.
PIDSelfTune.SUT.TuneRule	INT	0	Methods used to calculate parameters during pretuning: <ul style="list-style-type: none"> <li>• SUT.TuneRule = 0: PID according to Chien, Hrones and Reswick</li> <li>• SUT.TuneRule = 1: PI according to Chien, Hrones and Reswick</li> </ul>
PIDSelfTune.SUT.State	INT	0	The SUT.State tag indicates the current phase of pretuning: <ul style="list-style-type: none"> <li>• State = 0: Initialize pretuning</li> <li>• State = 100: Calculate the standard deviation</li> <li>• State = 200: Find the point of inflection</li> <li>• State = 9900: Pretuning successful</li> <li>• State = 1: Pretuning not successful</li> </ul>
PIDSelfTune.TIR.RunIn	BOOL	FALSE	With the RunIn tag, you can specify that fine tuning can also be performed without pretuning. <ul style="list-style-type: none"> <li>• RunIn = FALSE Pretuning is started when fine tuning is started from inactive or manual mode. If the requirements for pretuning are not met, PID_Compact reacts as if RunIn = TRUE. If fine tuning is started from automatic mode, the system uses the existing PID parameters to control to the setpoint. Only then will fine tuning start. If pretuning is not possible, PID_Compact switches to the mode from which tuning was started.</li> <li>• RunIn = TRUE The pretuning is skipped. PID_Compact tries to reach the setpoint with minimum or maximum output value. This can produce increased overshoot. Fine tuning then starts automatically. RunIn is set to FALSE after fine tuning.</li> </ul>

Tag	Data type	Default	Description
PIDSelfTune.TIR.CalculateParams	BOOL	FALSE	The properties of the controlled system are saved during tuning. If TIR.CalculateParams = TRUE, the parameters for fine tuning are recalculated according to these properties. This enables you to change the parameter calculation method without having to repeat controller tuning. TIR.CalculateParams is set to FALSE after the calculation.
PIDSelfTune.TIR.TuneRule	INT	0	Methods used to calculate parameters during fine tuning: <ul style="list-style-type: none"> <li>• TIR.TuneRule = 0: PID automatic</li> <li>• TIR.TuneRule = 1: PID fast</li> <li>• TIR.TuneRule = 2: PID slow</li> <li>• TIR.TuneRule = 3: Ziegler-Nichols PID</li> <li>• TIR.TuneRule = 4: Ziegler-Nichols PI</li> <li>• TIR.TuneRule = 5: Ziegler-Nichols P</li> </ul>
PIDSelfTune.TIR.State	INT	0	The TIR.State tag indicates the current phase of fine tuning: <ul style="list-style-type: none"> <li>• State = -100 Fine tuning is not possible. Pre-tuning will be performed first.</li> <li>• State = 0: Initialize fine tuning</li> <li>• State = 200: Calculate the standard deviation</li> <li>• State = 300: Attempt to reach setpoint</li> <li>• State = 400: Attempt to reach setpoint with existing PID parameters (if pretuning was successful)</li> <li>• State = 500: Determine oscillation and calculate parameters</li> <li>• State = 9900: Fine tuning successful</li> <li>• State = 1: Fine tuning not successful</li> </ul>
PIDCtrl.IntegralSum	REAL	0.0	Current integral action
PIDCtrl.PIDInit	BOOL	FALSE	PIDCtrl.PIDInit is available as of PID_Compact version 2.3. If PIDCtrl.PIDInit = TRUE in "Automatic mode", the integral action PIDCtrl.IntegralSum is pre-assigned automatically as if Output = OverwriteInitialOutputValue in the previous cycle. This can be used for a Override control with PID_Compact V2 (Page 94).
Retain.CtrlParams.Gain	REAL	1.0	Active proportional gain To invert the control logic, use the Config.InvertControl tag. Negative values at Gain also invert the control logic. We recommend you use only InvertControl to set the control logic. The control logic is also inverted if InvertControl = TRUE and Gain < 0.0. Gain is retentive.

8.1 PID\_Compact

Tag	Data type	Default	Description
Retain.CtrlParams.Ti	REAL	20.0	<ul style="list-style-type: none"> <li>CtrlParams.Ti &gt; 0.0: Active integral action time</li> <li>CtrlParams.Ti = 0.0: Integral action is deactivated</li> </ul> Ti is retentive.
Retain.CtrlParams.Td	REAL	0.0	<ul style="list-style-type: none"> <li>CtrlParams.Td &gt; 0.0: Active derivative action time</li> <li>CtrlParams.Td = 0.0: Derivative action is deactivated</li> </ul> Td is retentive.
Retain.CtrlParams.TdFiltRatio	REAL	0.2	Active derivative delay coefficient The derivative delay coefficient delays the effect of the derivative action. Derivative delay = derivative action time × derivative delay coefficient <ul style="list-style-type: none"> <li>0.0: Derivative action is effective for one cycle only and therefore almost not effective.</li> <li>0.5: This value has proved useful in practice for controlled systems with <b>one</b> dominant time constant.</li> <li>&gt; 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed.</li> </ul> TdFiltRatio is retentive.
Retain.CtrlParams.PWeighting	REAL	1.0	Active proportional action weighting The proportional action may weaken with changes to the setpoint. Values from 0.0 to 1.0 are applicable. <ul style="list-style-type: none"> <li>1.0: Proportional action for setpoint change is fully effective</li> <li>0.0: Proportional action for setpoint change is not effective</li> </ul> The proportional action is always fully effective when the process value is changed. PWeighting is retentive.
Retain.CtrlParams.DWeighting	REAL	1.0	Active derivative action weighting The derivative action may weaken with changes to the setpoint. Values from 0.0 to 1.0 are applicable. <ul style="list-style-type: none"> <li>1.0: Derivative action is fully effective upon setpoint change</li> <li>0.0: Derivative action is not effective upon setpoint change</li> </ul> The derivative action is always fully effective when the process value is changed. DWeighting is retentive.

<b>Tag</b>	<b>Data type</b>	<b>Default</b>	<b>Description</b>
Retain.CtrlParams.Cycle	REAL	1.0	Active sampling time of the PID algorithm CtrlParams.Cycle is calculated during tuning and rounded to an integer multiple of CycleTime.Value. Cycle is retentive.

---

**Note**

Change the tags listed in this table in "Inactive" mode to prevent malfunction of the PID controller.

---

**See also**

Tag ActivateRecoverMode V2 (Page 280)

Tag Warning V2 (Page 282)

Downloading technology objects to device (Page 72)

8.1.3.7 Changing the PID\_Compact V2 interface

The following table shows what has changed in the PID\_Compact instruction interface.

PID_Compact V1	PID_Compact V2	Change
Input_PER	Input_PER	Data type from Word to Int
	Disturbance	New
	ErrorAck	New
	ModeActivate	New
Output_PER	Output_PER	Data type from Word to Int
Error	ErrorBits	Renamed
	Error	New
	Mode	New
sb_RunModeByStartup	RunModeByStartup	Function
	IntegralResetMode	
	OverwriteInitialOutputValue	New
	SetSubstituteOutput	New
	CancelTuningLevel	New
	SubstituteOutput	New

The following table shows which variables have been renamed.

PID_Compact V1.x	PID_Compact V2
sb_GetCycleTime	CycleTime.StartEstimation
sb_EnCyclEstimation	CycleTime.EnEstimation
sb_EnCyclMonitoring	CycleTime.EnMonitoring
sb_RunModeByStartup	RunModeByStartup
si_Unit	PhysicalUnit
si_Type	PhysicalQuantity
sd_Warning	Warning
sBackUp.r_Gain	CtrlParamsBackUp.Gain
sBackUp.r_Ti	CtrlParamsBackUp.Ti
sBackUp.r_Td	CtrlParamsBackUp.Td
sBackUp.r_A	CtrlParamsBackUp.TdFiltRatio
sBackUp.r_B	CtrlParamsBackUp.PWeighting
sBackUp.r_C	CtrlParamsBackUp.DWeighting
sBackUp.r_Cycle	CtrlParamsBackUp.Cycle
sPid_Calc.r_Cycle	CycleTime.Value
sPid_Calc.b_RunIn	PIDSelfTune.TIR.RunIn
sPid_Calc.b_CalcParamSUT	PIDSelfTune.SUT.CalculateParams
sPid_Calc.b_CalcParamTIR	PIDSelfTune.TIR.CalculateParams
sPid_Calc.i_CtrlTypeSUT	PIDSelfTune.SUT.TuneRule
sPid_Calc.i_CtrlTypeTIR	PIDSelfTune.TIR.TuneRule
sPid_Calc.r_Progress	Progress



<b>PID_Compact V1.x</b>	<b>PID_Compact V2</b>
sPid_Cmpt.r_Sp_Hlm	Config.SetpointUpperLimit
sPid_Cmpt.r_Sp_Llm	Config.SetpointLowerLimit
sPid_Cmpt.r_Pv_Norm_IN_1	Config.InputScaling.LowerPointIn
sPid_Cmpt.r_Pv_Norm_IN_2	Config.InputScaling.UpperPointIn
sPid_Cmpt.r_Pv_Norm_OUT_1	Config.InputScaling.LowerPointOut
sPid_Cmpt.r_Pv_Norm_OUT_2	Config.InputScaling.UpperPointOut
sPid_Cmpt.r_Lmn_Hlm	Config.OutputUpperLimit
sPid_Cmpt.r_Lmn_Llm	Config.OutputLowerLimit
sPid_Cmpt.b_Input_PER_On	Config.InputPerOn
sPid_Cmpt.b_LoadBackUp	LoadBackUp
sPid_Cmpt.b_InvCtrl	Config.InvertControl
sPid_Cmpt.r_Lmn_Pwm_PPTm	Config.MinimumOnTime
sPid_Cmpt.r_Lmn_Pwm_PBTm	Config.MinimumOffTime
sPid_Cmpt.r_Pv_Hlm	Config.InputUpperLimit
sPid_Cmpt.r_Pv_Llm	Config.InputLowerLimit
sPid_Cmpt.r_Pv_HWrn	Config.InputUpperWarning
sPid_Cmpt.r_Pv_LWrn	Config.InputLowerWarning
sParamCalc.i_Event_SUT	PIDSelfTune.SUT.State
sParamCalc.i_Event_TIR	PIDSelfTune.TIR.State
sRet.i_Mode	sRet.i_Mode has been omitted. The operating mode is changed using Mode and ModeActivate.
sRet.r_Ctrl_Gain	Retain.CtrlParams.Gain
sRet.r_Ctrl_Ti	Retain.CtrlParams.Ti
sRet.r_Ctrl_Td	Retain.CtrlParams.Td
sRet.r_Ctrl_A	Retain.CtrlParams.TdFiltRatio
sRet.r_Ctrl_B	Retain.CtrlParams.PWeighting
sRet.r_Ctrl_C	Retain.CtrlParams.DWeighting
sRet.r_Ctrl_Cycle	Retain.CtrlParams.Cycle

### 8.1.3.8 Parameters State and Mode V2

#### Correlation of the parameters

The State parameter shows the current operating mode of the PID controller. You cannot change the State parameter.

With a rising edge at ModeActivate, PID\_Compact switches to the operating mode saved in the Mode in-out parameter.

When the CPU is switched on or switches from Stop to RUN mode, PID\_Compact starts in the operating mode that is saved in the Mode parameter. To leave PID\_Compact in "Inactive" mode, set RunModeByStartup = FALSE.

#### Meaning of values

State / Mode	Description of operating mode
0	<p>Inactive</p> <p>In "Inactive" operating mode, the output value 0.0 is always output, regardless of Config.OutputUpperLimit and Config.OutputLowerLimit. Pulse width modulation is off.</p>
1	<p>Pretuning</p> <p>The pretuning determines the process response to a jump change of the output value and searches for the point of inflection. The PID parameters are calculated from the maximum rate of rise and dead time of the controlled system. You obtain the best PID parameters when you perform pretuning and fine tuning.</p> <p>Pretuning requirements:</p> <ul style="list-style-type: none"> <li>• Inactive (State = 0), manual mode (State = 4), or automatic mode (State = 3)</li> <li>• ManualEnable = FALSE</li> <li>• Reset = FALSE</li> <li>• The process value must not be too close to the setpoint.  <math> \text{Setpoint} - \text{Input}  &gt; 0.3 *  \text{Config.InputUpperLimit} - \text{Config.InputLowerLimit} </math> and  <math> \text{Setpoint} - \text{Input}  &gt; 0.5 *  \text{Setpoint} </math></li> <li>• The setpoint and the process value lie within the configured limits.</li> </ul> <p>The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise.</p> <p>The setpoint is frozen in the CurrentSetpoint tag. Tuning is canceled when:</p> <ul style="list-style-type: none"> <li>• <math>\text{Setpoint} &gt; \text{CurrentSetpoint} + \text{CancelTuningLevel}</math> or</li> <li>• <math>\text{Setpoint} &lt; \text{CurrentSetpoint} - \text{CancelTuningLevel}</math></li> </ul> <p>Before the PID parameters are recalculated, they are backed up and can be reactivated with LoadBackUp.</p> <p>The controller switches to automatic mode following successful pretuning. If pretuning is unsuccessful, the switchover of the operating mode is dependent on ActivateRecoverMode.</p> <p>The phase of pretuning is indicated with PIDSelfTune.SUT.State.</p>

State / Mode	Description of operating mode
2	<p>Fine tuning</p> <p>Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are recalculated based on the amplitude and frequency of this oscillation. PID parameters from fine tuning usually have better master control and disturbance characteristics than PID parameters from pretuning. You obtain the best PID parameters when you perform pretuning and fine tuning.</p> <p>PID_Compact automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value.</p> <p>The setpoint is frozen in the CurrentSetpoint tag. Tuning is canceled when:</p> <ul style="list-style-type: none"> <li>• Setpoint &gt; CurrentSetpoint + CancelTuningLevel</li> <li>or</li> <li>• Setpoint &lt; CurrentSetpoint - CancelTuningLevel</li> </ul> <p>Before the PID parameters are recalculated, they are backed up and can be reactivated with LoadBackUp.</p> <p>Requirements for fine tuning:</p> <ul style="list-style-type: none"> <li>• No disturbances are expected.</li> <li>• The setpoint and the process value lie within the configured limits.</li> <li>• ManualEnable = FALSE</li> <li>• Reset = FALSE</li> <li>• Automatic (State = 3), inactive (State = 0) or manual (State = 4) mode</li> </ul> <p>Fine tuning proceeds as follows when started from:</p> <ul style="list-style-type: none"> <li>• Automatic mode (State = 3)</li> </ul> <p>Start fine tuning from automatic mode if you wish to improve the existing PID parameters through tuning.</p> <p>PID_Compact controls the system using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.</p> <ul style="list-style-type: none"> <li>• Inactive (State = 0) or manual mode (State = 4)</li> </ul> <p>If the requirements for pretuning are met, pretuning is started. The determined PID parameters will be used for control until the control loop has stabilized and the requirements for fine tuning have been met.</p> <p>If the process value for pretuning is already too near the setpoint or PIDSelfTune.TIR.RunIn = TRUE, an attempt is made to reach the setpoint with the minimum or maximum output value. This can produce increased overshoot.</p> <p>Only then will fine tuning start.</p> <p>The controller switches to automatic mode following successful fine tuning. If fine tuning is unsuccessful, the switchover of the operating mode is dependent on ActivateRecoverMode.</p> <p>The "Fine tuning" phase is indicated with PIDSelfTune.TIR.State.</p>
3	<p>Automatic mode</p> <p>In automatic mode, PID_Compact corrects the controlled system in accordance with the parameters specified.</p> <p>The controller switches to automatic mode if one the following requirements is fulfilled:</p> <ul style="list-style-type: none"> <li>• Pretuning successfully completed</li> <li>• Fine tuning successfully completed</li> <li>• Changing of the Mode in-out parameter to the value 3 and a rising edge at ModeActivate.</li> </ul> <p>The switchover from automatic mode to manual mode is only bumpless if carried out in the commissioning editor.</p> <p>The ActivateRecoverMode tag is taken into consideration in automatic mode.</p>

8.1 PID\_Compact

State / Mode	Description of operating mode
4	<p>Manual mode</p> <p>In manual mode, you specify a manual output value in the ManualValue parameter.</p> <p>You can also activate this operating mode using ManualEnable = TRUE. We recommend that you change the operating mode using Mode and ModeActivate only.</p> <p>The switchover from manual mode to automatic mode is bumpless. Manual mode is also possible when an error is pending.</p>
5	<p>Substitute output value with error monitoring</p> <p>The control algorithm is deactivated. The SetSubstituteOutput tag determines which output value is output in this operating mode.</p> <ul style="list-style-type: none"> <li>• SetSubstituteOutput = FALSE: Last valid output value</li> <li>• SetSubstituteOutput = TRUE: Substitute output value</li> </ul> <p>You cannot activate this operating mode using Mode = 5.</p> <p>In the event of an error, it is activated instead of "Inactive" operating mode if all the following conditions are met:</p> <ul style="list-style-type: none"> <li>• Automatic mode (Mode = 3)</li> <li>• ActivateRecoverMode = TRUE</li> <li>• One or more errors have occurred in which ActivateRecoverMode is effective.</li> </ul> <p>As soon as the errors are no longer pending, PID_Compact switches back to automatic mode.</p>

**ENO characteristics**

If State = 0, then ENO = FALSE.

If State ≠ 0, then ENO = TRUE.

**Automatic switchover of operating mode during commissioning**

Automatic mode is activated following successful pretuning or fine tuning. The following table shows how Mode and State change during successful pretuning.

Cycle no.	Mode	State	Action
0	4	4	Set Mode = 1
1	1	4	Set ModeActivate = TRUE
1	4	1	Value of State is saved in Mode parameter Pretuning is started
n	4	1	Pretuning successfully completed
n	3	3	Automatic mode is started

PID\_Compact automatically switches the operating mode in the event of an error. The following table shows how Mode and State change during pretuning with errors.

Cycle no.	Mode	State	Action
0	4	4	Set Mode = 1
1	1	4	Set ModeActivate = TRUE
1	4	1	Value of State is saved in Mode parameter Pretuning is started
n	4	1	Pretuning canceled
n	4	4	Manual mode is started

If ActivateRecoverMode = TRUE, the operating mode that is saved in the Mode parameter is activated. At the start of pretuning or fine tuning, PID\_Compact has saved the value of State in the Mode in/out parameter. PID\_Compact therefore switches to the operating mode from which tuning was started.

If ActivateRecoverMode = FALSE, the system switches to "Inactive" operating mode.

**See also**

Output parameters of PID\_Compact V2 (Page 262)

**8.1.3.9 Parameter ErrorBits V2**

If several errors are pending simultaneously, the values of the ErrorBits are displayed with binary addition. The display of ErrorBits = 0003h, for example, indicates that the errors 0001h and 0002h are pending simultaneously.

In manual mode, PID\_Compact uses ManualValue as output value. The exception is Errorbits = 10000h.

ErrorBits (DW#16#...)	Description
0000	There is no error.
0001	The "Input" parameter is outside the process value limits. <ul style="list-style-type: none"> <li>• Input &gt; Config.InputUpperLimit or</li> <li>• Input &lt; Config.InputLowerLimit</li> </ul> If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact remains in automatic mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact switches to the operating mode that is saved in the Mode parameter.
0002	Invalid value at "Input_PER" parameter. Check whether an error is pending at the analog input. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact outputs the configured substitute output value. As soon as the error is no longer pending, PID_Compact switches back to automatic mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact switches to the operating mode that is saved in the Mode parameter.
0004	Error during fine tuning. Oscillation of the process value could not be maintained. If ActivateRecoverMode = TRUE before the error occurred, PID_Compact cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0008	Error at start of pretuning. The process value is too close to the setpoint. Start fine tuning. If ActivateRecoverMode = TRUE before the error occurred, PID_Compact cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0010	The setpoint was changed during tuning. You can set the permitted fluctuation of the setpoint at the CancelTuningLevel tag. If ActivateRecoverMode = TRUE before the error occurred, PID_Compact cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0020	Pretuning is not permitted during fine tuning. If ActivateRecoverMode = TRUE before the error occurred, PID_Compact remains in fine tuning mode.
0080	Error during pretuning. Incorrect configuration of output value limits. Check whether the limits of the output value are configured correctly and match the control logic. If ActivateRecoverMode = TRUE before the error occurred, PID_Compact cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0100	Error during fine tuning resulted in invalid parameters. If ActivateRecoverMode = TRUE before the error occurred, PID_Compact cancels the tuning and switches to the operating mode that is saved in the Mode parameter.

ErrorBits (DW#16#...)	Description
0200	<p>Invalid value at "Input" parameter: Value has an invalid number format.</p> <p>If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact outputs the configured substitute output value. As soon as the error is no longer pending, PID_Compact switches back to automatic mode.</p> <p>If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact switches to the operating mode that is saved in the Mode parameter.</p>
0400	<p>Calculation of output value failed. Check the PID parameters.</p> <p>If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact outputs the configured substitute output value. As soon as the error is no longer pending, PID_Compact switches back to automatic mode.</p> <p>If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact switches to the operating mode that is saved in the Mode parameter.</p>
0800	<p>Sampling time error: PID_Compact is not called within the sampling time of the cyclic interrupt OB.</p> <p>If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact remains in automatic mode.</p> <p>If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact switches to the operating mode that is saved in the Mode parameter.</p> <p>If this error occurred during simulation with PLCSIM, see the notes under Simulating PID_Compact V2 with PLCSIM (Page 98).</p>
1000	<p>Invalid value at "Setpoint" parameter: Value has an invalid number format.</p> <p>If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact outputs the configured substitute output value. As soon as the error is no longer pending, PID_Compact switches back to automatic mode.</p> <p>If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact switches to the operating mode that is saved in the Mode parameter.</p>
10000	<p>Invalid value at ManualValue parameter. Value has an invalid number format.</p> <p>If ActivateRecoverMode = TRUE before an error occurred, PID_Compact uses SubstituteOutput as the output value. As soon as you specify a valid value in ManualValue, PID_Compact uses it as the output value.</p>
20000	<p>Invalid value at SubstituteOutput tag. Value has an invalid number format.</p> <p>PID_Compact uses the output value low limit as the output value.</p> <p>If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_Compact switches back to automatic mode.</p>
40000	<p>Invalid value at Disturbance parameter. Value has an invalid number format.</p> <p>If automatic mode was active and ActivateRecoverMode = TRUE before the error occurred, Disturbance is set to zero. PID_Compact remains in automatic mode.</p> <p>If pretuning or fine tuning mode was active and ActivateRecoverMode = TRUE before the error occurred, PID_Compact switches to the operating mode saved in the Mode parameter. If Disturbance in the current phase has no effect on the output value, tuning is not be canceled.</p>

### 8.1.3.10 Tag ActivateRecoverMode V2

The ActivateRecoverMode tag determines the reaction to error. The Error parameter indicates if an error is pending. When the error is no longer pending, Error = FALSE. The ErrorBits parameter shows which errors have occurred.

#### Automatic mode

**NOTICE**

**Your system may be damaged.**

If ActivateRecoverMode = TRUE, PID\_Compact remains in automatic mode even if there is an error and the process limit values are exceeded. This may damage your system.

It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

ActivateRecoverMode	Description
FALSE	PID_Compact automatically switches to "Inactive" mode in the event of an error. The controller is only activated by a falling edge at Reset or a rising edge at ModeActivate.
TRUE	<p>If errors occur frequently in automatic mode, this setting has a negative effect on the control response, because PID_Compact switches between the calculated output value and the substitute output value at each error. In this case, check the ErrorBits parameter and eliminate the cause of the error.</p> <p>If one or more of the following errors occur, PID_Compact stays in automatic mode:</p> <ul style="list-style-type: none"> <li>• 0001h: The "Input" parameter is outside the process value limits.</li> <li>• 0800h: Sampling time error</li> <li>• 40000h: Invalid value at parameter Disturbance.</li> </ul> <p>If one or more of the following errors occur, PID_Compact switches to "Substitute output value with error monitoring" mode:</p> <ul style="list-style-type: none"> <li>• 0002h: Invalid value at Input_PER parameter.</li> <li>• 0200h: Invalid value at Input parameter.</li> <li>• 0400h: Calculation of output value failed.</li> <li>• 1000h: Invalid value at Setpoint parameter.</li> </ul> <p>If the following error occurs, PID_Compact switches to "Substitute output value with error monitoring" mode and moves the actuator to Config.OutputLowerLimit:</p> <ul style="list-style-type: none"> <li>• 20000h: Invalid value at SubstituteOutput tag. Value has an invalid number format.</li> </ul> <p>This characteristics are independent of SetSubstituteOutput.</p> <p>As soon as the errors are no longer pending, PID_Compact switches back to automatic mode.</p>



## Pretuning and fine tuning

ActivateRecoverMode	Description
FALSE	PID_Compact automatically switches to "Inactive" mode in the event of an error. The controller is only activated by a falling edge at Reset or a rising edge at ModeActivate.
TRUE	If the following error occurs, PID_Compact remains in the active mode: <ul style="list-style-type: none"><li>• 0020h: Pretuning is not permitted during fine tuning.</li></ul> The following errors are ignored: <ul style="list-style-type: none"><li>• 10000h: Invalid value at ManualValue parameter.</li><li>• 20000h: Invalid value at SubstituteOutput tag.</li></ul> When any other error occurs, PID_Compact cancels the tuning and switches to the mode from which tuning was started.

## Manual mode

ActivateRecoverMode is not effective in manual mode.

**8.1.3.11 Tag Warning V2**

If several warnings are pending simultaneously, the values of the Warning tag are displayed with binary addition. The display of warning 0003h, for example, indicates that the warnings 0001h and 0002h are pending simultaneously.

Warning (DW#16#....)	Description
0000	No warning pending.
0001	The point of inflection was not found during pretuning.
0004	The setpoint was limited to the configured limits.
0008	Not all the necessary controlled system properties were defined for the selected method of calculation. Instead, the PID parameters were calculated using the TIR.TuneRule = 3 method.
0010	The operating mode could not be changed because Reset = TRUE or ManualEnable = TRUE.
0020	The cycle time of the calling OB limits the sampling time of the PID algorithm. Improve results by using shorter OB cycle times.
0040	The process value exceeded one of its warning limits.
0080	Invalid value at Mode. The operating mode is not switched.
0100	The manual value was limited to the limits of the controller output.
0200	The specified rule for tuning is not supported. No PID parameters are calculated.
1000	The substitute output value cannot be reached because it is outside the output value limits.

The following warnings are deleted as soon as the cause is eliminated:

- 0001h
- 0004h
- 0008h
- 0040h
- 0100h

All other warnings are cleared with a rising edge at Reset or ErrorAck.

### 8.1.3.12 IntegralResetMode V2 tag

The IntegralResetMode tag determines how the integral action PIDCtrl.IntegralSum is pre-assigned:

- When switching from "Inactive" operating mode to "Automatic mode"
- With edge TRUE -> FALSE at parameter Reset and parameter Mode = 3

This setting only works for one cycle and is only effective if the integral action is activated (Retain.CtrlParams.Ti > 0.0 tag).

IntegralReset-Mode	Description
0	<p>Smooth</p> <p>The value of PIDCtrl.IntegralSum is pre-assigned so that the switchover is bumpless, which means "Automatic mode" starts with the output value = 0.0 (parameter Output) and there is no jump of the output value regardless of the control deviation (setpoint – actual value).</p>
1	<p>Delete</p> <p>We recommend setting the weighting of the proportional action (Retain.CtrlParams.PWeighting) to 1.0 if this option is used.</p> <p>The value of PIDCtrl.IntegralSum is deleted. Any control deviation will cause a jump change of the output value. The direction of the output value jump depends on the configured weighting of the proportional action (Retain.CtrlParams.PWeighting tag) and the control deviation:</p> <ul style="list-style-type: none"> <li>• Proportional action weighting = 1.0:           <p>Output value jump and control deviation have identical signs. Example: If the actual value value is smaller than the setpoint (positive control deviation), the output value jumps to a positive value.</p> </li> <li>• Proportional action weighting &lt; 1.0:           <p>For large control deviations, the output value jump and control deviation have identical signs. Example: If the actual value is much smaller than the setpoint (positive control deviation), the output value jumps to a positive value.</p> <p>For small control deviations, the output value jump and control deviation have different signs. Example: If the actual value value is just below the setpoint (positive control deviation), the output value jumps to a negative value. This is usually not desirable, because it results in a temporary increase in the control deviation.</p> <p>The smaller the configured weighting of the proportional action, the greater the control deviation must be to receive an output value jump with identical sign.</p> </li> </ul> <p>We recommend setting the weighting of the proportional action (Retain.CtrlParams.PWeighting) to 1.0 when this option is used. Otherwise, you may experience the undesirable behavior described for small control deviations. Alternatively, you can also use IntegralResetMode = 4. This option guarantees identical signs of the output value jump and control deviation independent of the configured weighting of the proportional action and the control deviation.</p>
2	<p>Hold</p> <p>The value of PIDCtrl.IntegralSum is not changed. You can define a new value using the user program.</p>

8.1 PID\_Compact

IntegralReset-Mode	Description
3	<p>Pre-assign</p> <p>The value of PIDCtrl.IntegralSum is automatically pre-assigned as if Output = OverwriteInitialOutput-Value in the last cycle.</p>
4	<p>Like setpoint change (only for PID_Compact with version <math>\geq 2.3</math>)</p> <p>The value of PIDCtrl.IntegralSum is automatically pre-assigned so that a similar output value jump results as for a PI controller in automatic mode in case of a setpoint change from the current actual value to the current setpoint.</p> <p>Any control deviation will cause a jump of the output value. Output value jump and control deviation have identical signs.</p> <p>Example: If the actual value value is smaller than the setpoint (positive control deviation), the output value jumps to a positive value. This is independent of the configured weighting of the proportional action and the control deviation.</p>

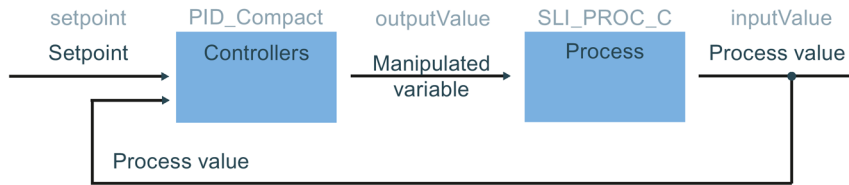
If IntegralResetMode is assigned a value outside the valid value range, PID\_Compact behaves as with the pre-assignment of IntegralResetMode:

- PID\_Compact up to V2.2: IntegralResetMode = 1
- PID\_Compact V2.3 and higher: IntegralResetMode = 4

All statements made above regarding the sign of the output value jump are based on a normal control logic (Config.InvertControl = FALSE tag). With an inverted control logic (Config.InvertControl = TRUE), the output value jump will have a reverse sign.

### 8.1.3.13 Sample program for PID\_Compact

In the following example, you are controlling temperature values with the technology object of the instruction "PID\_Compact". The temperature values are simulated based on a block which simulates a delay element of the third order (PT3 element). The PID parameters of the technology object can be set automatically via the pretuning.



### Data storage

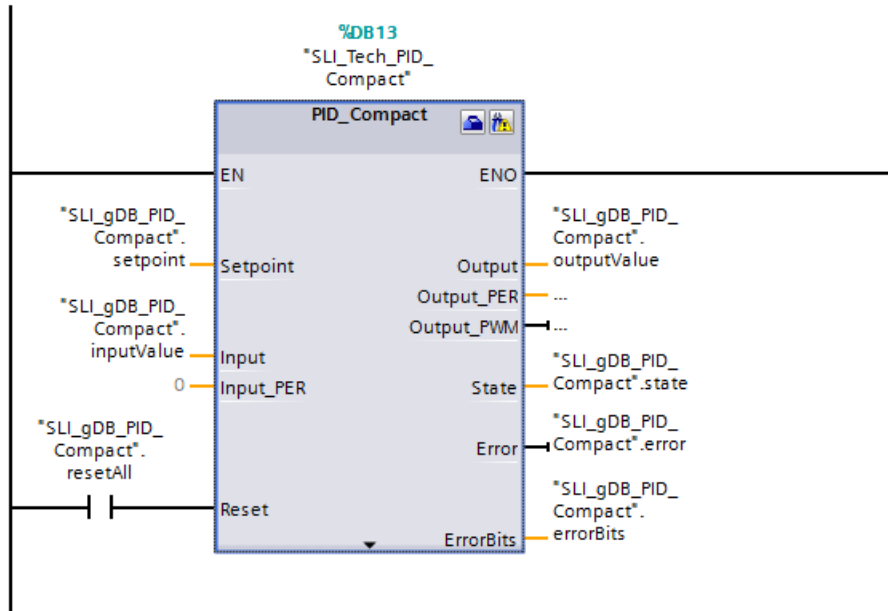
Create seven tags in a global data block for storage of the interconnection data.

SLI_gDB_PID_Compact			
	Name	Data type	Start value
1	Static		
2	setpoint	Real	75.0
3	inputValue	Real	0.0
4	outputValue	Real	0.0
5	state	Int	0
6	error	Bool	false
7	errorBits	DWord	16#0
8	resetAll	Bool	false

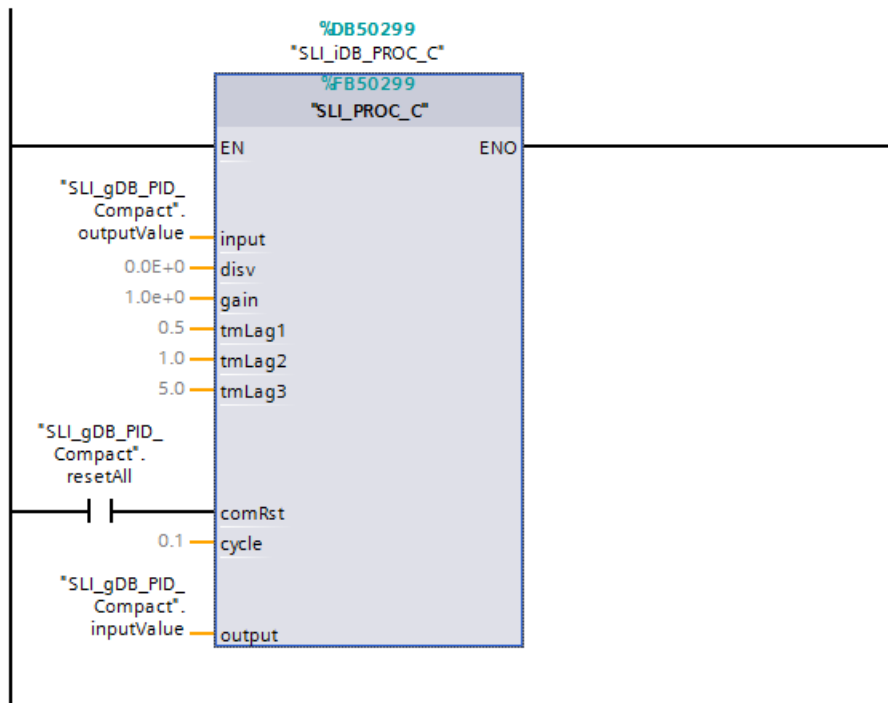
**Interconnection of the parameters**

You call the following interconnections in a cyclic interrupt OB.

**Network 1:** You interconnect the parameters of the instruction "PID\_Compact" as follows.



**Network 2:** You interconnect the parameters of the block simulating the temperature values "SLI\_PROC\_C" as follows.



## Technology object

You configure the technology object with the properties of the instruction "PID\_Compact" or by using the path Technology object > Configuration. The controller type and the input/output parameters are important for the example. With the controller type, you make a preselection for the unit of the value to be controlled. In this example, "Temperature" with the unit "°C" is used as controller type. The parameters of the "PID\_Compact" are already interconnected with global tags. Therefore, the information on use of the parameters Input and Output is sufficient.

The screenshot displays the configuration interface for the PID\_Compact instruction, divided into two main sections: "Basic settings" and "Input / output parameters".

**Basic settings:**

- Controller type:** A dropdown menu is set to "Temperature", and a unit dropdown is set to "°C".
- Invert control logic
- Activate Mode after CPU restart
- Set Mode to:** A dropdown menu is set to "Manual mode".

**Input / output parameters:**

This section features a central diagram of a PID controller block with a graph icon. It is connected to three input/output fields:

- Setpoint:** A dropdown menu with a blue icon and an empty text box.
- Input:** A dropdown menu set to "Input" and an empty text box with a blue icon.
- Output:** A dropdown menu set to "Output" and an empty text box with a blue icon.

## Procedure for starting the control

After the download to the CPU the PID\_Compact is in manual mode with manual value 0.0. To start the control, follow these steps:

1. Open the Commissioning of the technology object "SLI\_Tech\_PID\_Compact".
2. Click the "Start" button in the "Measurement" area.

The screenshot shows a control interface with two main modes: "Measurement" and "Tuning mode".

**Measurement mode:**

- Sampling time: 0.3 s
- A green "Start" button with a play icon.

**Tuning mode:**

- Pretuning: A dropdown menu.
- A green "Start" button with a play icon.

Measurement starts and PID\_Compact can be activated.

3. Pretuning is selected.

Click the "Start" button in the "Tuning mode" area.

A pretuning is performed. The PID parameters are automatically adjusted to the process. After the completion of the pretuning PID\_Compact switches to automatic mode.

---

**Note**

**Alternative to start PID\_Compact**

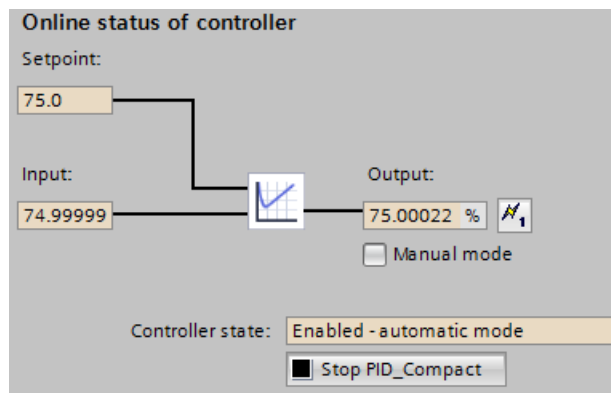
Alternatively, you can switch PID\_Compact to automatic mode in the "Online status of controller" area with the "Stop PID\_Compact" / "Start PID\_Compact" without pretuning. In this case the controller uses default values for the PID parameters and shows a worse controller behavior for the application case.

---

**Procedure for stopping control**

To stop and exit PID\_Compact and the program, follow these steps:

1. Click the "Stop PID\_Compact" button in the technology object "SLI\_Tech\_PID\_Compact" in the "Online status of controller" area.



The instruction "PID\_Compact" exits the control and outputs the value "0.0" as manipulated variable.

2. Click the "Stop" button in the "Measurement" area.
3. To set the process value immediately to the value "0.0", follow these steps:

In the block "SLI\_OB\_PID\_Compact", set the "resetAll" tag to the value "TRUE", and then to the value "FALSE".

**"PID\_Compact" instruction**

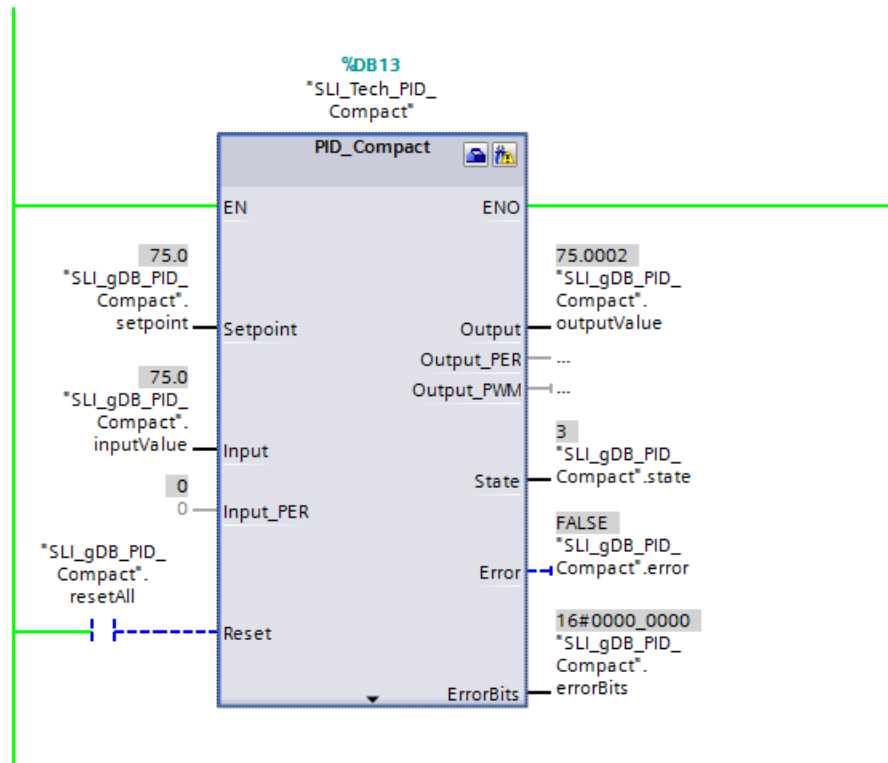
The setpoint for the temperature that is to be controlled is specified at the parameter Setpoint ("setpoint"). The control is started when the instruction "PID\_Compact" was started with the technology object. The instruction "PID\_Compact" outputs a manipulated variable at the output parameter Output ("outputValue"). The process value of the temperature is transferred to the instruction "PID\_Compact" with the input parameter Input ("inputValue").



The instruction "PID\_Compact" adjusts the manipulated variable ("outputValue") depending on the history of the difference between setpoint ("setpoint") and process value ("inputValue"). The process is repeated so that the process value ("inputValue") approaches the setpoint ("setpoint") through the manipulated variable ("outputValue").

The current operating mode of the instruction "PID\_Compact" is displayed at the output parameter State ("state"). After pretuning (the value of "state" is "1"), the PID\_Compact switches to automatic mode (the value is "3").

The output parameter Error ("error") currently shows that no error is pending. The output parameter ErrorBits ("errorBits") provides information on the error type in case of error. If an error occurs, this can be acknowledged in the technology object, in the optimization status area with the "ErrorAck" button.

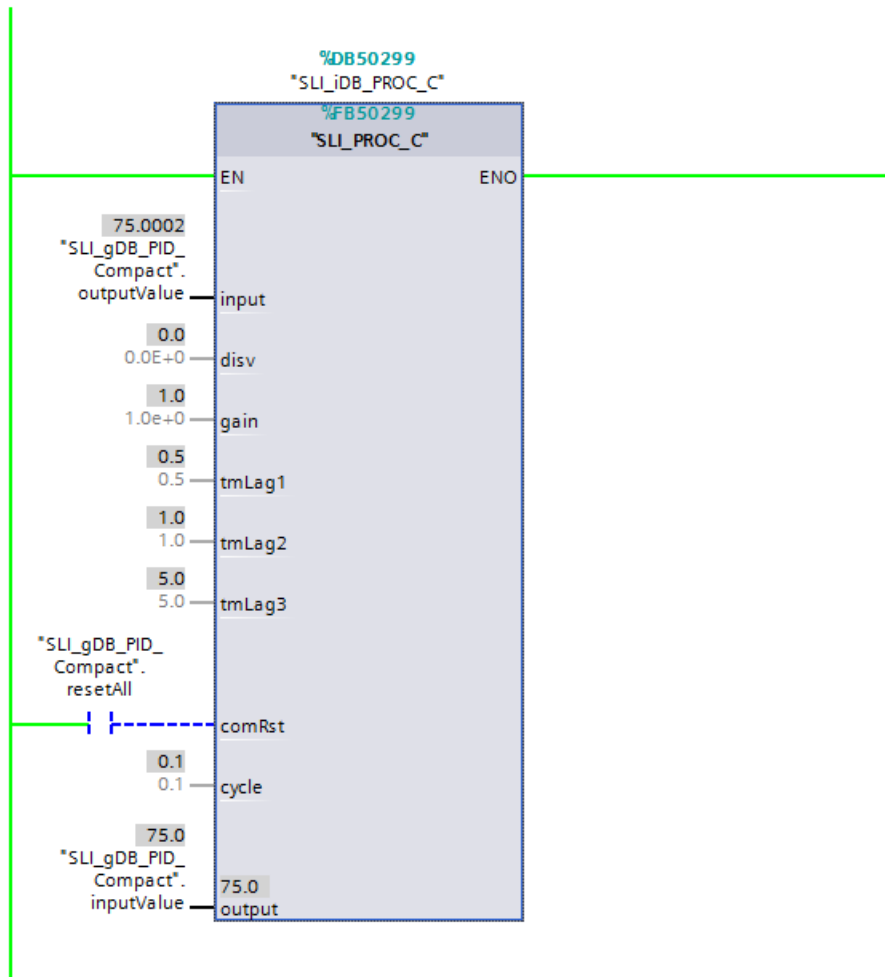


### "SLI\_PROC\_C" block

The "SLI\_PROC\_C" block simulates the process value ("inputValue") of the rising temperature of a plant. The block "SLI\_PROC\_C" contains the manipulated variable of the controller ("outputValue") and simulates the temperature behavior of the process. This temperature is fed as process value ("inputValue") into the controller.

A change in the values of the "resetAll" tag (of the comRst parameter) has the following effects:

Parameter comRst ("resetAll")	The instruction "PID_Compact" is running	The instruction "PID_Compact" was stopped
comRst ("resetAll") remains set to the value "FALSE"	The "SLI_PROC_C" block outputs a new process value ("inputValue") based on a manipulated variable ("outputValue").	The "SLI_PROC_C" block does not receive a manipulated variable > "0.0", but it still outputs a new process value > "0.0".
comRst ("resetAll"): Change from "FALSE" to the value "TRUE"	Both manipulated variable ("outputValue") and output process value ("inputValue") are reset to "0.0".	The output process value ("inputValue") / the temperature of the "SLI_PROC_C" block is reset to "0.0".
comRst ("resetAll"): Change from "TRUE" to the value "FALSE"	Temperature control starts again.	The output process value / the temperature ("inputValue") remains "0.0".



**Program code**

You can find additional information about the program code for the above-named example under the keyword "Sample Library for Instructions".

## 8.1.4 CPU processing time and memory requirement PID\_Compact V2.x

### CPU processing time

Typical CPU processing times of the PID\_Compact technology object as of Version V2.0, depending on CPU type.

CPU	Typ. CPU processing time PID_Compact V2.x
CPU 1211C ≥ V4.0	300 μs
CPU 1215C ≥ V4.0	300 μs
CPU 1217C ≥ V4.0	300 μs
CPU 1505S ≥ V1.0	45 μs
CPU 1510SP-1 PN ≥ V1.6	85 μs
CPU 1511-1 PN ≥ V1.5	85 μs
CPU 1512SP-1 PN ≥ V1.6	85 μs
CPU 1516-3 PN/DP ≥ V1.5	50 μs
CPU 1518-4 PN/DP ≥ V1.5	4 μs

### Memory requirement

Memory requirement of an instance DB of the PID\_Compact technology object as of Version V2.0.

	Memory requirement of the instance DB of PID_Compact V2.x
Load memory requirement	Approx. 12000 bytes
Total work memory requirement	788 bytes
Retentive work memory requirement	44 bytes

## 8.1.5 PID\_Compact V1

### 8.1.5.1 Description of PID\_Compact V1

#### Description

The PID\_Compact instruction provides a PID controller with integrated tuning for automatic and manual mode.

#### Call

PID\_Compact is called in the constant interval of the cycle time of the calling OB (preferably in a cyclic interrupt OB).

#### Download to device

The actual values of retentive tags are only updated when you download PID\_Compact completely.

Downloading technology objects to device (Page 72)

#### Startup

At the startup of the CPU, PID\_Compact starts in the operating mode that was last active. To retain PID\_Compact in "Inactive" mode, set `sb_RunModeByStartup = FALSE`.

#### Monitoring of the sampling time PID\_Compact

Ideally, the sampling time is equivalent to the cycle time of the calling OB. The PID\_Compact instruction measures the time interval between two calls. This is the current sampling time. On every switchover of operating mode and during the initial startup, the mean value is formed from the first 10 sampling times. If the current sampling time deviates too much from this mean value, Error = 0800 hex occurs and PID\_Compact switches to "Inactive" mode.

PID\_Compact, Version 1.1 or higher is set to "Inactive" mode during controller tuning under the following conditions:

- New mean value  $\geq 1.1 \times$  old mean value
- New mean value  $\leq 0.9 \times$  old mean value

In automatic mode, PID\_Compact, Version 1.1 or higher, is set to "Inactive" mode under the following conditions:

- New mean value  $\geq 1.5 \times$  old mean value
- New mean value  $\leq 0.5 \times$  old mean value

During controller tuning and in automatic mode, PID\_Compact 1.0 is set to "Inactive" operating mode under the following conditions:

- New mean value  $\geq 1.1 \times$  old mean value
- New mean value  $\leq 0.9 \times$  old mean value
- Current sampling time  $\geq 1.5 \times$  current mean value
- Current sampling time  $\leq 0.5 \times$  current mean value

### Sampling time of the PID algorithm

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the cycle time. All other functions of PID\_Compact are executed at every call.

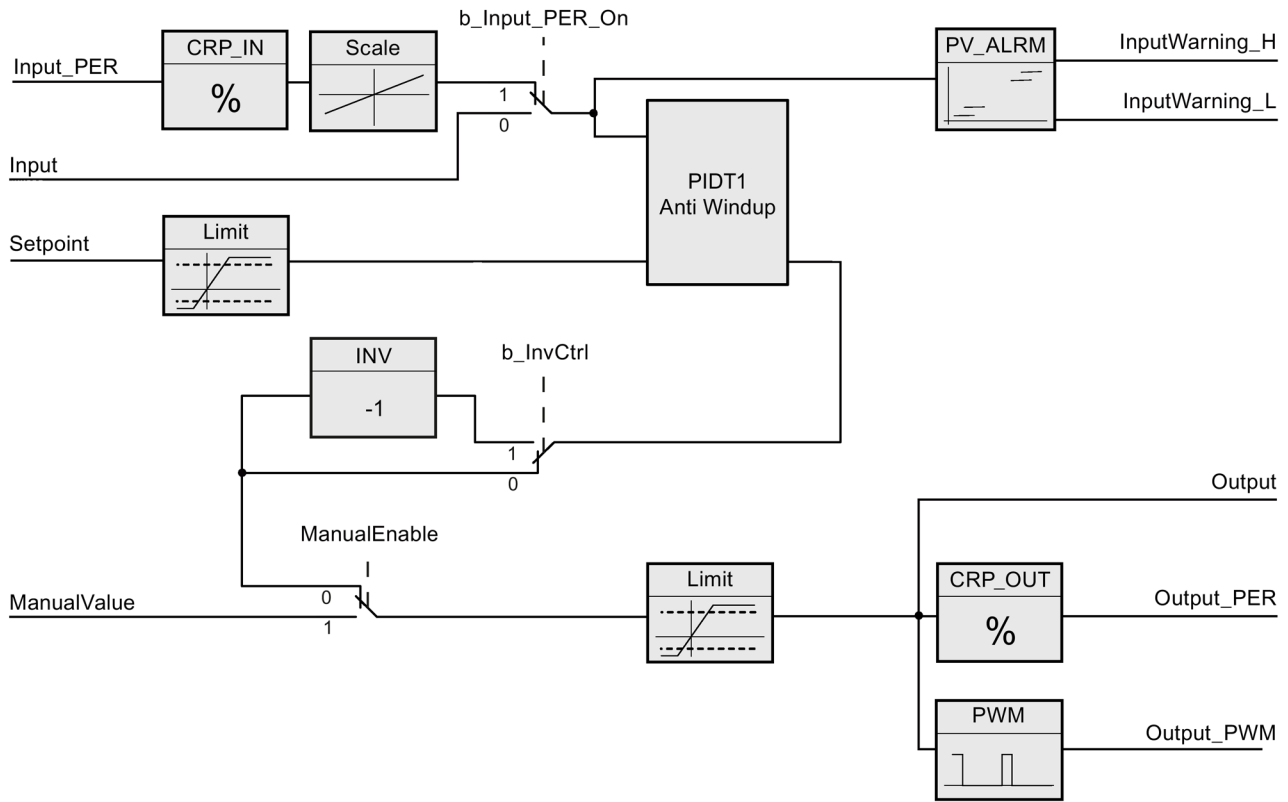
### PID algorithm

PID\_Compact is a PIDT1 controller with anti-windup and weighting of the proportional and derivative actions. The following equation is used to calculate the output value.

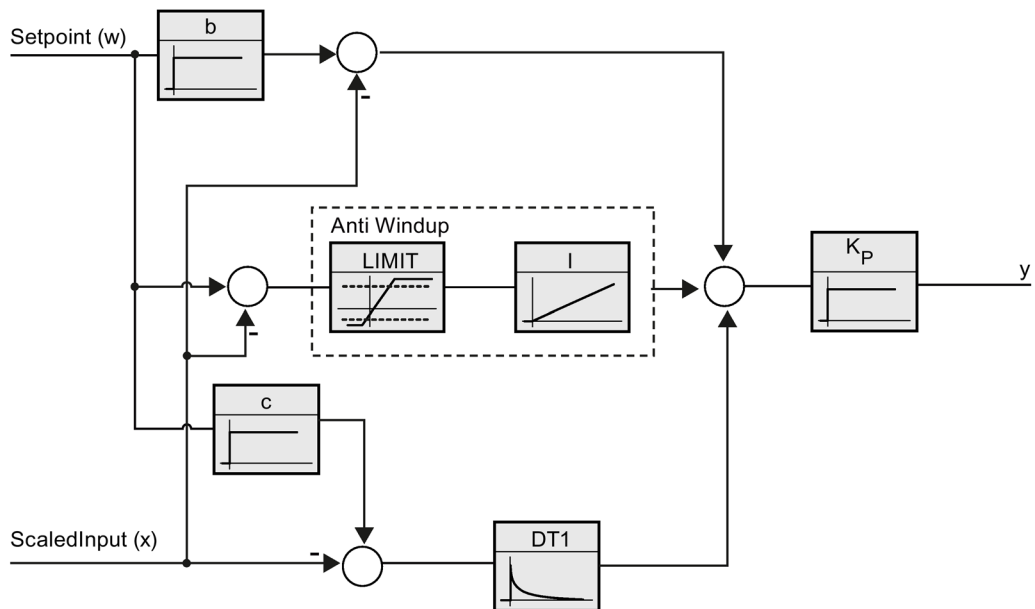
$$y = K_p \left[ (b \cdot w - x) + \frac{1}{T_i \cdot s} (w - x) + \frac{T_d \cdot s}{a \cdot T_d \cdot s + 1} (c \cdot w - x) \right]$$

<b>Symbol</b>	<b>Description</b>
y	Output value
K <sub>p</sub>	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T <sub>i</sub>	Integral action time
a	Derivative delay coefficient (T <sub>1</sub> = a × T <sub>D</sub> )
	Derivative action time
c	Derivative action weighting

Block diagram of PID\_Compact



Block diagram of PIDT1 with anti-windup



### **Reaction to error**

If errors occur, they are output in parameter Error, and PID\_Compact changes to "Inactive" mode. Reset the errors using the Reset parameter.

### **Control logic**

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic. For cooling and discharge control systems, it may be necessary to invert the control logic. PID\_Compact does not work with negative proportional gain. If InvertControl = TRUE, an increasing control deviation causes a reduction in the output value. The control logic is also taken into account during pretuning and fine tuning.

### **See also**

Control mode V1 (Page 100)

8.1.5.2 Input parameters of PID\_Compact V1

Table 8- 4

Parameter	Data type	Default	Description
Setpoint	REAL	0.0	Setpoint of the PID controller in automatic mode
Input	REAL	0.0	A variable of the user program is used as source for the process value. If you are using parameter Input, then sPid_Cmpt.b_Input_PER_On = FALSE must be set.
Input_PER	WORD	W#16#0	Analog input as the source of the process value If you are using parameter Input_PER, then sPid_Cmpt.b_Input_PER_On = TRUE must be set.
ManualEnable	BOOL	FALSE	<ul style="list-style-type: none"> <li>• A FALSE -&gt; TRUE edge selects "Manual mode", while State = 4, sRet.i_Mode remains unchanged.</li> <li>• A TRUE -&gt; FALSE edge selects the most recently active operating mode, State =sRet.i_Mode</li> </ul> <p>A change of sRet.i_Mode will not take effect during ManualEnable = TRUE. The change of sRet.i_Mode will only be considered upon a TRUE -&gt; FALSE edge at ManualEnable .</p> <p><b>PID_Compact V1.2 und PID_Compact V1.0</b> If at start of the CPU ManualEnable = TRUE, PID_Compact starts in manual mode. A rising edge (FALSE &gt; TRUE) at ManualEnable is not necessary.</p> <p><b>PID_Compact V1.1</b> At the start of the CPU, PID_Compact only switches to manual mode with a rising edge (FALSE-&gt;TRUE) at ManualEnable . Without rising edge, PID_Compact starts in the last operating mode in which ManualEnable was FALSE.</p>
ManualValue	REAL	0.0	Manual value This value is used as the output value in manual mode.
Reset	BOOL	FALSE	The Reset parameter (Page 308) restarts the controller.



### 8.1.5.3 Output parameters of PID\_Compact V1

Table 8- 5

Parameter	Data type	Default	Description
ScaledInput	REAL	0.0	Output of the scaled process value
Outputs "Output", "Output_PER", and "Output_PWM" can be used concurrently.			
Output	REAL	0.0	Output value in REAL format
Output_PER	WORD	W#16#0	Analog output value
Output_PWM	BOOL	FALSE	Pulse-width-modulated output value The output value is formed by minimum On and Off times.
SetpointLimit_H	BOOL	FALSE	If SetpointLimit_H = TRUE, the setpoint absolute high limit is reached. The setpoint in the CPU is limited to the configured setpoint absolute high limit. The configured process value absolute high limit is the default for the setpoint high limit.  If you set sPid_Cmpt.r_Sp_Hlm to a value within the process value limits, this value is used as the setpoint high limit.
SetpointLimit_L	BOOL	FALSE	If SetpointLimit_L = TRUE, the setpoint absolute low limit has been reached. In the CPU, the setpoint is limited to the configured setpoint absolute low limit. The configured process value absolute low limit is the default setting for the setpoint low limit.  If you set sPid_Cmpt.r_Sp_Llm to a value within the process value limits, this value is used as the setpoint low limit.
InputWarning_H	BOOL	FALSE	If InputWarning_H = TRUE, the process value has reached or exceeded the warning high limit.
InputWarning_L	BOOL	FALSE	If InputWarning_L = TRUE, the process value has reached or fallen below the warning low limit.
State	INT	0	The State parameter (Page 303) shows the current operating mode of the PID controller. To change the operating mode, use variable sRet.i_Mode. <ul style="list-style-type: none"> <li>• State = 0: Inactive</li> <li>• State = 1: pretuning</li> <li>• State = 2: fine tuning</li> <li>• State = 3: Automatic mode</li> <li>• State = 4: Manual mode</li> </ul>
Error	DWORD	W#16#0	The Error parameter (Page 307) indicates the error messages. Error = 0000: No error pending.

8.1.5.4 Static tags of PID\_Compact V1

You must not change tags that are not listed. These are used for internal purposes only.

Table 8- 6

Tag	Data type	Default	Description
sb_GetCycleTime	BOOL	TRUE	If sb_GetCycleTime = TRUE, the automatic determination of the cycle time is started. CycleTime.StartEstimation = FALSE once measurement is complete.
sb_EnCyclEstimation	BOOL	TRUE	If sb_EnCyclEstimation = TRUE, the PID_Compact sampling time is not monitored.
sb_EnCyclMonitoring	BOOL	TRUE	If sb_EnCyclMonitoring = FALSE, the PID_Compact sampling time is not monitored. If it is not possible to execute PID_Compact within the sampling time, an 0800 error is not output and PID_Compact does not change to "Inactive" mode.
sb_RunModeByStartup	BOOL	TRUE	Activate Mode after CPU restart If sb_RunModeByStartup = FALSE, the controller will remain inactive after a CPU startup. After a CPU startup and if sb_RunModeByStartup = TRUE, the controller will return to the most recently active operating mode.
si_Unit	INT	0	Unit of measurement of the process value and setpoint, e.g., °C, or °F.
si_Type	INT	0	Physical quantity of the process value and setpoint, e.g., temperature.
sd_Warning	DWORD	DW#16#0	Variable sd_warning (Page 310) displays the warnings generated since the reset, or since the last change of the operating mode.
sBackUp.r_Gain	REAL	1.0	Saved proportional gain You can reload values from the sBackUp structure with sPid_Cmpt.b_LoadBackUp = TRUE.
sBackUp.r_Ti	REAL	20.0	Saved integral action time [s]
sBackUp.r_Td	REAL	0.0	Saved derivative action time [s]
sBackUp.r_A	REAL	0.0	Saved derivative delay coefficient
sBackUp.r_B	REAL	0.0	Saved proportional action weighting factor
sBackUp.r_C	REAL	0.0	Saved derivative action weighting factor
sBackUp.r_Cycle	REAL	1.0	Saved sampling time of PID algorithm
sPid_Calc.r_Cycle	REAL	0.1	Sampling time of the PID_Compact instruction r_Cycle is determined automatically and is usually equivalent to the cycle time of the calling OB.

Tag	Data type	Default	Description
sPid_Calc.b_RunIn	BOOL	FALSE	<ul style="list-style-type: none"> <li>b_RunIn = FALSE</li> </ul> <p>Pretuning is started when fine tuning is started from inactive or manual mode. If the requirements for pretuning are not met, PID_Compact reacts as if b_RunIn = TRUE.</p> <p>If fine tuning is started from automatic mode, the system uses the existing PID parameters to control to the setpoint.</p> <p>Only then will fine tuning start. If pretuning is not possible, PID_Compact switches to "Inactive" mode.</p> <ul style="list-style-type: none"> <li>b_RunIn = TRUE</li> </ul> <p>The pretuning is skipped. PID_3Compact tries to reach the setpoint with minimum or maximum output value. This can produce increased overshoot. Fine tuning then starts automatically.</p> <p>b_RunIn is set to FALSE after fine tuning.</p>
sPid_Calc.b_CalcParamSUT	BOOL	FALSE	<p>The parameters for pretuning will be recalculated if b_CalcParamSUT = TRUE. This enables you to change the parameter calculation method without having to repeat controller tuning.</p> <p>b_CalcParamSUT is set to FALSE after the calculation.</p>
sPid_Calc.b_CalcParamTIR	BOOL	FALSE	<p>The parameters for fine tuning will be recalculated if b_CalcParamTIR = TRUE. This enables you to change the parameter calculation method without having to repeat controller tuning.</p> <p>b_CalcParamTIR will be set to FALSE after calculation.</p>
sPid_Calc.i_CtrlTypeSUT	INT	0	<p>Methods used to calculate parameters during pretuning:</p> <ul style="list-style-type: none"> <li>i_CtrlTypeSUT = 0: PID according to Chien, Hrones and Reswick</li> <li>i_CtrlTypeSUT = 1: PI according to Chien, Hrones and Reswick</li> </ul>
sPid_Calc.i_CtrlTypeTIR	INT	0	<p>Methods used to calculate parameters during fine tuning:</p> <ul style="list-style-type: none"> <li>i_CtrlTypeTIR = 0: PID automatic</li> <li>i_CtrlTypeTIR = 1: PID fast</li> <li>i_CtrlTypeTIR = 2: PID slow</li> <li>i_CtrlTypeTIR = 3: Ziegler-Nichols PID</li> <li>i_CtrlTypeTIR = 4: Ziegler-Nichols PI</li> <li>i_CtrlTypeTIR = 5: Ziegler-Nichols P</li> </ul>
sPid_Calc.r_Progress	REAL	0.0	Progress of tuning as a percentage (0.0 - 100.0)

8.1 PID\_Compact

Tag	Data type	Default	Description
sPid_Cmpt.r_Sp_Hlm	REAL	+3.402822e+38	High limit of setpoint If you configure sPid_Cmpt.r_Sp_Hlm outside the process value limits, the configured process value absolute high limit is used as the setpoint high limit. If you set sPid_Cmpt.r_Sp_Hlm within the process value limits, this value is used as the setpoint high limit.
sPid_Cmpt.r_Sp_Llm	REAL	-3.402822e+38	Low limit of the setpoint If you set sPid_Cmpt.r_Sp_Llm outside the process value limits, the configured process value absolute low limit is used as the setpoint low limit. If you set sPid_Cmpt.r_Sp_Llm within the process value limits, this value is used as the setpoint low limit.
sPid_Cmpt.r_Pv_Norm_IN_1	REAL	0.0	Scaling Input_PER low Input_PER is converted to percent based on the two value pairs r_Pv_Norm_OUT_1, r_Pv_Norm_IN_1 and r_Pv_Norm_OUT_2, r_Pv_Norm_IN_2 from the sPid_Cmpt structure.
sPid_Cmpt.r_Pv_Norm_IN_2	REAL	27648.0	Scaling Input_PER high Input_PER is converted to percent based on the two value pairs r_Pv_Norm_OUT_1, r_Pv_Norm_IN_1 and r_Pv_Norm_OUT_2, r_Pv_Norm_IN_2 from the sPid_Cmpt structure.
sPid_Cmpt.r_Pv_Norm_OUT_1	REAL	0.0	Scaled low process value Input_PER is converted to percent based on the two value pairs r_Pv_Norm_OUT_1, r_Pv_Norm_IN_1 and r_Pv_Norm_OUT_2, r_Pv_Norm_IN_2 from the sPid_Cmpt structure.
sPid_Cmpt.r_Pv_Norm_OUT_2	REAL	100.0	Scaled high process value Input_PER is converted to percent based on the two value pairs r_Pv_Norm_OUT_1, r_Pv_Norm_IN_1 and r_Pv_Norm_OUT_2, r_Pv_Norm_IN_2 from the sPid_Cmpt structure.
sPid_Cmpt.r_Lmn_Hlm	REAL	100.0	Output value high limit for output parameter "Output"
sPid_Cmpt.r_Lmn_Llm	REAL	0.0	Low output value limit for output parameter "Output"
sPid_Cmpt.b_Input_PER_On	BOOL	TRUE	If b_Input_PER_On = TRUE, then parameter Input_PER is used. If b_Input_PER_On = FALSE, then parameter Input is used.
sPid_Cmpt.b_LoadBackUp	BOOL	FALSE	Activate the back-up parameter set. If an optimization has failed, you can reactivate the previous PID parameters by setting this bit.
sPid_Cmpt.b_InvCtrl	BOOL	FALSE	Invert control logic With b_InvCtrl = TRUE, an increasing control deviation causes a reduction in the output value.
sPid_Cmpt.r_Lmn_Pwm_PPTm	REAL	0.0	The minimum ON time of the pulse width modulation in seconds is rounded to $r\_Lmn\_Pwm\_PPTm = r\_Cycle$ or $r\_Lmn\_Pwm\_PPTm = n * r\_Cycle$

Tag	Data type	Default	Description
sPid_Cmpt.r_Lmn_Pwm_PBTm	REAL	0.0	The minimum OFF time of the pulse width modulation in seconds is rounded to $r\_Lmn\_Pwm\_PBTm = r\_Cycle$ or $r\_Lmn\_Pwm\_PBTm = n * r\_Cycle$
sPid_Cmpt.r_Pv_Hlm	REAL	120.0	High limit of the process value At the I/O input, the process value can be a maximum of 18% higher than the standard range (over-range). An error is no longer reported for a violation of the "Process value high limit". Only a wire-break and a short-circuit are recognized and the PID_Compact switches to "Inactive" mode. $r\_Pv\_Hlm > r\_Pv\_Llm$
sPid_Cmpt.r_Pv_Llm	REAL	0.0	Low limit of the process value $r\_Pv\_Llm < r\_Pv\_Hlm$
sPid_Cmpt.r_Pv_HWrn	REAL	+3.402822e+38	Warning high limit of the process value If you set $r\_Pv\_HWrn$ outside the process value limits, the configured process value absolute high limit is used as the warning high limit. If you configure $r\_Pv\_HWrn$ within the process value limits, this value is used as the warning high limit. $r\_Pv\_HWrn > r\_Pv\_LWrn$ $r\_Pv\_HWrn \leq r\_Pv\_Hlm$
sPid_Cmpt.r_Pv_LWrn	REAL	-3.402822e+38	Warning low limit of the process value If you set $r\_Pv\_LWrn$ outside the process value limits, the configured process value absolute low limit is used as the warning low limit. If you configure $r\_Pv\_LWrn$ within the process value limits, this value is used as the warning low limit. $r\_Pv\_LWrn < r\_Pv\_HWrn$ $r\_Pv\_LWrn \geq r\_Pv\_LWrn$
sPidCalc.i_Ctrl_IOutv	REAL	0.0	Current integral action
sParamCalc.i_Event_SUT	INT	0	Variable $i\_Event\_SUT$ (Page 310) indicates the current phase of "pretuning":
sParamCalc.i_Event_TIR	INT	0	Variable $i\_Event\_TIR$ (Page 311) indicates the current phase of "fine tuning":
sRet.i_Mode	INT	0	The operating mode is changed edge-triggered. The following operating mode is enabled on a change to <ul style="list-style-type: none"> <li>• <math>i\_Mode = 0</math>: "Inactive" mode (controller stop)</li> <li>• <math>i\_Mode = 1</math>: "Pretuning" mode</li> <li>• <math>i\_Mode = 2</math>: "Fine tuning" mode</li> <li>• <math>i\_Mode = 3</math>: "Automatic" mode</li> <li>• <math>i\_Mode = 4</math>: "Manual" mode</li> </ul> $i\_Mode$ is retentive.
sRet.r_Ctrl_Gain	REAL	1.0	Active proportional gain Gain is retentive.

Tag	Data type	Default	Description
sRet.r_Ctrl_Ti	REAL	20.0	<ul style="list-style-type: none"> <li>r_Ctrl_Ti &gt; 0.0: Active integral action time</li> <li>r_Ctrl_Ti = 0.0: Integral action is deactivated</li> </ul> r_Ctrl_Ti is retentive.
sRet.r_Ctrl_Td	REAL	0.0	<ul style="list-style-type: none"> <li>r_Ctrl_Td &gt; 0.0: Active derivative action time</li> <li>r_Ctrl_Td = 0.0: Derivative action is deactivated</li> </ul> r_Ctrl_Td is retentive.
sRet.r_Ctrl_A	REAL	0.0	Active derivative delay coefficient r_Ctrl_A is retentive.
sRet.r_Ctrl_B	REAL	0.0	Active proportional action weighting r_Ctrl_B is retentive.
sRet.r_Ctrl_C	REAL	0.0	Active derivative action weighting r_Ctrl_C is retentive.
sRet.r_Ctrl_Cycle	REAL	1.0	Active sampling time of the PID algorithm r_Ctrl_Cycle is calculated during tuning and rounded to an integer multiple of r_Cycle. r_Ctrl_Cycle is retentive.

**Note**

Change the tags listed in this table in "Inactive" mode to prevent malfunction of the PID controller. "Inactive" mode is forced by setting the "sRet.i\_Mode" tag to "0".

**See also**

Downloading technology objects to device (Page 72)

### 8.1.5.5 Parameters State and sRet.i\_Mode V1

#### Correlation of the parameters

The State parameter indicates the current operating mode of the PID controller. You cannot modify the State parameter.

You need to modify the sRet.i\_Mode tag to change the operating mode. This also applies when the value for the new operating mode is already in sRet.i\_Mode. First set sRet.i\_Mode = 0 and then sRet.i\_Mode = 3. Provided the current operating mode of the controller supports this change, State is set to the value of sRet.i\_Mode.

When PID\_Compact automatically switches the operating mode, the following applies: State != sRet.i\_Mode.

Examples:

- Successful pretuning  
State = 3 and sRet.i\_Mode = 1
- Error  
State = 0 and sRet.i\_Mode remains at the same value, e.g sRet.i\_Mode = 3
- ManualEnable = TRUE  
State = 4 and sRet.i\_Mode remain at the previous value, for example, sRet.i\_Mode = 3

---

#### **Note**

You wish to repeat successful fine tuning without exiting automatic mode with i\_Mode = 0.

Setting sRet.i\_Mode to an invalid value such as 9999 for one cycle has no effect on State. Set Mode = 2 in the next cycle. You can generate a change to sRet.i\_Mode without first switching to "inactive" mode.

---

Meaning of values

State / sRet.i_Mode	Description of the operating mode
0	<p>Inactive</p> <p>The controller is switched off.</p> <p>The controller was in "inactive" mode before pretuning was performed.</p> <p>The PID controller will change to "inactive" mode when running if an error occurs or if the "Deactivate controller" icon is clicked in the commissioning window.</p>
1	<p>Pretuning</p> <p>The pretuning determines the process response to a jump of the output value and searches for the point of inflection. The optimized PID parameters are calculated as a function of the maximum rate of rise and dead time of the controlled system.</p> <p>Pretuning requirements:</p> <ul style="list-style-type: none"> <li>• The controller is in inactive mode or manual mode</li> <li>• ManualEnable = FALSE</li> <li>• The process value must not be too close to the setpoint.                               <math> Setpoint - Input  &gt; 0.3 *  sPid\_Cmpt.r\_Pv\_Hlm - sPid\_Cmpt.r\_Pv\_Llm </math> and                               <math> Setpoint - Input  &gt; 0.5 *  Setpoint </math> </li> <li>• The setpoint may not be changed during pretuning.</li> </ul> <p>The higher the stability of the process value, the easier it is to calculate the PID parameters and increase precision of the result. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise.</p> <p>PID parameters are backed up before they are recalculated and can be reactivated with sPid_Cmpt.b_LoadBackUp.</p> <p>There is a change to automatic mode following successful pretuning and to "inactive" mode following unsuccessful pretuning.</p> <p>The phase of pretuning is indicated with Tag i_Event_SUT V1 (Page 310).</p>



State / sRet.i_Mode	Description of the operating mode
2	<p data-bbox="319 319 438 351">Fine tuning</p> <p data-bbox="319 357 1460 500">Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are optimized based on the amplitude and frequency of this oscillation. The differences between the process response during pretuning and fine tuning are analyzed. All PID parameters are recalculated on the basis of the findings. PID parameters from fine tuning usually have better master control and disturbance behavior than PID parameters from pretuning.</p> <p data-bbox="319 506 1460 570">PID_Compact automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value.</p> <p data-bbox="319 576 1252 640">PID parameters are backed up before they are recalculated and can be reactivated with sPid_Cmpt.b_LoadBackUp.</p> <p data-bbox="319 646 630 678">Requirements for fine tuning:</p> <ul data-bbox="319 685 1101 876" style="list-style-type: none"> <li data-bbox="319 685 686 717">• No disturbances are expected.</li> <li data-bbox="319 723 1061 755">• The setpoint and the process value lie within the configured limits.</li> <li data-bbox="319 761 917 793">• The setpoint may not be changed during fine tuning.</li> <li data-bbox="319 800 614 832">• ManualEnable = FALSE</li> <li data-bbox="319 838 1101 876">• Automatic (State = 3), inactive (State = 0) or manual (State = 4) mode</li> </ul> <p data-bbox="319 883 837 915">Fine tuning proceeds as follows when started in:</p> <ul data-bbox="319 921 654 953" style="list-style-type: none"> <li data-bbox="319 921 654 953">• Automatic mode (State = 3)</li> </ul> <p data-bbox="319 959 1452 1023">Start fine tuning in automatic mode if you wish to improve the existing PID parameters using controller tuning.</p> <p data-bbox="319 1029 1460 1093">PID_Compact will regulate using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.</p> <ul data-bbox="319 1100 861 1132" style="list-style-type: none"> <li data-bbox="319 1100 861 1132">• Inactive (State = 0) or manual (State = 4) mode</li> </ul> <p data-bbox="319 1138 1460 1287">If the requirements for pretuning are met, pretuning is started. The PID parameters established will be used for adjustment until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. If pretuning is not possible, PID_Compact will change to "Inactive" mode.</p> <p data-bbox="319 1293 1460 1400">An attempt is made to reach the setpoint with a minimum or maximum output value if the process value for pretuning is already too near the setpoint or sPid_Calc.b_RunIn = TRUE. This can produce increased overshoot.</p> <p data-bbox="319 1406 1460 1470">The controller will change to "automatic mode" after successfully completed "fine tuning" and to "inactive" mode if "fine tuning" has not been successfully completed.</p> <p data-bbox="319 1476 1109 1508">The "Fine tuning" phase is indicated with Tag i_Event_TIR V1 (Page 311).</p>

8.1 PID\_Compact

State / sRet.i_Mode	Description of the operating mode
3	<p>Automatic mode</p> <p>In automatic mode, PID_Compact corrects the controlled system in accordance with the parameters specified.</p> <p>The controller changes to automatic mode if one the following conditions is fulfilled:</p> <ul style="list-style-type: none"> <li>• Pretuning successfully completed</li> <li>• Fine tuning successfully completed</li> <li>• Change of variable sRet.i_Mode to the value 3.</li> </ul> <p>After CPU startup or change from Stop to RUN mode, PID_Compact will start in the most recently active operating mode. To retain PID_Compact in "Inactive" mode, set sb_RunModeByStartup = FALSE.</p>
4	<p>Manual mode</p> <p>In manual mode, you specify a manual output value in the ManualValue parameter.</p> <p>This operating mode is enabled if sRet.i_Mode = 4, or at the rising edge on ManualEnable. If ManualEnable changes to TRUE, only State will change. sRet.i_Mode will retain its current value. PID_Compact will return to the previous operating mode upon a falling edge at ManualEnable.</p> <p>The change to automatic mode is bumpless.</p>

See also

Output parameters of PID\_Compact V1 (Page 297)

Pretuning V1 (Page 113)

Fine tuning V1 (Page 115)

"Manual" mode V1 (Page 117)

Tag i\_Event\_SUT V1 (Page 310)

Tag i\_Event\_TIR V1 (Page 311)

### 8.1.5.6 Parameter Error V1

If several errors are pending simultaneously, the values of the error codes are displayed with binary addition. The display of error code 0003, for example, indicates that the errors 0001 and 0002 are pending simultaneously.

Error (DW#16#...)	Description
0000	There is no error.
0001	The "Input" parameter is outside the process value limits. <ul style="list-style-type: none"> <li>• Input &gt; sPid_Cmpt.r_Pv_Hlm or</li> <li>• Input &lt; sPid_Cmpt.r_Pv_Llm</li> </ul> You cannot move the actuator again until you eliminate the error.
0002	Invalid value at "Input_PER" parameter. Check whether an error is pending at the analog input.
0004	Error during fine tuning. Oscillation of the process value could not be maintained.
0008	Error at start of pretuning. The process value is too close to the setpoint. Start fine tuning.
0010	The setpoint was changed during tuning.
0020	Pretuning is not permitted in automatic mode or during fine tuning.
0080	Incorrect configuration of output value limits. Check whether the limits of the output value are configured correctly and match the control logic.
0100	Error during tuning resulted in invalid parameters.
0200	Invalid value at "Input" parameter: Value has an invalid number format.
0400	Calculation of output value failed. Check the PID parameters.
0800	Sampling time error: PID_Compact is not called within the sampling time of the cyclic interrupt OB. If this error occurred during simulation with PLCSIM, see the notes under Simulating PID_Compact V1 with PLCSIM (Page 118).
1000	Invalid value at "Setpoint" parameter: Value has an invalid number format.

**See also**

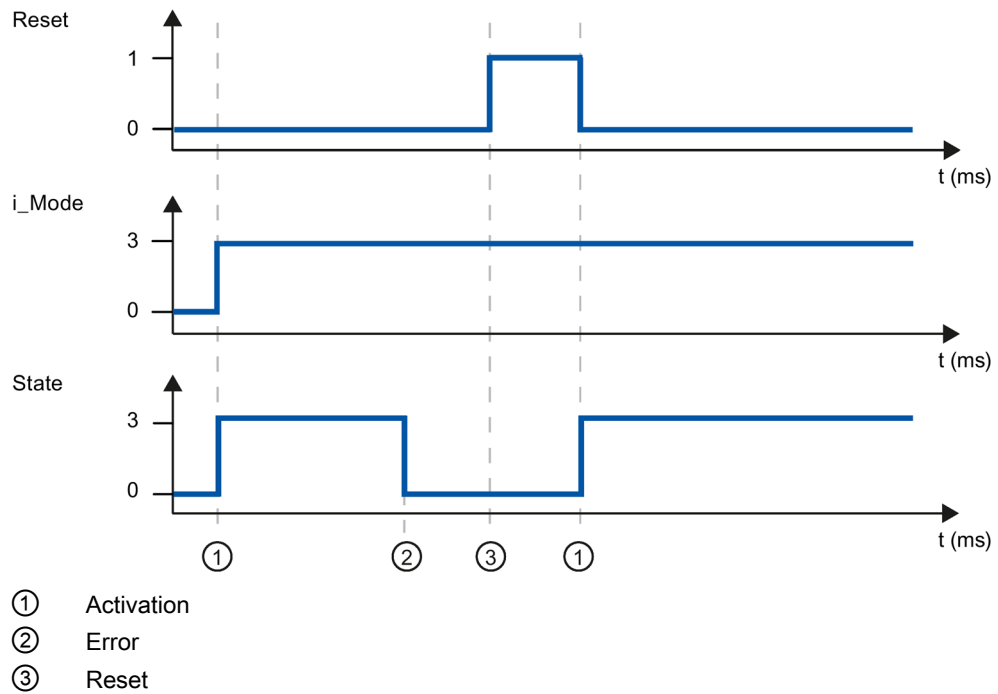
Output parameters of PID\_Compact V1 (Page 297)

### 8.1.5.7 Parameter Reset V1

The response to Reset = TRUE depends on the version of the PID\_Compact instruction.

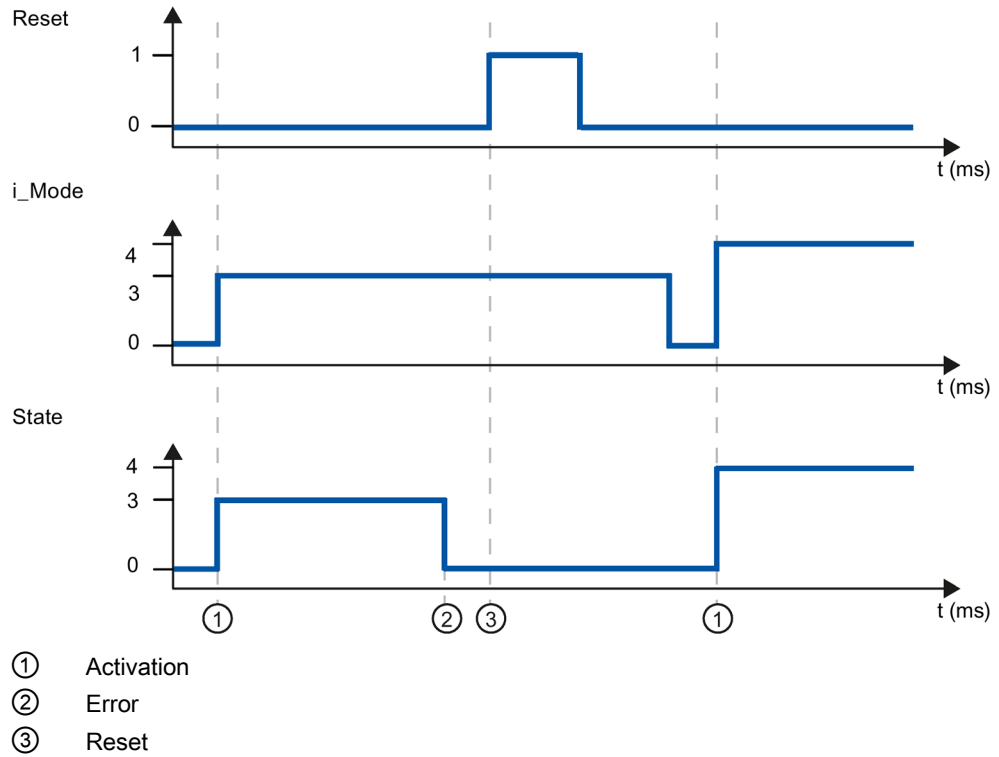
#### Reset response PID\_Compact V.1.1 or higher

A rising edge at Reset triggers a change to "Inactive" mode; errors and warnings are reset and the integral action is deleted. A falling edge at Reset triggers a change to the most recently active operating mode. If automatic mode was active before, the integral action is pre-assigned in such a way that the switchover is bumpless.



### Reset response PID\_Compact V.1.0

A rising edge at Reset triggers a change to "Inactive" mode; errors and warnings are reset and the integral action is deleted. The controller is not reactivated until the next edge at i\_Mode.



**8.1.5.8 Tag sd\_warning V1**

If several warnings are pending, the values of variable sd\_warning are displayed by means of binary addition. The display of warning 0003, for example, indicates that the warnings 0001 and 0002 are also pending.

sd_warning (DW#16#....)	Description
0000	No warning pending.
0001	The point of inflection was not found during pretuning.
0002	Oscillation increased during fine tuning.
0004	The setpoint was outside the set limits.
0008	Not all the necessary controlled system properties were defined for the selected method of calculation. The PID parameters were instead calculated using the "i_CtrlTypeTIR = 3" method.
0010	The operating mode could not be changed because ManualEnable = TRUE.
0020	The cycle time of the calling OB limits the sampling time of the PID algorithm. Improve results by using shorter OB cycle times.
0040	The process value exceeded one of its warning limits.

The following warnings are deleted as soon as the cause is dealt with:

- 0004
- 0020
- 0040

All other warnings are cleared with a rising edge at Reset.

**8.1.5.9 Tag i\_Event\_SUT V1**

i_Event_SUT	Name	Description
0	SUT_INIT	Initialize pretuning
100	SUT_STDABW	Calculate the standard deviation
200	SUT_GET_POI	Find the point of inflection
9900	SUT_IO	Pretuning successful
1	SUT_NIO	Pretuning not successful

**See also**

Static tags of PID\_Compact V1 (Page 298)

Parameters State and sRet.i\_Mode V1 (Page 303)

### 8.1.5.10 Tag i\_Event\_TIR V1

<b>i_Event_TIR</b>	<b>Name</b>	<b>Description</b>
-100	TIR_FIRST_SUT	Fine tuning is not possible. Pretuning will be executed first.
0	TIR_INIT	Initialize fine tuning
200	TIR_STDABW	Calculate the standard deviation
300	TIR_RUN_IN	Attempt to reach the setpoint
400	TIR_CTRLN	Attempt to reach the setpoint with the existing PID parameters (if pretuning has been successful)
500	TIR_OSZIL	Determine oscillation and calculate parameters
9900	TIR_IO	Fine tuning successful
1	TIR_NIO	Fine tuning not successful

#### See also

Static tags of PID\_Compact V1 (Page 298)

Parameters State and sRet.i\_Mode V1 (Page 303)

## 8.2 PID\_3Step

### 8.2.1 New features of PID\_3Step

#### PID\_3Step V2.3

- As of PID\_3Step Version 2.3 the monitoring and limiting of the travel time can be deactivated with `Config.VirtualActuatorLimit = 0.0`.

#### PID\_3Step V2.2

- **Use with S7-1200**

As of PID\_3Step V2.2, the instruction with V2 functionality can also be used on S7-1200 with firmware version 4.0 or higher.

#### PID\_3Step V2.0

- **Reaction to error**

The reaction to `ActivateRecoverMode = TRUE` has been completely overhauled. PID\_3Step reacts in a more fault tolerant manner in the default setting.

<b>NOTICE</b>
<b>Your system may be damaged.</b>
If you use the default setting, PID_3Step remains in automatic mode even if the process value limits are exceeded. This may damage your system.
It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

You use the `ErrorAck` input parameter to acknowledge the errors and warnings without restarting the controller or clearing the integral action.

Switching operating modes does not acknowledge errors that are no longer pending.

- **Switching the operating mode**

You specify the operating mode at the `Mode in/out` parameter and use a positive edge at `ModeActivate` to start the operating mode. The `Retain.Mode` tag has been omitted.

The transition time measurement can no longer be started with `GetTransitTime.Start`, but only with `Mode = 6` and a positive edge at `ModeActivate`.

- **Multi-instance capability**

You can call up PID\_3Step as multi-instance DB. No technology object is created in this case and no parameter assignment interface or commissioning interface is available. You must assign parameters for PID\_3Step directly in the multi-instance DB and commission it via a watch table.



- **Startup characteristics**

The operating mode specified at the Mode parameter is also started on a negative edge at Reset and during a CPU cold restart, if RunModeByStartup = TRUE.
- **ENO characteristics**

ENO is set depending on the operating mode.  
If State = 0, then ENO = FALSE.  
If State ≠ 0, then ENO = TRUE.
- **Manual mode**

The Manual\_UP and Manual\_DN input parameters no longer function as edge-triggered parameters. Edge-triggered manual mode continues to be possible using the ManualUpInternal and ManualDnInternal tags.  
In "Manual mode without endstop signals" (Mode = 10), the endstop signals Actuator\_H and Actuator\_L are ignored even though they are activated.
- **Default value of PID parameters**

The following default settings have been changed:

  - Proportional action weighting (PWeighting) from 0.0 to 1.0
  - Derivative action weighting (DWeighting) from 0.0 to 1.0
  - Coefficient for derivative delay (TdFiltRatio) from 0.0 to 0.2
- **Limiting of motor transition time**

You configure the maximum percentage of the motor transition time that the actuator will travel in one direction in the Config.VirtualActuatorLimit tag.
- **Setpoint value specification during tuning**

You configure the permitted fluctuation of the setpoint during tuning at the CancelTuningLevel tag.
- **Switching a disturbance variable on**

You can switch a disturbance variable on at the Disturbance parameter.
- **Troubleshooting**

If the endstop signals are not activated (ActuatorEndStopOn = FALSE), ScaledFeedback is determined without Actuator\_H or Actuator\_L.

## PID\_3Step V1.1

- **Manual mode on CPU startup**

If ManualEnable = TRUE when the CPU starts, PID\_3Step starts in manual mode. A positive edge at ManualEnable is not necessary.
- **Reaction to error**

The ActivateRecoverMode tag is no longer effective in manual mode.
- **Troubleshooting**

The Progress tag is reset following successful tuning or transition time measurement.

### 8.2.2 Compatibility with CPU and FW

The following table shows which version of PID\_3Step can be used on which CPU.

CPU	FW	PID_3Step
S7-1200	V4.2 or higher	V2.3 V2.2 V1.1
	V4.0 to V4.1	V2.2 V1.1
	V3.x	V1.1 V1.0
	V2.x	V1.1 V1.0
	V1.x	-
S7-1500	V2.0 or higher	V2.3 V2.2 V2.1 V2.0
	V1.5 to V1.8	V2.2 V2.1 V2.0
	V1.1	V2.1 V2.0
	V1.0	V2.0

## 8.2.3 CPU processing time and memory requirement PID\_3Step V2.x

### CPU processing time

Typical CPU processing times of the PID\_3Step technology object as of Version V2.0, depending on CPU type.

CPU	Typ. CPU processing time PID_3Step V2.x
CPU 1211C ≥ V4.0	410 μs
CPU 1215C ≥ V4.0	410 μs
CPU 1217C ≥ V4.0	410 μs
CPU 1505S ≥ V1.0	50 μs
CPU 1510SP-1 PN ≥ V1.6	120 μs
CPU 1511-1 PN ≥ V1.5	120 μs
CPU 1512SP-1 PN ≥ V1.6	120 μs
CPU 1516-3 PN/DP ≥ V1.5	65 μs
CPU 1518-4 PN/DP ≥ V1.5	5 μs

### Memory requirement

Memory requirement of an instance DB of the PID\_3Step technology object as of Version V2.0.

	Memory requirement of the instance DB of PID_3Step V2.x
Load memory requirement	Approx. 15000 bytes
Total work memory requirement	1040 bytes
Retentive work memory requirement	60 bytes

## 8.2.4 PID\_3Step V2

### 8.2.4.1 Description of PID\_3Step V2

#### Description

You use the PID\_3Step instruction to configure a PID controller with self tuning for valves or actuators with integrating behavior.

The following operating modes are possible:

- Inactive
- Pretuning
- Fine tuning
- Automatic mode
- Manual mode
- Approach substitute output value
- Transition time measurement
- Error monitoring
- Approach substitute output value with error monitoring
- Manual mode without endstop signals

For a more detailed description of the operating modes, see the State parameter.

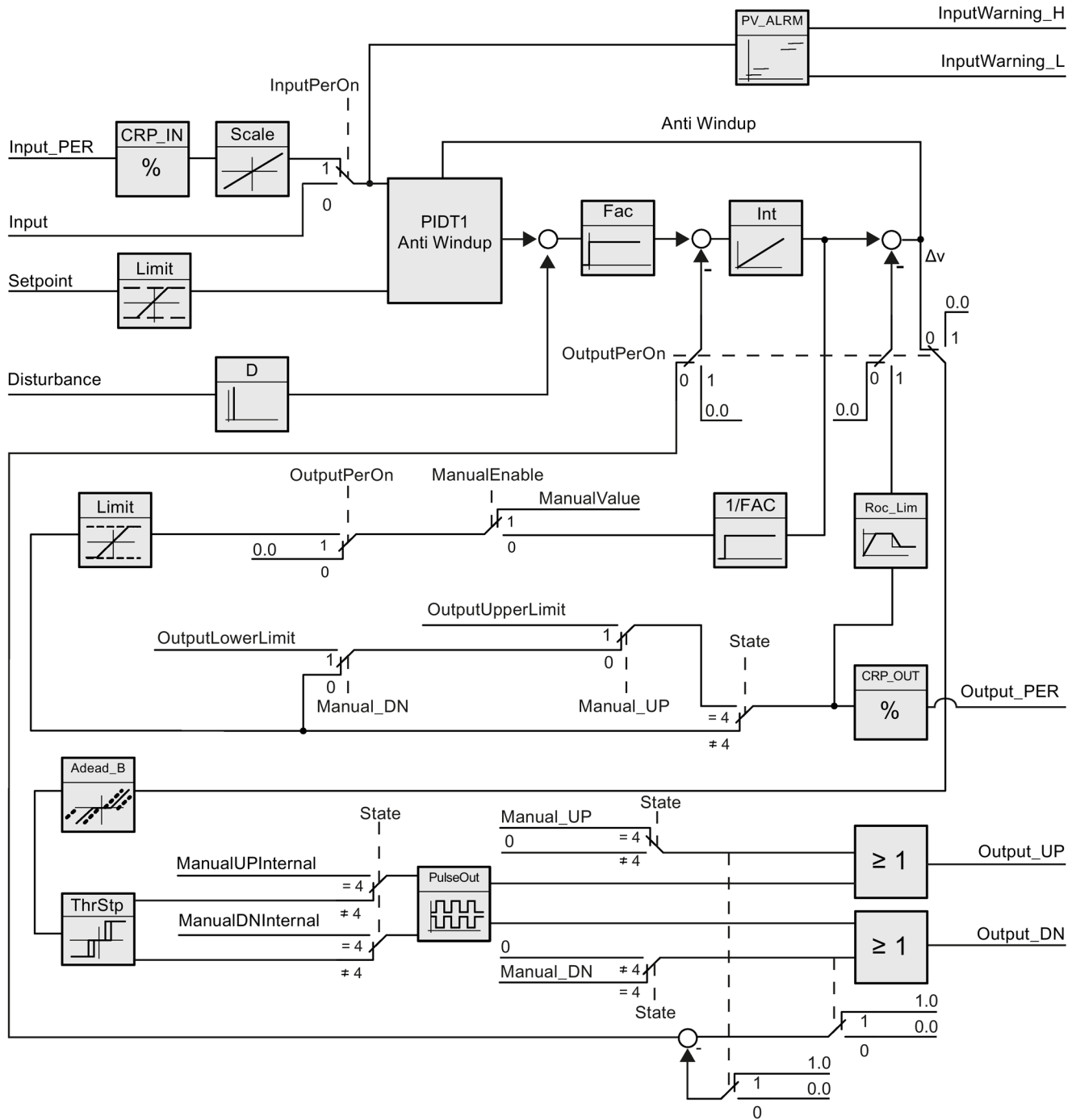
## PID algorithm

PID\_3Step is a PIDT1 controller with anti-windup and weighting of the proportional and derivative actions. The PID algorithm operates according to the following equation:

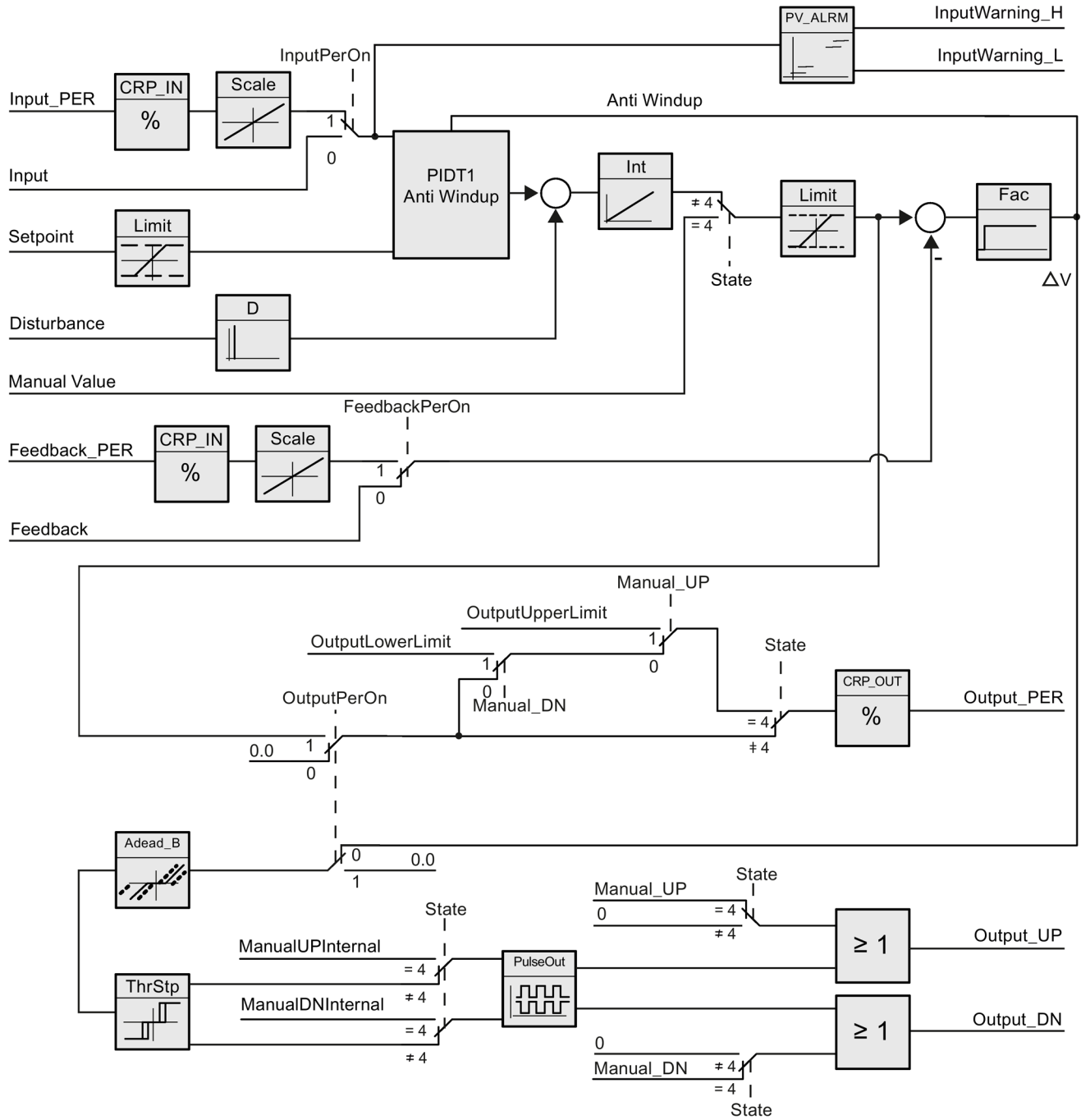
$$\Delta y = K_p \cdot s \cdot \left[ (b \cdot w - x) + \frac{1}{T_i \cdot s} (w - x) + \frac{T_d \cdot s}{a \cdot T_d \cdot s + 1} (c \cdot w - x) \right]$$

<b>Symbol</b>	<b>Description</b>
$\Delta y$	Output value of the PID algorithm
$K_p$	Proportional gain
$s$	Laplace operator
$b$	Proportional action weighting
$w$	Setpoint
$x$	Process value
$T_i$	Integral action time
$T_d$	Derivative action time
$a$	Derivative delay coefficient (derivative delay $T_1 = a \times T_d$ )
$c$	Derivative action weighting

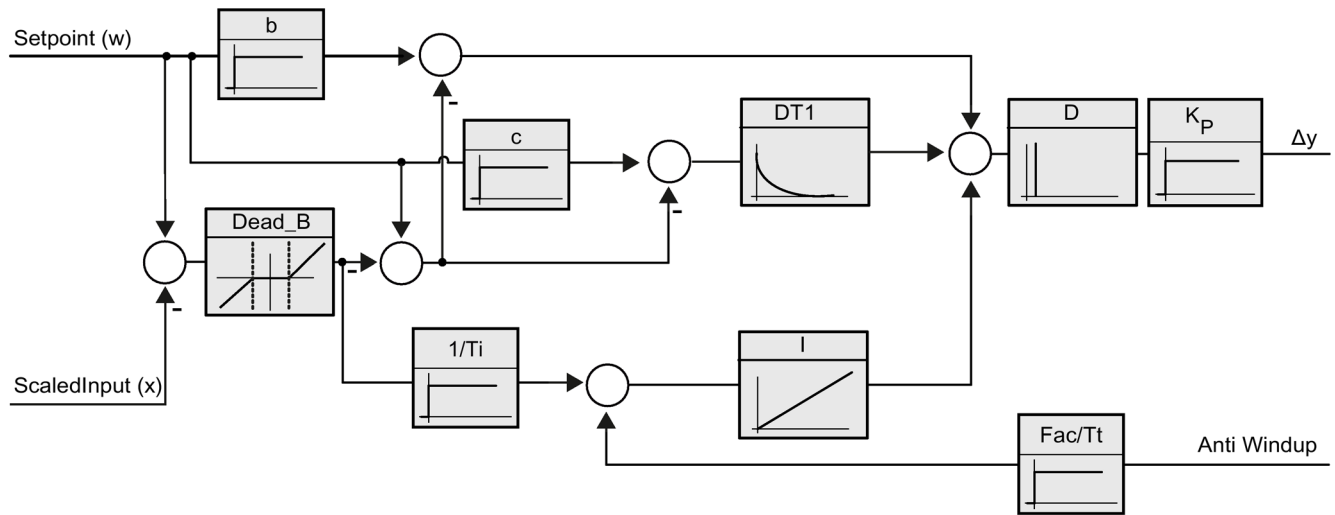
Block diagram without position feedback



Block diagram with position feedback



**Block diagram of PIDT1 with anti-windup**



**Call**

PID\_3Step is called in the constant time scale of a cycle interrupt OB.

If you call PID\_3Step as a multi-instance DB, no technology object is created. No parameter assignment interface or commissioning interface is available. You must assign parameters for PID\_3Step directly in the multi-instance DB and commission it via a watch table.

**Download to device**

The actual values of retentive tags are only updated when you download PID\_3Step completely.

Downloading technology objects to device (Page 72)

**Startup**

When the CPU starts up, PID\_3Step starts in the operating mode that is saved in the Mode in/out parameter. To leave PID\_3Step in "Inactive" mode, set RunModeByStartup = FALSE.



## Reaction to error

In automatic mode and during commissioning, the reaction to error depends on the ErrorBehaviour and ActivateRecoverMode tags. In manual mode, the reaction is independent of ErrorBehaviour and ActivateRecoverMode. If ActivateRecoverMode = TRUE, the reaction additionally depends on the error that occurred.

ErrorBehaviour	ActivateRecoverMode	Configuration editor > actuator setting > Set Output to	Reaction
FALSE	FALSE	Current output value	Switch to "Inactive" mode (State = 0) The actuator remains in the current position.
FALSE	TRUE	Current output value while error is pending	Switch to "Error monitoring" mode (State = 7) The actuator remains in the current position while the error is pending.
TRUE	FALSE	Substitute output value	Switch to "Approach substitute output value" mode (State = 5) The actuator moves to the configured substitute output value. Switch to "Inactive" mode (State = 0) The actuator remains in the current position.
TRUE	TRUE	Substitute output value while error is pending	Switch to "Approach substitute output value with error monitoring" mode (State = 8) The actuator moves to the configured substitute output value. Switch to "Error monitoring" mode (State = 7)

In manual mode, PID\_3Step uses ManualValue as output value, unless the following errors occur:

- 2000h: Invalid value at Feedback\_PER parameter.
- 4000h: Invalid value at Feedback parameter.
- 8000h: Error during digital position feedback.

You can only change the position of the actuator with Manual\_UP and Manual\_DN, not with ManualValue.

The Error parameter indicates whether an error has occurred in this cycle. The ErrorBits parameter shows which errors have occurred. ErrorBits is reset by a rising edge at Reset or ErrorAck.

## See also

Parameters State and Mode V2 (Page 342)

Parameter ErrorBits V2 (Page 347)

Configuring PID\_3Step V2 (Page 121)

### 8.2.4.2 Mode of operation of PID\_3Step V2

#### Monitoring process value limits

You specify the high limit and low limit of the process value in the Config.InputUpperLimit and Config.InputLowerLimit tags. If the process value is outside these limits, an error occurs (ErrorBits = 0001h).

You specify a high and low warning limit of the process value in the Config.InputUpperWarning and Config.InputLowerWarning tags. If the process value is outside these warning limits, a warning occurs (Warning = 0040h), and the InputWarning\_H or InputWarning\_L output parameter changes to TRUE.

#### Limiting the setpoint

You specify a high limit and low limit of the setpoint in the Config.SetpointUpperLimit and Config.SetpointLowerLimit tags. PID\_3Step automatically limits the setpoint to the process value limits. You can limit the setpoint to a smaller range. PID\_3Step checks whether this range falls within the process value limits. If the setpoint is outside these limits, the high or low limit is used as the setpoint, and output parameter SetpointLimit\_H or SetpointLimit\_L is set to TRUE.

The setpoint is limited in all operating modes.

#### Limiting the output value

You specify a high limit and low limit of the output value in the Config.OutputUpperLimit and Config.OutputLowerLimit tags. The output value limits must be within "Low endstop" and "High endstop".

- High endstop: Config.FeedbackScaling.UpperPointOut
- Low endstop: Config.FeedbackScaling.LowerPointOut

Rule:

$$\text{UpperPointOut} \geq \text{OutputUpperLimit} > \text{OutputLowerLimit} \geq \text{LowerPointOut}$$

The valid values for "High endstop" and "Low endstop" depend upon:

- FeedbackOn
- FeedbackPerOn
- OutputPerOn

OutputPerOn	FeedbackOn	FeedbackPerOn	LowerPointOut	UpperPointOut
FALSE	FALSE	FALSE	Cannot be set (0.0%)	Cannot be set (100.0%)
FALSE	TRUE	FALSE	-100.0% or 0.0%	0.0% or +100.0%
FALSE	TRUE	TRUE	-100.0% or 0.0%	0.0% or +100.0%
TRUE	FALSE	FALSE	Cannot be set (0.0%)	Cannot be set (100.0%)
TRUE	TRUE	FALSE	-100.0% or 0.0%	0.0% or +100.0%
TRUE	TRUE	TRUE	-100.0% or 0.0%	0.0% or +100.0%

If OutputPerOn = FALSE and FeedbackOn = FALSE, you cannot limit the output value. Output\_UP and Output\_DN are then reset upon Actuator\_H = TRUE or Actuator\_L = TRUE. If endstop signals are also not present, Output\_UP and Output\_DN are reset after a travel time of Config.VirtualActuatorLimit × Retain.TransitTime/100. As of PID\_3Step Version 2.3 the monitoring and limiting of the travel time can be deactivated with Config.VirtualActuatorLimit = 0.0.

The output value is 27648 at 100% and -27648 at -100%. PID\_3Step must be able to completely close the valve.

---

#### **Note**

##### **Use with two or more actuators**

PID\_3 Step is not suitable for use with two or more actuators (for example, in heating/cooling applications), because different actuators need different PID parameters to achieve a good control response.

---

#### **Substitute output value**

If an error has occurred, PID\_3Step can output a substitute output value and move the actuator to a safe position that is specified in the SavePosition tag. The substitute output value must be within the output value limits.

#### **Monitoring signal validity**

The values of the following parameters are monitored for validity when used:

- Setpoint
- Input
- Input\_PER
- Input\_PER
- Feedback
- Feedback\_PER
- Disturbance
- ManualValue
- SavePosition
- Output\_PER

### Monitoring the PID\_3Step sampling time

Ideally, the sampling time is equivalent to the cycle time of the calling OB. The PID\_3Step instruction measures the time interval between two calls. This is the current sampling time. On every switchover of operating mode and during the initial startup, the mean value is formed from the first 10 sampling times. Too great a difference between the current sampling time and this mean value triggers an error (ErrorBits = 0800h).

The error occurs during tuning if:

- New mean value  $\geq 1.1 \times$  old mean value
- New mean value  $\leq 0.9 \times$  old mean value

The error occurs in automatic mode if:

- New mean value  $\geq 1.5 \times$  old mean value
- New mean value  $\leq 0.5 \times$  old mean value

If you deactivate the sampling time monitoring (CycleTime.EnMonitoring = FALSE), you can also call PID\_3Step in OB1. You must then accept a lower control quality due to the deviating sampling time.

### Sampling time of the PID algorithm

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the cycle time. All other functions of PID\_3Step are executed at every call.

### Measuring the motor transition time

The motor transition time is the time in seconds the motor requires to move the actuator from the closed to the opened state. The actuator is moved in one direction for a maximum time of  $\text{Config.VirtualActuatorLimit} \times \text{Retain.TransitTime}/100$ . PID\_3Step requires the motor transition time to be as accurate as possible for good controller results. The data in the actuator documentation contains average values for this type of actuator. The value for the specific actuator used may differ. You can measure the motor transition time during commissioning. The output value limits are not taken into consideration during the motor transition time measurement. The actuator can travel to the high or the low endstop.

The motor transition time is taken into consideration in the calculation of the analog output value as well as in the calculation of the digital output values. It is mainly required for correct operation during auto-tuning and the anti-windup behavior. You should therefore configure the motor transition time with the value that the motor requires to move the actuator from the closed to the opened state.

If no relevant motor transition time is in effect in your process (e.g. with solenoid valves), so that the output value has a direct and full effect on the process, use PID\_Compact instead.

## **Control logic**

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic. For cooling and discharge control systems, it may be necessary to invert the control logic. PID\_3Step does not work with negative proportional gain. If InvertControl = TRUE, an increasing control deviation causes a reduction in the output value. The control logic is also taken into account during pretuning and fine tuning.

## **See also**

Configuring PID\_3Step V1 (Page 144)

**8.2.4.3 Changing the PID\_3Step V2 interface**

The following table shows what has changed in the PID\_3Step instruction interface.

PID_3Step V1	PID_3Step V2	Change
Input_PER	Input_PER	Data type from Word to Int
Feedback_PER	Feedback_PER	Data type from Word to Int
	Disturbance	New
Manual_UP	Manual_UP	Function
Manual_DN	Manual_DN	Function
	ErrorAck	New
	ModeActivate	New
Output_PER	Output_PER	Data type from Word to Int
	ManualUPInternal	New
	ManualDNInternal	New
	CancelTuningLevel	New
	VirtualActuatorLimit	New
Config.Loadbackup	Loadbackup	Renamed
Config.TransitTime	Retain.TransitTime	Renamed and retentivity added
GetTransitTime.Start		Replaced by Mode and ModeActivate
SUT.CalculateSUTParams	SUT.CalculateParams	Renamed
SUT.TuneRuleSUT	SUT.TuneRule	Renamed
TIR.CalculateTIRParams	TIR.CalculateParams	Renamed
TIR.TuneRuleTIR	TIR.TuneRule	Renamed
Retain.Mode	Mode	Function Declaration of static for in-out parameters

### 8.2.4.4 Input parameters of PID\_3Step V2

Table 8- 7

Parameter	Data type	Default	Description
Setpoint	REAL	0.0	Setpoint of the PID controller in automatic mode
Input	REAL	0.0	A tag of the user program is used as source for the process value. If you are using parameter Input, then Config.InputPerOn = FALSE must be set.
Input_PER	INT	0	An analog input is used as the source of the process value. If you are using parameter Input_PER, then Config.InputPerOn = TRUE must be set.
Actuator_H	BOOL	FALSE	Digital position feedback of the valve for the high endstop If Actuator_H = TRUE, the valve is at the high endstop and is no longer moved towards this direction.
Actuator_L	BOOL	FALSE	Digital position feedback of the valve for the low endstop If Actuator_L = TRUE, the valve is at the low endstop and is no longer moved towards this direction.
Feedback	REAL	0.0	Position feedback of the valve If you are using parameter Feedback, then Config.FeedbackPerOn = FALSE must be set.
Feedback_PER	INT	0	Analog position feedback of a valve If you are using parameter Feedback_PER, then Config.FeedbackPerOn = TRUE must be set. Feedback_PER is scaled based on the tags: <ul style="list-style-type: none"> <li>• Config.FeedbackScaling.LowerPointIn</li> <li>• Config.FeedbackScaling.UpperPointIn</li> <li>• Config.FeedbackScaling.LowerPointOut</li> <li>• Config.FeedbackScaling.UpperPointOut</li> </ul>
Disturbance	REAL	0.0	Disturbance variable or precontrol value
ManualEnable	BOOL	FALSE	<ul style="list-style-type: none"> <li>• A FALSE -&gt; TRUE edge activates "manual mode", while State = 4, Mode remains unchanged.</li> </ul> <p>As long as ManualEnable = TRUE, you cannot change the operating mode via a rising edge at ModeActivate or use the commissioning dialog.</p> <ul style="list-style-type: none"> <li>• A TRUE -&gt; FALSE edge activates the operating mode that is specified by Mode.</li> </ul> <p>We recommend that you change the operating mode using ModeActivate only.</p>
ManualValue	REAL	0.0	In manual mode, the absolute position of the valve is specified. ManualValue is only evaluated if you are using Output_PER, or if position feedback is available.

Parameter	Data type	Default	Description
Manual_UP	BOOL	FALSE	<ul style="list-style-type: none"> <li>Manual_UP = TRUE</li> </ul> <p>The valve is opened even if you are using Output_PER or a position feedback. The valve is no longer moved if the high endstop has been reached.</p> <p>See also Config.VirtualActuatorLimit</p> <ul style="list-style-type: none"> <li>Manual_UP = FALSE</li> </ul> <p>If you are using Output_PER or a position feedback, the valve is moved to ManualValue. Otherwise, the valve is no longer moved.</p> <p>If Manual_UP and Manual_DN are set to TRUE simultaneously, the valve is not moved.</p>
Manual_DN	BOOL	FALSE	<ul style="list-style-type: none"> <li>Manual_DN = TRUE</li> </ul> <p>The valve is closed even if you are using Output_PER or a position feedback. The valve is no longer moved if the low endstop has been reached.</p> <p>See also Config.VirtualActuatorLimit</p> <ul style="list-style-type: none"> <li>Manual_DN = FALSE</li> </ul> <p>If you are using Output_PER or a position feedback, the valve is moved to ManualValue. Otherwise, the valve is no longer moved.</p>
ErrorAck	BOOL	FALSE	<ul style="list-style-type: none"> <li>FALSE -&gt; TRUE edge</li> </ul> <p>ErrorBits and Warning are reset.</p>
Reset	BOOL	FALSE	<p>Restarts the controller.</p> <ul style="list-style-type: none"> <li>FALSE -&gt; TRUE edge                             <ul style="list-style-type: none"> <li>Switch to "Inactive" mode</li> <li>ErrorBits and Warnings are reset.</li> </ul> </li> <li>As long as Reset = TRUE,                             <ul style="list-style-type: none"> <li>PID_3Step remains in "Inactive" mode (State = 0).</li> <li>You cannot change the operating mode with Mode and ModeActivate or ManualEnable.</li> <li>You cannot use the commissioning dialog.</li> </ul> </li> <li>TRUE -&gt; FALSE edge                             <ul style="list-style-type: none"> <li>If ManualEnable = FALSE, PID_3Step switches to the operating mode that is saved in Mode.</li> <li>If Mode = 3, the integral action is treated as configured with the tag IntegralResetMode.</li> </ul> </li> </ul>
ModeActivate	BOOL	FALSE	<ul style="list-style-type: none"> <li>FALSE -&gt; TRUE edge</li> </ul> <p>PID_3Step switches to the operating mode that is saved in the Mode parameter.</p>



### 8.2.4.5 Output parameters of PID\_3Step V2

Table 8- 8

Parameter	Data type	Default	Description
ScaledInput	REAL	0.0	Scaled process value
ScaledFeedback	REAL	0.0	Scaled position feedback For an actuator without position feedback, the position of the actuator indicated by ScaledFeedback is very imprecise. ScaledFeedback may only be used for rough estimation of the current position in this case.
Output_UP	BOOL	FALSE	Digital output value for opening the valve If Config.OutputPerOn = FALSE, the Output_UP parameter is used.
Output_DN	BOOL	FALSE	Digital output value for closing the valve If Config.OutputPerOn = FALSE, the Output_DN parameter is used.
Output_PER	INT	0	Analog output value If Config.OutputPerOn = TRUE, Output_PER is used. Use Output_PER if you are using a valve as actuator which is triggered via an analog output and controlled with a continuous signal, e.g. 0...10 V or 4...20 mA. The value at Output_PER corresponds to the target position of the valve, e.g. Output_PER = 13824, when the valve is to be opened by 50%.
SetpointLimit_H	BOOL	FALSE	If SetpointLimit_H = TRUE, the absolute setpoint high limit is reached (Setpoint ≥ Config.SetpointUpperLimit). The setpoint is limited to Config.SetpointUpperLimit .
SetpointLimit_L	BOOL	FALSE	If SetpointLimit_L = TRUE, the absolute setpoint low limit has been reached (Setpoint ≤ Config.SetpointLowerLimit). The setpoint is limited to Config.SetpointLowerLimit .
InputWarning_H	BOOL	FALSE	If InputWarning_H = TRUE, the process value has reached or exceeded the warning high limit.
InputWarning_L	BOOL	FALSE	If InputWarning_L = TRUE, the process value has reached or fallen below the warning low limit.
State	INT	0	The State parameter (Page 342) shows the current operating mode of the PID controller. You can change the operating mode using the input parameter Mode and a rising edge at ModeActivate. <ul style="list-style-type: none"> <li>• State = 0: Inactive</li> <li>• State = 1: Pretuning</li> <li>• State = 2: Fine tuning</li> <li>• State = 3: Automatic mode</li> <li>• State = 4: Manual mode</li> <li>• State = 5: Approach substitute output value</li> <li>• State = 6: Transition time measurement</li> <li>• State = 7: Error monitoring</li> <li>• State = 8: Approach substitute output value with error monitoring</li> <li>• State = 10: Manual mode without endstop signals</li> </ul>

Parameter	Data type	Default	Description
Error	BOOL	FALSE	If Error = TRUE, at least one error message is pending in this cycle.
ErrorBits	DWORD	DW#16#0	The ErrorBits parameter (Page 347) shows which error messages are pending. ErrorBits is retentive and is reset upon a rising edge at Reset or ErrorAck.

**See also**

Parameters State and Mode V2 (Page 342)

Parameter ErrorBits V2 (Page 347)

**8.2.4.6 In/out parameters of PID-3Step V2**

Table 8- 9

Parameter	Data type	Default	Description
Mode	INT	4	<p>At the Mode parameter, you specify the operating mode to which PID_3Step is to switch. Options are:</p> <ul style="list-style-type: none"> <li>• Mode = 0: Inactive</li> <li>• Mode = 1: Pretuning</li> <li>• Mode = 2: Fine tuning</li> <li>• Mode = 3: Automatic mode</li> <li>• Mode = 4: Manual mode</li> <li>• Mode = 6: Transition time measurement</li> <li>• Mode = 10: Manual mode without endstop signals</li> </ul> <p>The operating mode is activated by:</p> <ul style="list-style-type: none"> <li>• Rising edge at ModeActivate</li> <li>• Falling edge at Reset</li> <li>• Falling edge at ManualEnable</li> <li>• Cold restart of CPU if RunModeByStartup = TRUE</li> </ul> <p>Mode is retentive.</p> <p>A detailed description of the operating modes can be found in Parameters State and Mode V2 (Page 342).</p>

### 8.2.4.7 Static tags of PID\_3Step V2

You must not change tags that are not listed. These are used for internal purposes only.

Tag	Data type	Default	Description
ManualUpInternal	BOOL	FALSE	In manual mode, each rising edge opens the valve by 5% of the total control range or for the duration of the minimum motor transition time. ManualUpInternal is only evaluated if you are not using Output_PER or a position feedback. This tag is used in the commissioning dialog.
ManualDnInternal	BOOL	FALSE	In manual mode, every rising edge closes the valve by 5% of the total control range or for the duration of the minimum motor transition time. ManualDnInternal is only evaluated if you are not using Output_PER or position feedback. This tag is used in the commissioning dialog.
ActivateRecoverMode	BOOL	TRUE	The ActivateRecoverMode V2 (Page 350) tag determines the reaction to error.
RunModeByStartup	BOOL	TRUE	Activate operating mode at Mode parameter after CPU restart If RunModeByStartup = TRUE, PID_3Step starts in the operating mode saved in the Mode parameter after CPU startup. If RunModeByStartup = FALSE, PID_3Step remains in "Inactive" mode after CPU startup.
LoadBackUp	BOOL	FALSE	If LoadBackUp = TRUE, the last set of PID parameters is reloaded. The set was saved prior to the last tuning. LoadBackUp is automatically set back to FALSE.
PhysicalUnit	INT	0	Unit of measurement of the process value and setpoint, e.g., °C, or °F.
PhysicalQuantity	INT	0	Physical quantity of the process value and setpoint, e.g., temperature

Tag	Data type	Default	Description
ErrorBehaviour	BOOL	FALSE	<p>If ErrorBehaviour = FALSE and an error has occurred, the valve stays at its current position and the controller switches directly to "Inactive" or "Error monitoring" mode.</p> <p>If ErrorBehaviour = TRUE and an error occurs, the actuator moves to the substitute output value and only then switches to "Inactive" or "Error monitoring" mode.</p> <p>If the following errors occur, you can no longer move the valve to a configured substitute output value.</p> <ul style="list-style-type: none"> <li>• 2000h: Invalid value at Feedback_PER parameter.</li> <li>• 4000h: Invalid value at Feedback parameter.</li> <li>• 8000h: Error during digital position feedback.</li> <li>• 20000h: Invalid value at SavePosition tag.</li> </ul>
Warning	DWORD	DW#16#0	<p>The Warning tag (Page 342) shows the warnings since Reset = TRUE or ErrorAck =TRUE. Warning is retentive.</p> <p>Cyclic warnings (for example, process value warning) are shown until the cause of the warning is removed. They are automatically deleted once their cause has gone. Non-cyclic warnings (for example, point of inflection not found) remain and are deleted like errors.</p>
SavePosition	REAL	0.0	<p>Substitute output value</p> <p>If ErrorBehaviour = TRUE, the actuator is moved to a position that is safe for the plant when an error occurs. As soon as the substitute output value has been reached, PID_3Step switches the operating mode according to ActivateRecoverMode.</p>
CurrentSetpoint	REAL	0.0	<p>Currently active setpoint. This value is frozen at the start of tuning.</p>
CancelTuningLevel	REAL	10.0	<p>Permissible fluctuation of setpoint during tuning. Tuning is not canceled until:</p> <ul style="list-style-type: none"> <li>• Setpoint &gt; CurrentSetpoint + CancelTuningLevel</li> <li>or</li> <li>• Setpoint &lt; CurrentSetpoint - CancelTuningLevel</li> </ul>
Progress	REAL	0.0	<p>Progress of tuning as a percentage (0.0 - 100.0)</p>
Config.InputPerOn	BOOL	TRUE	<p>If InputPerOn = TRUE, the Input_PER parameter is used. If InputPerOn = FALSE, the Input parameter is used.</p>

Tag	Data type	Default	Description
Config.OutputPerOn	BOOL	FALSE	If OutputPerOn = TRUE, the Output_PER parameter is used. If OutputPerOn = FALSE, the Ouput_UP and Output_DN parameters are used.
Config.InvertControl	BOOL	FALSE	Invert control logic If InvertControl = TRUE, an increasing control deviation causes a reduction in the output value.
Config.FeedbackOn	BOOL	FALSE	If FeedbackOn = FALSE, a position feedback is simulated. Position feedback is generally activated when FeedbackOn = TRUE.
Config.FeedbackPerOn	BOOL	FALSE	FeedbackPerOn is only effective when FeedbackOn = TRUE. If FeedbackPerOn = TRUE, the analog input is used for the position feedback (Feedback_PER parameter). If FeedbackPerOn = FALSE, the Feedback parameter is used for the position feedback.
Config.ActuatorEndStopOn	BOOL	FALSE	If ActuatorEndStopOn = TRUE, the digital position feedback Actuator_L and Actuator_H are taken into consideration.
Config.InputUpperLimit	REAL	120.0	High limit of the process value Input and Input_PER are monitored to ensure adherence to this limit. At the I/O input, the process value can be a maximum of 18% higher than the standard range (overrange). An error is no longer signaled due to a violation of the "Process value high limit". Only a wire-break and a short-circuit are recognized and PID_3Step reacts according to the configured reaction to error. InputUpperLimit > InputLowerLimit
Config.InputLowerLimit	REAL	0.0	Low limit of the process value InputLowerLimit < InputUpperLimit
Config.InputUpperWarning	REAL	+3.402822e+38	Warning high limit of the process value If you set InputUpperWarning outside the process value limits, the configured absolute process value high limit is used as the warning high limit. If you configure InputUpperWarning within the process value limits, this value is used as the warning high limit. InputUpperWarning > InputLowerWarning InputUpperWarning ≤ InputUpperLimit

Tag	Data type	Default	Description
Config.InputLowerWarning	REAL	-3.402822e+38	Warning low limit of the process value If you set InputLowerWarning outside the process value limits, the configured absolute process value low limit is used as the warning low limit. If you configure InputLowerWarning within the process value limits, this value is used as the warning low limit. InputLowerWarning < InputUpperWarning InputLowerWarning ≥ InputLowerLimit
Config.OutputUpperLimit	REAL	100.0	High limit of output value For details, see OutputLowerLimit
Config.OutputLowerLimit	REAL	0.0	Low limit of output value If OutputPerOn = TRUE or FeedbackOn = TRUE, the range of values from -100% to +100%, including zero, is valid. At -100%, Output = -27648; at +100% Output = 27648 If OutputPerOn = FALSE, the range of values from 0% to 100% is valid. The valve is completely closed at 0% and completely opened at 100%.
Config.SetpointUpperLimit	REAL	+3.402822e+38	High limit of setpoint If you set SetpointUpperLimit outside the process value limits, the configured absolute process value high limit is preassigned as the setpoint high limit. If you configure SetpointUpperLimit within the process value limits, this value is used as the setpoint high limit.
Config.SetpointLowerLimit	REAL	- 3.402822e+38	Low limit of the setpoint If you set SetpointLowerLimit outside the process value limits, the configured absolute process value low limit is preassigned as the setpoint low limit. If you set SetpointLowerLimit within the process value limits, this value is used as the setpoint low limit.
Config.MinimumOnTime	REAL	0.0	Minimum ON time Minimum time in seconds for which the servo drive must be switched on. Config.MinimumOnTime is only effective if Output_UP and Output_DN are used (Config.OutputPerOn = FALSE).

Tag	Data type	Default	Description
Config.MinimumOffTime	REAL	0.0	Minimum OFF time Minimum time in seconds for which the servo drive must be switched off. Config.MinimumOffTime is only effective if Output_UP and Output_DN are used (Config.OutputPerOn = FALSE).
Config.VirtualActuatorLimit	REAL	150.0	If all the following conditions have been satisfied, the actuator is moved in one direction for the maximum period of VirtualActuatorLimit × Retain.TransitTime/100 and the warning 2000h is output: <ul style="list-style-type: none"> <li>• Config.OutputPerOn = FALSE</li> <li>• Config.ActuatorEndStopOn = FALSE</li> <li>• Config.FeedbackOn = FALSE</li> </ul> If Config.OutputPerOn = FALSE and Config.ActuatorEndStopOn = TRUE or Config.FeedbackOn = TRUE, only the warning 2000h is output. If Config.OutputPerOn = TRUE, VirtualActuatorLimit is not taken into consideration. As of PID_3Step Version 2.3 the monitoring and limiting of the travel time can be deactivated with Config.VirtualActuatorLimit = 0.0.
Config.InputScaling.UpperPointIn	REAL	27648.0	Scaling Input_PER high Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.InputScaling.LowerPointIn	REAL	0.0	Scaling Input_PER low Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.InputScaling.UpperPointOut	REAL	100.0	Scaled high process value Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.InputScaling.LowerPointOut	REAL	0.0	Scaled low process value Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.FeedbackScaling.UpperPointIn	REAL	27648.0	Scaling Feedback_PER high Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.

Tag	Data type	Default	Description
Config.FeedbackScaling.LowerPointIn	REAL	0.0	Scaling Feedback_PER low Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.
Config.FeedbackScaling.UpperPointOut	REAL	100.0	High endstop Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.
Config.FeedbackScaling.LowerPointOut	REAL	0.0	Low endstop Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.
GetTransitTime.InvertDirection	BOOL	FALSE	If InvertDirection = FALSE, the valve is fully opened, closed, and then reopened in order to determine the valve transition time. If InvertDirection = TRUE, the valve is fully closed, opened, and then closed again.
GetTransitTime.SelectFeedback	BOOL	FALSE	If SelectFeedback = TRUE, then Feedback_PER, or Feedback is taken into consideration in the transition time measurement. If SelectFeedback = FALSE, then Actuator_H and Actuator_L are taken into consideration in the transition time measurement.
GetTransitTime.State	INT	0	Current phase of the transition time measurement <ul style="list-style-type: none"> <li>• State = 0: Inactive</li> <li>• State = 1: Open valve completely</li> <li>• State = 2: Close valve completely</li> <li>• State = 3: Move valve to target position (NewOutput)</li> <li>• State = 4: Transition time measurement successfully completed</li> <li>• State = 5: Transition time measurement canceled</li> </ul>
GetTransitTime.NewOutput	REAL	0.0	Target position for transition time measurement with position feedback The target position must be between "High endstop" and "Low endstop". The difference between NewOutput and ScaledFeedback must be at least 50% of the permissible control range.
CycleTime.StartEstimation	BOOL	TRUE	If StartEstimation = TRUE, the measurement of the PID_3Step sampling time is started. CycleTime.StartEstimation = FALSE once measurement is complete.



Tag	Data type	Default	Description
CycleTime.EnEstimation	BOOL	TRUE	If EnEstimation = TRUE, the PID_3Step sampling time is calculated. If CycleTime.EnEstimation = FALSE, the PID_3Step sampling time is not calculated and you need to correct the configuration of CycleTime.Value manually.
CycleTime.EnMonitoring	BOOL	TRUE	If EnMonitoring = TRUE, the PID_3Step sampling time is monitored. If it is not possible to execute PID_3Step within the sampling time, the error 0800h is output and the operating mode is switched. ActivateRecoverMode and ErrorBehaviour determine which operating mode is switched to. If EnMonitoring = FALSE, the PID_3Step sampling time is not monitored, the error 0800h is not output, and the operating mode is not switched.
CycleTime.Value	REAL	0.1	PID_3Step sampling time in seconds CycleTime.Value is determined automatically and is usually equivalent to the cycle time of the calling OB.
CtrlParamsBackUp.SetByUser	BOOL	FALSE	Saved value of Retain.CtrlParams.SetByUser You can reload values from the CtrlParamsBackUp structure with LoadBackUp = TRUE.
CtrlParamsBackUp.Gain	REAL	1.0	Saved proportional gain
CtrlParamsBackUp.Ti	REAL	20.0	Saved integration time in seconds
CtrlParamsBackUp.Td	REAL	0.0	Saved derivative action time in seconds
CtrlParamsBackUp.TdFiltRatio	REAL	0.2	Saved derivative delay coefficient
CtrlParamsBackUp.PWeighting	REAL	1.0	Saved proportional action weighting
CtrlParamsBackUp.DWeighting	REAL	1.0	Saved derivative action weighting
CtrlParamsBackUp.Cycle	REAL	1.0	Saved sampling time of PID algorithm in seconds
CtrlParamsBackUp.InputDeadBand	REAL	0.0	Saved dead band width of the control deviation
PIDSelfTune.SUT.CalculateParams	BOOL	FALSE	The properties of the controlled system are saved during tuning. If CalculateParams = TRUE, the PID parameters are recalculated on the basis of these properties. The PID parameters are calculated using the method set in TuneRule. CalculateParams is set to FALSE following calculation.

Tag	Data type	Default	Description
PIDSelfTune.SUT.TuneRule	INT	1	<p>Methods used to calculate parameters during pretuning:</p> <ul style="list-style-type: none"> <li>• SUT.TuneRule = 0: PID fast I</li> <li>• SUT.TuneRule = 1: PID slow I</li> <li>• SUT.TuneRule = 2: Chien, Hrones and Reswick PID</li> <li>• SUT.TuneRule = 3: Chien, Hrones, Reswick PI</li> <li>• SUT.TuneRule = 4: PID fast II</li> <li>• SUT.TuneRule = 5: PID slow II</li> </ul>
PIDSelfTune.SUT.State	INT	0	<p>The SUT.State tag indicates the current phase of pretuning:</p> <ul style="list-style-type: none"> <li>• State = 0: Initialize pretuning</li> <li>• State = 50: Determine start position without position feedback</li> <li>• State = 100: Calculate the standard deviation</li> <li>• State = 200: Find the point of inflection</li> <li>• State = 300: Determine the rise time</li> <li>• State = 9900: Pretuning successful</li> <li>• State = 1: Pretuning not successful</li> </ul>
PIDSelfTune.TIR.RunIn	BOOL	FALSE	<p>With the RunIn tag, you can specify that fine tuning can also be performed without pretuning.</p> <ul style="list-style-type: none"> <li>• RunIn = FALSE</li> </ul> <p>Pretuning is started when fine tuning is started from inactive or manual mode.</p> <p>If fine tuning is started from automatic mode, the system uses the existing PID parameters to control to the setpoint.</p> <p>Only then will fine tuning start. If pretuning is not possible, PID_3Step switches to the mode from which tuning was started.</p> <ul style="list-style-type: none"> <li>• RunIn = TRUE</li> </ul> <p>The pretuning is skipped. PID_3Step attempts to reach the setpoint with the minimum or maximum output value. This can produce increased overshoot. Only then will fine tuning start.</p> <p>RunIn is set to FALSE after fine tuning.</p>

Tag	Data type	Default	Description
PIDSelfTune.TIR.CalculateParams	BOOL	FALSE	The properties of the controlled system are saved during tuning. If CalculateParams = TRUE, the PID parameters are recalculated on the basis of these properties. The PID parameters are calculated using the method set in TuneRule. CalculateParams is set to FALSE following calculation.
PIDSelfTune.TIR.TuneRule	INT	0	Methods used to calculate parameters during fine tuning: <ul style="list-style-type: none"> <li>• TIR.TuneRule = 0: PID automatic</li> <li>• TIR.TuneRule = 1: PID fast</li> <li>• TIR.TuneRule = 2: PID slow</li> <li>• TIR.TuneRule = 3: Ziegler-Nichols PID</li> <li>• TIR.TuneRule = 4: Ziegler-Nichols PI</li> <li>• TIR.TuneRule = 5: Ziegler-Nichols P</li> </ul>
PIDSelfTune.TIR.State	INT	0	The TIR.State tag indicates the current phase of fine tuning: <ul style="list-style-type: none"> <li>• State = -100 Fine tuning is not possible. Pretuning will be performed first.</li> <li>• State = 0: Initialize fine tuning</li> <li>• State = 200: Calculate the standard deviation</li> <li>• State = 300: Attempt to reach the setpoint with the maximum or minimum output value</li> <li>• State = 400: Attempt to reach the setpoint with existing PID parameters (if pretuning was successful)</li> <li>• State = 500: Determine oscillation and calculate parameters</li> <li>• State = 9900: Fine tuning successful</li> <li>• State = 1: Fine tuning not successful</li> </ul>
Retain.TransitTime	REAL	30.0	Motor transition time in seconds Time in seconds the actuating drive requires to move the valve from the closed to the opened state. TransitTime is retentive.

Tag	Data type	Default	Description
Retain.CtrlParams.SetByUser	BOOL	FALSE	<p>If SetByUser = FALSE, the PID parameters are determined automatically and PID_3Step operates with a dead zone at the output value. The dead band width is calculated during tuning on the basis of the standard deviation of the output value and saved in Retain.CtrlParams.OutputDeadBand.</p> <p>If SetByUser = TRUE, the PID parameters are entered manually and PID_3 Step operates without a dead zone at the output value. Retain.CtrlParams.OutputDeadBand = 0.0</p> <p>SetByUser is retentive.</p>
Retain.CtrlParams.Gain	REAL	1.0	<p>Active proportional gain</p> <p>To invert the control logic, use the Config.InvertControl tag. Negative values at Gain also invert the control logic. We recommend you use only InvertControl to set the control logic. The control logic is also inverted if InvertControl = TRUE and Gain &lt; 0.0.</p> <p>Gain is retentive.</p>
Retain.CtrlParams.Ti	REAL	20.0	<ul style="list-style-type: none"> <li>• Ti &gt; 0.0: Active integration time in seconds</li> <li>• Ti = 0.0: Integral action is deactivated</li> </ul> <p>Ti is retentive.</p>
Retain.CtrlParams.Td	REAL	0.0	<ul style="list-style-type: none"> <li>• Td &gt; 0.0: Active derivative action time in seconds</li> <li>• Td = 0.0: Derivative action is deactivated</li> </ul> <p>Td is retentive.</p>
Retain.CtrlParams.TdFiltRatio	REAL	0.2	<p>Active derivative delay coefficient</p> <p>The derivative delay coefficient delays the effect of the derivative action.</p> <p>Derivative delay = derivative action time × derivative delay coefficient</p> <ul style="list-style-type: none"> <li>• 0.0: Derivative action is effective for one cycle only and therefore almost not effective.</li> <li>• 0.5: This value has proved useful in practice for controlled systems with <b>one</b> dominant time constant.</li> <li>• &gt; 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed.</li> </ul> <p>TdFiltRatio is retentive.</p>

Tag	Data type	Default	Description
Retain.CtrlParams.PWeighting	REAL	1.0	Active proportional action weighting The proportional action may weaken with changes to the setpoint. Values from 0.0 to 1.0 are applicable. <ul style="list-style-type: none"> <li>• 1.0: Proportional action for setpoint change is fully effective</li> <li>• 0.0: Proportional action for setpoint change is not effective</li> </ul> The proportional action is always fully effective when the process value is changed. PWeighting is retentive.
Retain.CtrlParams.DWeighting	REAL	1.0	Active derivative action weighting The derivative action may weaken with changes to the setpoint. Values from 0.0 to 1.0 are applicable. <ul style="list-style-type: none"> <li>• 1.0: Derivative action is fully effective upon setpoint change</li> <li>• 0.0: Derivative action is not effective upon setpoint change</li> </ul> The derivative action is always fully effective when the process value is changed. DWeighting is retentive.
Retain.CtrlParams.Cycle	REAL	1.0	Active sampling time of PID algorithm in seconds, rounded to an integer multiple of the cycle time of the calling OB. Cycle is retentive.
Retain.CtrlParams.InputDeadBand	REAL	0.0	Dead band width of the control deviation InputDeadBand is retentive.

---

**Note**

Change the tags listed in this table in "Inactive" mode to prevent malfunction of the PID controller.

---

**See also**

Parameters State and Mode V2 (Page 342)

Tag ActivateRecoverMode V2 (Page 350)

Downloading technology objects to device (Page 72)

### 8.2.4.8 Parameters State and Mode V2

#### Correlation of the parameters

The State parameter shows the current operating mode of the PID controller. You cannot change the State parameter.

With a rising edge at ModeActivate, PID\_3Step switches to the operating mode saved in the Mode in-out parameter.

When the CPU is switched on or switches from Stop to RUN mode, PID\_3Step starts in the operating mode that is saved in the Mode parameter. To leave PID\_3Step in "Inactive" mode, set RunModeByStartup = FALSE.

#### Meaning of values

State	Description of operating mode
0	Inactive The controller is switched off and no longer changes the valve position.
1	<p>Pretuning</p> <p>The pretuning determines the process response to a pulse of the output value and searches for the point of inflection. The PID parameters are calculated from the maximum rate of rise and dead time of the controlled system. You obtain the best PID parameters when you perform pretuning and fine tuning.</p> <p>Pretuning requirements:</p> <ul style="list-style-type: none"> <li>• The motor transition time has been configured or measured.</li> <li>• Inactive (State = 0), manual mode (State = 4), or automatic mode (State = 3)</li> <li>• ManualEnable = FALSE</li> <li>• Reset = FALSE</li> <li>• The setpoint and the process value lie within the configured limits.</li> </ul> <p>The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher as compared to the noise. This is most likely the case in operating modes "Inactive" and "manual mode".</p> <p>The setpoint is frozen in the CurrentSetpoint tag. Tuning is canceled when:</p> <ul style="list-style-type: none"> <li>• <math>Setpoint &gt; CurrentSetpoint + CancelTuningLevel</math></li> <li>or</li> <li>• <math>Setpoint &lt; CurrentSetpoint - CancelTuningLevel</math></li> </ul> <p>Before the PID parameters are recalculated, they are backed up and can be reactivated with LoadBackUp.</p> <p>The controller switches to automatic mode following successful pretuning. If pretuning is unsuccessful, the switchover of operating mode is dependent on ActivateRecoverMode and ErrorBehaviour.</p> <p>The pretuning phase is indicated with the SUT.State tag.</p>

State	Description of operating mode
2	<p>Fine tuning</p> <p>Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are recalculated based on the amplitude and frequency of this oscillation. PID parameters from fine tuning usually have better master control and disturbance characteristics than PID parameters from pretuning. You obtain the best PID parameters when you perform pretuning and fine tuning.</p> <p>PID_3Step automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value.</p> <p>The setpoint is frozen in the CurrentSetpoint tag. Tuning is canceled when:</p> <ul style="list-style-type: none"> <li>• Setpoint &gt; CurrentSetpoint + CancelTuningLevel</li> <li>or</li> <li>• Setpoint &lt; CurrentSetpoint - CancelTuningLevel</li> </ul> <p>The PID parameters are backed up before fine tuning. They can be reactivated with LoadBackUp.</p> <p>Requirements for fine tuning:</p> <ul style="list-style-type: none"> <li>• The motor transition time has been configured or measured.</li> <li>• The setpoint and the process value lie within the configured limits.</li> <li>• ManualEnable = FALSE</li> <li>• Reset = FALSE</li> <li>• Automatic (State = 3), inactive (State = 0) or manual (State = 4) mode</li> </ul> <p>Fine tuning proceeds as follows when started from:</p> <ul style="list-style-type: none"> <li>• Automatic mode (State = 3)</li> </ul> <p>Start fine tuning from automatic mode if you wish to improve the existing PID parameters through tuning.</p> <p>PID_3Step controls the system using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.</p> <ul style="list-style-type: none"> <li>• Inactive (State = 0) or manual mode (State = 4)</li> </ul> <p>If the requirements for pretuning are met, pretuning is started. The determined PID parameters will be used for control until the control loop has stabilized and the requirements for fine tuning have been met.</p> <p>If PIDSelfTune.TIR.RunIn = TRUE, pretuning is skipped and an attempt is made to reach the setpoint with the minimum or maximum output value. This can produce increased overshoot. Fine tuning then starts automatically.</p> <p>The controller switches to automatic mode following successful fine tuning. If fine tuning is unsuccessful, the switchover of operating mode is dependent on ActivateRecoverMode and ErrorBehaviour.</p> <p>The fine tuning phase is indicated with the TIR.State tag.</p>
3	<p>Automatic mode</p> <p>In automatic mode, PID_3Step controls the controlled system in accordance with the parameters specified.</p> <p>The controller switches to automatic mode if one the following requirements is fulfilled:</p> <ul style="list-style-type: none"> <li>• Pretuning successfully completed</li> <li>• Fine tuning successfully completed</li> <li>• Changing of the Mode in-out parameter to the value 3 and a rising edge at ModeActivate.</li> </ul> <p>The switchover from automatic mode to manual mode is only bumpless if carried out in the commissioning editor.</p> <p>The ActivateRecoverMode tag is taken into consideration in automatic mode.</p>

State	Description of operating mode
4	<p>Manual mode</p> <p>In manual mode, you specify manual output values in the Manual_UP and Manual_DN parameters or ManualValue parameter. Whether or not the actuator can be moved to the output value in the event of an error is described in the ErrorBits parameter.</p> <p>You can also activate this operating mode using ManualEnable = TRUE. We recommend that you change the operating mode using Mode and ModeActivate only.</p> <p>The switchover from manual mode to automatic mode is bumpless. Manual mode is also possible when an error is pending.</p>
5	<p>Approach substitute output value</p> <p>This operating mode is activated in the event of an error, if Errorbehaviour = TRUE and ActivateRecoverMode = FALSE..</p> <p>PID_3Step moves the actuator to the substitute output value and then switches to "Inactive" mode.</p>
6	<p>Transition time measurement</p> <p>The time that the motor needs to completely open the valve from the closed condition is determined.</p> <p>This operating mode is activated when Mode = 6 and ModeActivate = TRUE is set.</p> <p>If endstop signals are used to measure the transition time, the valve will be opened completely from its current position, closed completely, and opened completely again. If GetTransitTime.InvertDirection = TRUE, this behavior is inverted.</p> <p>If position feedback is used to measure the transition time, the actuator will be moved from its current position to a target position.</p> <p>The output value limits are not taken into consideration during the transition time measurement. The actuator can travel to the high or the low endstop.</p>
7	<p>Error monitoring</p> <p>The control algorithm is switched off and no longer changes the valve position.</p> <p>This operating mode is activated instead of "Inactive" mode in the event of an error.</p> <p>All the following conditions must be met:</p> <ul style="list-style-type: none"> <li>• Automatic mode (Mode = 3)</li> <li>• Errorbehaviour = FALSE</li> <li>• ActivateRecoverMode = TRUE</li> <li>• One or more errors have occurred in which ActivateRecoverMode (Page 350) is effective.</li> </ul> <p>As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.</p>
8	<p>Approach substitute output value with error monitoring</p> <p>This operating mode is activated instead of "approach substitute output value" mode when an error occurs. PID_3Step moves the actuator to the substitute output value and then switches to "error monitoring" mode.</p> <p>All the following conditions must be met:</p> <ul style="list-style-type: none"> <li>• Automatic mode (Mode = 3)</li> <li>• Errorbehaviour = TRUE</li> <li>• ActivateRecoverMode = TRUE</li> <li>• One or more errors have occurred in which ActivateRecoverMode (Page 350) is effective.</li> </ul> <p>As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.</p>
10	<p>Manual mode without endstop signals</p> <p>The endstop signals are <b>not</b> taken into consideration, even though Config.ActuatorEndStopOn = TRUE. The output value limits are not taken into consideration. Otherwise, PID_3Step behaves the same as in manual mode.</p>



## ENO characteristics

If State = 0, then ENO = FALSE.

If State ≠ 0, then ENO = TRUE.

## Automatic switchover of operating mode during commissioning

Automatic mode is activated following successful pretuning or fine tuning. The following table shows how Mode and State change during successful pretuning.

Cycle no.	Mode	State	Action
0	4	4	Set Mode = 1
1	1	4	Set ModeActivate = TRUE
1	4	1	Value of State is saved in Mode parameter Pretuning is started
n	4	1	Pretuning successfully completed
n	3	3	Automatic mode is started

PID\_3Step automatically switches the operating mode in the event of an error. The following table shows how Mode and State change during pretuning with errors.

Cycle no.	Mode	State	Action
0	4	4	Set Mode = 1
1	1	4	Set ModeActivate = TRUE
1	4	1	Value of State is saved in Mode parameter Pretuning is started
n	4	1	Pretuning canceled
n	4	4	Manual mode is started

If ActivateRecoverMode = TRUE, the operating mode that is saved in the Mode parameter is activated. At the start of transition time measurement, pretuning, or fine tuning, PID\_3Step saved the value of State in the Mode in/out parameter. PID\_3Step therefore switches to the operating mode from which transition time measurement or tuning was started.

If ActivateRecoverMode = FALSE, "Inactive" or "Approach substitute output value" mode is activated.

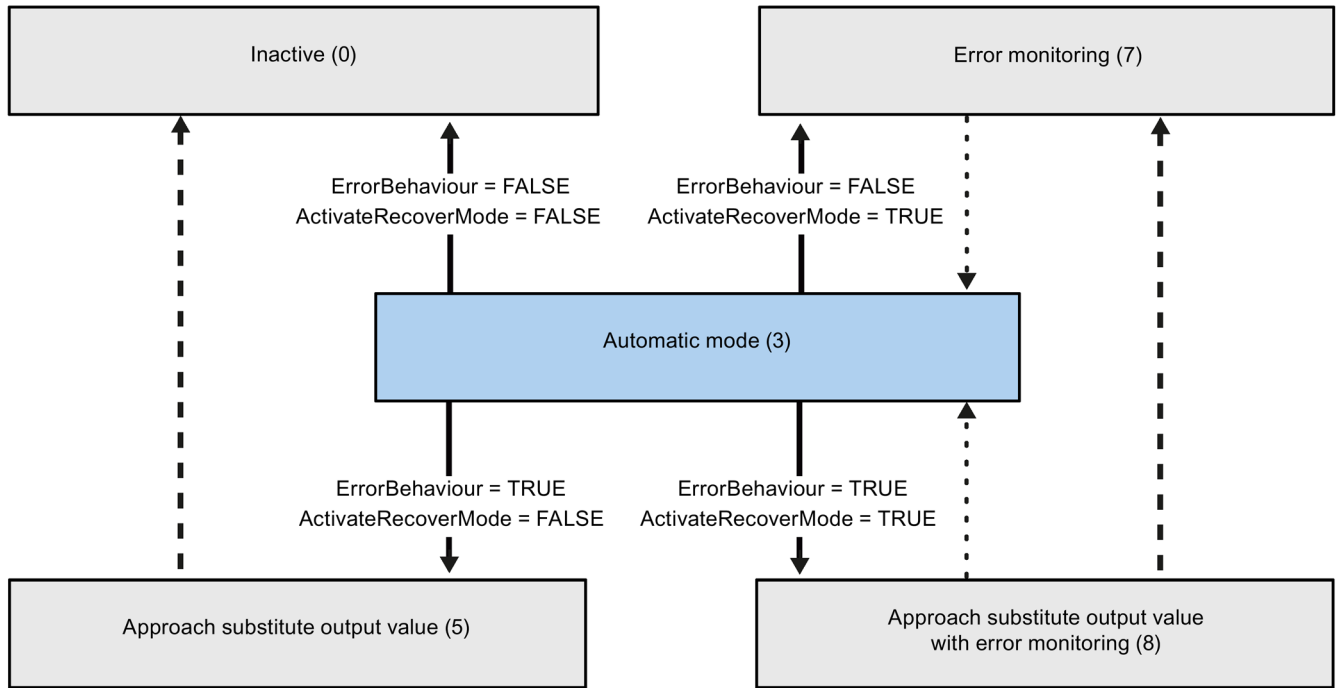
## Automatic switchover of operating mode after transition time measurement

If ActivateRecoverMode = TRUE, the operating mode that is saved in the Mode parameter is activated after successful transition time measurement.

If ActivateRecoverMode = FALSE, the system switches to "Inactive" operating mode after successful transition time measurement.

**Automatic switchover of operating mode in automatic mode**

PID\_3Step automatically switches the operating mode in the event of an error. The following diagram illustrates the influence of ErrorBehaviour and ActivateRecoverMode on this switchover of operating mode.



- ← Automatic switchover of operating mode in the event of an error
- ← - - - Automatic switchover of operating mode once the current operation has been completed.
- ← . . . . Automatic switchover of operating mode when error is no longer pending.

**See also**

- Tag ActivateRecoverMode V2 (Page 350)
- Parameter ErrorBits V2 (Page 347)

### 8.2.4.9 Parameter ErrorBits V2

If several errors are pending simultaneously, the values of the ErrorBits are displayed with binary addition. The display of ErrorBits = 0003h, for example, indicates that the errors 0001h and 0002h are pending simultaneously.

If there is a position feedback, PID\_3Step uses ManualValue as output value in manual mode. The exception is Errorbits = 10000h.

ErrorBits (DW#16#...)	Description
0000	There is no error.
0001	<p>The "Input" parameter is outside the process value limits.</p> <ul style="list-style-type: none"> <li>• Input &gt; Config.InputUpperLimit or</li> <li>• Input &lt; Config.InputLowerLimit</li> </ul> <p>If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_3Step remains in automatic mode.</p> <p>If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.</p>
0002	<p>Invalid value at "Input_PER" parameter. Check whether an error is pending at the analog input.</p> <p>If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode. As soon as the error is no longer pending, PID_3Step switches back to automatic mode.</p> <p>If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.</p>
0004	<p>Error during fine tuning. Oscillation of the process value could not be maintained.</p> <p>If ActivateRecoverMode = TRUE before the error occurred, PID_3Step cancels the tuning and switches to the operating mode that is saved in the Mode parameter.</p>
0010	<p>The setpoint was changed during tuning.</p> <p>You can set the permitted fluctuation of the setpoint at the CancelTuningLevel tag.</p> <p>If ActivateRecoverMode = TRUE before the error occurred, PID_3Step cancels the tuning and switches to the operating mode that is saved in the Mode parameter.</p>
0020	<p>Pretuning is not permitted during fine tuning.</p> <p>If ActivateRecoverMode = TRUE before the error occurred, PID_3Step remains in fine tuning mode.</p>
0080	<p>Error during pretuning. Incorrect configuration of output value limits.</p> <p>Check whether the limits of the output value are configured correctly and match the control logic.</p> <p>If ActivateRecoverMode = TRUE before the error occurred, PID_3Step cancels the tuning and switches to the operating mode that is saved in the Mode parameter.</p>
0100	<p>Error during fine tuning resulted in invalid parameters.</p> <p>If ActivateRecoverMode = TRUE before the error occurred, PID_3Step cancels the tuning and switches to the operating mode that is saved in the Mode parameter.</p>

ErrorBits (DW#16#...)	Description
0200	<p>Invalid value at "Input" parameter: Value has an invalid number format.</p> <p>If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode. As soon as the error is no longer pending, PID_3Step switches back to automatic mode.</p> <p>If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.</p>
0400	<p>Calculation of output value failed. Check the PID parameters.</p> <p>If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode. As soon as the error is no longer pending, PID_3Step switches back to automatic mode.</p> <p>If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.</p>
0800	<p>Sampling time error: PID_3Step is not called within the sampling time of the cyclic interrupt OB.</p> <p>If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_3Step remains in automatic mode.</p> <p>If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.</p> <p>If this error occurred during simulation with PLCSIM, see the notes under Simulating PID_3Step V2 with PLCSIM (Page 143).</p>
1000	<p>Invalid value at "Setpoint" parameter: Value has an invalid number format.</p> <p>If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode. As soon as the error is no longer pending, PID_3Step switches back to automatic mode.</p> <p>If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.</p>
2000	<p>Invalid value at Feedback_PER parameter.</p> <p>Check whether an error is pending at the analog input.</p> <p>The actuator cannot be moved to the substitute output value and remains in its current position. In manual mode, you can change the position of the actuator only with Manual_UP and Manual_DN, and not with ManualValue.</p> <p>If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.</p> <p>If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.</p>
4000	<p>Invalid value at Feedback parameter. Value has an invalid number format.</p> <p>The actuator cannot be moved to the substitute output value and remains in its current position. In manual mode, you can change the position of the actuator only with Manual_UP and Manual_DN, and not with ManualValue.</p> <p>If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.</p> <p>If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.</p>

<b>ErrorBits (DW#16#...)</b>	<b>Description</b>
8000	<p>Error during digital position feedback. Actuator_H = TRUE and Actuator_L = TRUE.</p> <p>The actuator cannot be moved to the substitute output value and remains in its current position. Manual mode is not possible in this state.</p> <p>In order to move the actuator from this state, you must deactivate the "Actuator endstop" (Config.ActuatorEndStopOn = FALSE) or switch to manual mode without endstop signals (Mode = 10).</p> <p>If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.</p> <p>If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.</p>
10000	<p>Invalid value at ManualValue parameter. Value has an invalid number format.</p> <p>The actuator cannot be moved to the manual value and remains in its current position.</p> <p>Specify a valid value in ManualValue or move the actuator in manual mode with Manual_UP and Manual_DN.</p>
20000	<p>Invalid value at SavePosition tag. Value has an invalid number format.</p> <p>The actuator cannot be moved to the substitute output value and remains in its current position.</p>
40000	<p>Invalid value at Disturbance parameter. Value has an invalid number format.</p> <p>If automatic mode was active and ActivateRecoverMode = TRUE before the error occurred, Disturbance is set to zero. PID_3Step remains in automatic mode.</p> <p>If pretuning or fine tuning mode was active and ActivateRecoverMode = TRUE before the error occurred, PID_3Step switches to the operating mode saved in the Mode parameter. If Disturbance in the current phase has no effect on the output value, tuning is not be canceled.</p> <p>The error has no effect during transition time measurement.</p>

### 8.2.4.10 Tag ActivateRecoverMode V2

The ActivateRecoverMode tag determines the reaction to error. The Error parameter indicates if an error is pending. When the error is no longer pending, Error = FALSE. The ErrorBits parameter shows which errors have occurred.

**NOTICE**

**Your system may be damaged.**

If ActivateRecoverMode = TRUE, PID\_3Step remains in automatic mode even if the process limit values are exceeded. This may damage your system.

It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

#### Automatic mode

ActivateRecoverMode	Description
FALSE	In the event of an error, PID_3Step switches to "Inactive" or "Approach substitute output value" mode. The controller is only activated by a falling edge at Reset or a rising edge at ModeActivate.
TRUE	<p>If errors occur frequently in automatic mode, this setting has a negative effect on the control response, because PID_3Step switches between the calculated output value and the substitute output value at each error. In this case, check the ErrorBits parameter and eliminate the cause of the error.</p> <p>If one or more of the following errors occur, PID_3Step stays in automatic mode:</p> <ul style="list-style-type: none"> <li>• 0001h: The "Input" parameter is outside the process value limits.</li> <li>• 0800h: Sampling time error</li> <li>• 40000h: Invalid value at Disturbance parameter.</li> </ul> <p>If one or more of the following errors occur, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode:</p> <ul style="list-style-type: none"> <li>• 0002h: Invalid value at Input_PER parameter.</li> <li>• 0200h: Invalid value at Input parameter.</li> <li>• 0400h: Calculation of output value failed.</li> <li>• 1000h: Invalid value at Setpoint parameter.</li> </ul> <p>If one or more of the following errors occur, PID_3Step can no longer move the actuator:</p> <ul style="list-style-type: none"> <li>• 2000h: Invalid value at Feedback_PER parameter.</li> <li>• 4000h: Invalid value at Feedback parameter.</li> <li>• 8000h: Error during digital position feedback.</li> <li>• 20000h: Invalid value at SavePosition tag. Value has an invalid number format.</li> </ul> <p>The characteristics are independent of ErrorBehaviour.</p> <p>As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.</p>

## Pretuning, fine tuning, and transition time measurement

ActivateRecoverMode	Description
FALSE	In the event of an error, PID_3Step switches to "Inactive" or "Approach substitute output value" mode. The controller is only activated by a falling edge at Reset or a rising edge at ModeActivate. The controller changes to "Inactive" mode after successful transition time measurement.
TRUE	If the following error occurs, PID_3Step remains in the active mode: <ul style="list-style-type: none"><li>• 0020h: Pretuning is not permitted during fine tuning.</li></ul> The following errors are ignored: <ul style="list-style-type: none"><li>• 10000h: Invalid value at ManualValue parameter.</li><li>• 20000h: Invalid value at SavePosition tag.</li></ul> When any other error occurs, PID_3Step cancels the tuning and switches to the mode from which tuning was started.

### Manual mode

ActivateRecoverMode is not effective in manual mode.

### See also

Static tags of PID\_3Step V2 (Page 331)

Parameters State and Mode V2 (Page 342)

### 8.2.4.11 Tag Warning V2

If several warnings are pending simultaneously, their values are displayed with binary addition. The display of warning 0005h, for example, indicates that the warnings 0001h and 0004h are pending simultaneously.

Warning (DW#16#...)	Description
0000	No warning pending.
0001	The point of inflection was not found during pretuning.
0004	The setpoint was limited to the configured limits.
0008	Not all the necessary controlled system properties were defined for the selected method of calculation. Instead, the PID parameters were calculated using the TIR.TuneRule = 3 method.
0010	The operating mode could not be changed because Reset = TRUE or ManualEnable = TRUE.
0020	The cycle time of the calling OB limits the sampling time of the PID algorithm. Improve results by using shorter OB cycle times.
0040	The process value exceeded one of its warning limits.
0080	Invalid value at Mode. The operating mode is not switched.
0100	The manual value was limited to the limits of the controller output.
0200	The specified rule for tuning is not supported. No PID parameters are calculated.
0400	The transition time cannot be measured because the actuator settings do not match the selected measuring method.
0800	The difference between the current position and the new output value is too small for transition time measurement. This can produce incorrect results. The difference between the current output value and new output value must be at least 50% of the entire control range.
1000	The substitute output value cannot be reached because it is outside the output value limits.
2000	The actuator was moved in one direction for longer than Config.VirtualActuatorLimit × Retain.TransitTime. Check whether the actuator has reached an endstop signal.

The following warnings are deleted as soon as the cause is eliminated:

- 0001h
- 0004h
- 0008h
- 0040h
- 0100h
- 2000h

All other warnings are cleared with a rising edge at Reset or ErrorAck.



## 8.2.5 PID\_3Step V1

### 8.2.5.1 Description PID\_3Step V1

#### Description

You use the PID\_3Step instruction to configure a PID controller with self tuning for valves or actuators with integrating behavior.

The following operating modes are possible:

- Inactive
- Pretuning
- Fine tuning
- Automatic mode
- Manual mode
- Approach substitute output value
- Transition time measurement
- Approach substitute output value with error monitoring
- Error monitoring

For a more detailed description of the operating modes, see the State parameter.

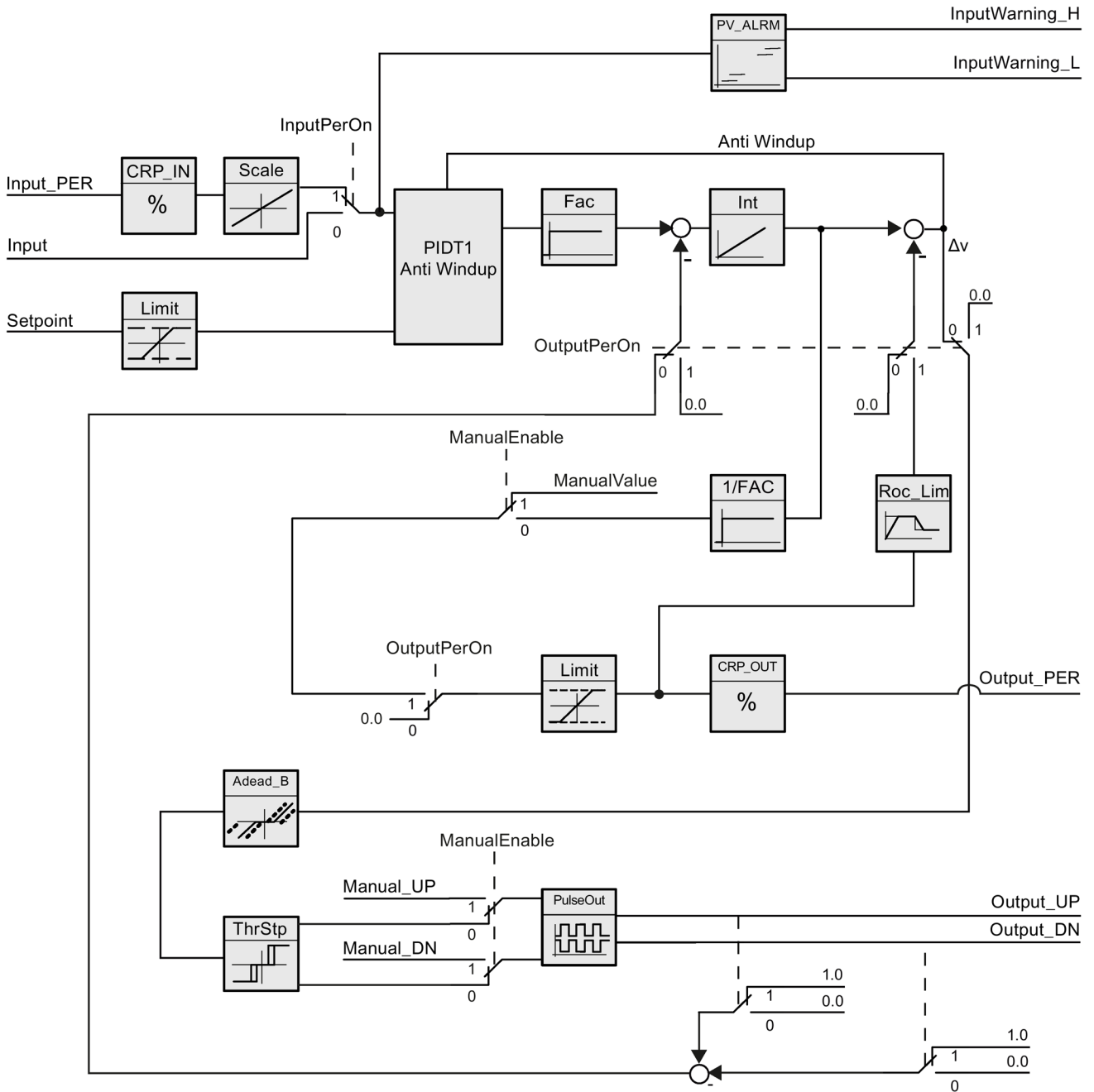
**PID algorithm**

PID\_3Step is a PIDT1 controller with anti-windup and weighting of the proportional and derivative actions. The following equation is used to calculate the output value.

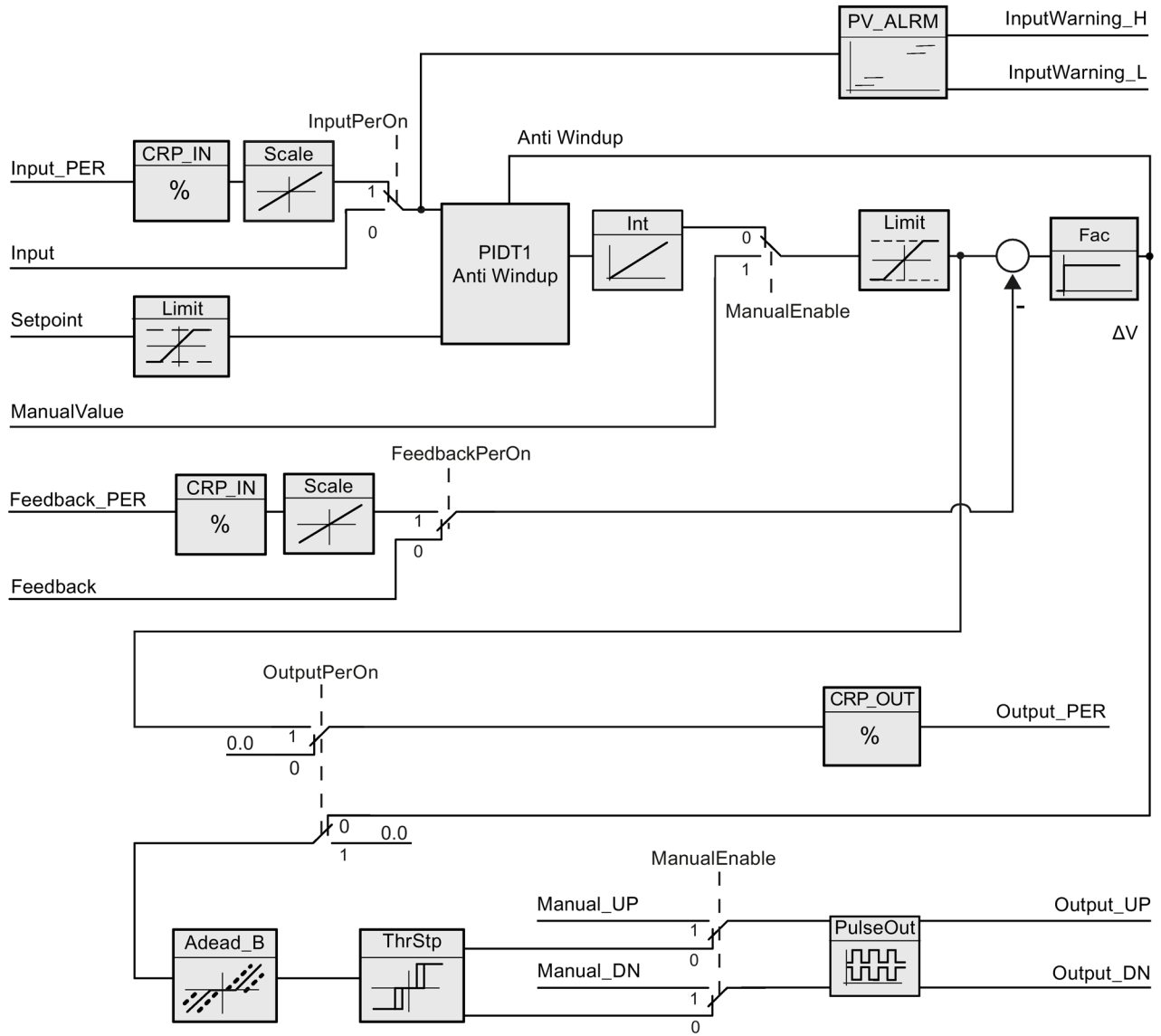
$$\Delta y = K_p \cdot s \cdot \left[ (b \cdot w - x) + \frac{1}{T_i \cdot s} (w - x) + \frac{T_d \cdot s}{a \cdot T_d \cdot s + 1} (c \cdot w - x) \right]$$

Symbol	Description
y	Output value
K <sub>p</sub>	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T <sub>i</sub>	Integral action time
a	Derivative delay coefficient (T <sub>1</sub> = a × T <sub>D</sub> )
T <sub>D</sub>	Derivative action time
c	Derivative action weighting

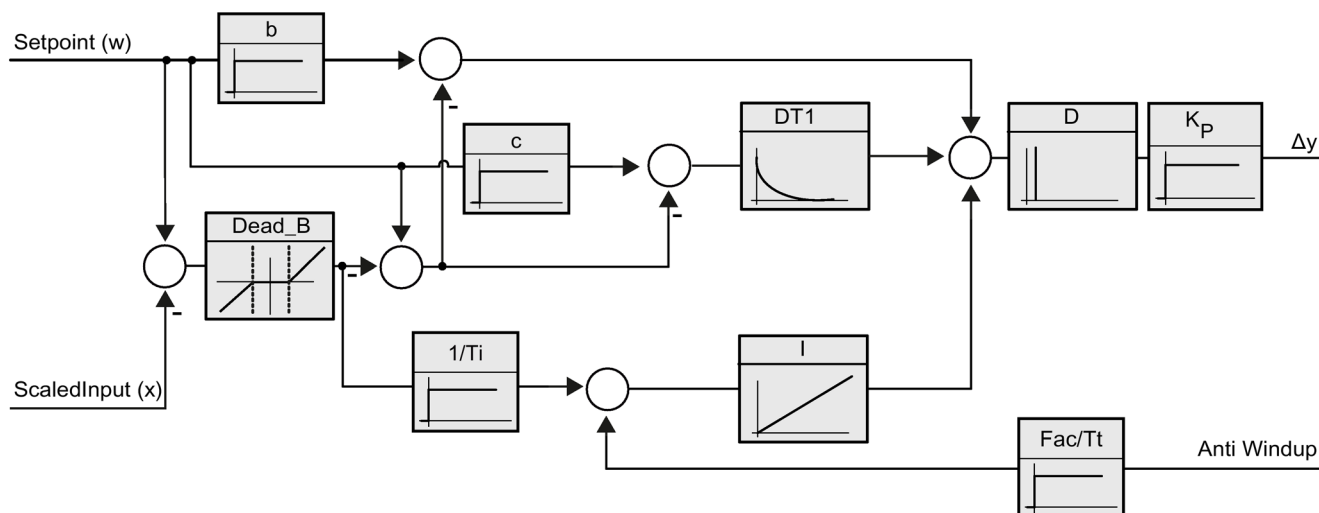
Block diagram without position feedback



Block diagram with position feedback



### Block diagram of PIDT1 with anti-windup



### Call

PID\_3Step is called in a constant time interval of the cycle time of the calling OB (preferably in a cyclic interrupt OB).

### Download to device

The actual values of retentive tags are only updated when you download PID\_3Step completely.

Downloading technology objects to device (Page 72)

### Startup

At the startup of the CPU, PID\_3Step starts in the operating mode that was last active. To leave PID\_3Step in "Inactive" mode, set RunModeByStartup = FALSE.

### Reaction to error

If errors occur, these are output in the Error parameter. You configure the reaction of PID\_3Step using the ErrorBehaviour and ActivateRecoverMode tags.

ErrorBehaviour	ActivateRecoverMode	Actuator setting configuration Set Output to	Reaction
0	FALSE	Current output value	Switch to "Inactive" mode (Mode = 0)
0	TRUE	Current output value while error is pending	Switch to "Error monitoring" mode (Mode = 7)

ErrorBehaviour	ActivateRecover-Mode	Actuator setting configuration Set Output to	Reaction
1	FALSE	Substitute output value	Switch to "Approach substitute output value" mode (Mode = 5) Switch to "Inactive" mode (Mode = 0)
1	TRUE	Substitute output value while error is pending	Switch to "Approach substitute output value with error monitoring" mode (Mode = 8) Switch to "Error monitoring" mode (Mode = 7)

The ErrorBits parameter shows which errors have occurred.

**See also**

Parameter State and Retain.Mode V1 (Page 374)

Parameter ErrorBits V1 (Page 382)

Configuring PID\_3Step V1 (Page 144)

## 8.2.5.2 Operating principle PID\_3Step V1

### Monitoring process value limits

You specify the high limit and low limit of the process value in the Config.InputUpperLimit and Config.InputLowerLimit tags. If the process value is outside these limits, an error occurs (ErrorBits = 0001hex).

You specify a high and low warning limit of the process value in the Config.InputUpperWarning and Config.InputLowerWarning tags. If the process value is outside these warning limits, a warning occurs (Warnings = 0040hex), and the InputWarning\_H or InputWarning\_L output parameter changes to TRUE.

### Limiting the setpoint

You specify a high limit and low limit of the setpoint in the Config.SetpointUpperLimit and Config.SetpointLowerLimit tags. PID\_3Step automatically limits the setpoint to the process value limits. You can limit the setpoint to a smaller range. PID\_3Step checks whether this range falls within the process value limits. If the setpoint is outside these limits, the high or low limit is used as the setpoint, and output parameter SetpointLimit\_H or SetpointLimit\_L is set to TRUE.

The setpoint is limited in all operating modes.

### Limiting the output value

You specify a high limit and low limit of the output value in the Config.OutputUpperLimit and Config.OutputLowerLimit tags. The output value limits must be within "Low endstop" and "High endstop".

- High endstop: Config.FeedbackScaling.UpperPointOut
- Low endstop: Config.FeedbackScaling.LowerPointOut

Rule:

$UpperPointOut \geq OutputUpperLimit > OutputLowerLimit \geq LowerPointOut$

The valid values for "High endstop" and "Low endstop" depend upon:

- FeedbackOn
- FeedbackPerOn
- OutputPerOn

OutputPerOn	FeedbackOn	FeedbackPerOn	LowerPointOut	UpperPointOut
FALSE	FALSE	FALSE	Cannot be set (0.0%)	Cannot be set (100.0%)
FALSE	TRUE	FALSE	-100.0% or 0.0%	0.0% or +100.0%
FALSE	TRUE	TRUE	-100.0% or 0.0%	0.0% or +100.0%
TRUE	FALSE	FALSE	Cannot be set (100.0%)	Cannot be set (100.0%)
TRUE	TRUE	FALSE	-100.0% or 0.0%	0.0% or +100.0%
TRUE	TRUE	TRUE	-100.0% or 0.0%	0.0% or +100.0%

If OutputPerOn = FALSE and FeedbackOn = FALSE, you cannot limit the output value. The digital outputs are reset with Actuator\_H = TRUE or Actuator\_L = TRUE, or after a travel time amounting to 110% of the motor transition time.

The output value is 27648 at 100% and -27648 at -100%. PID\_3Step must be able to completely close the valve. Therefore, zero must be included in the output value limits.

---

**Note**

**Use with two or more actuators**

PID\_3 Step is not suitable for use with two or more actuators (for example, in heating/cooling applications), because different actuators need different PID parameters to achieve a good control response.

---

**Substitute output value**

If an error has occurred, PID\_3Step can output a substitute output value and move the actuator to a safe position that is specified in the SavePosition tag. The substitute output value must be within the output value limits.

**Monitoring signal validity**

The values of the following parameters are monitored for validity:

- Setpoint
- Input
- Input\_PER
- Feedback
- Feedback\_PER
- Output

**Monitoring the PID\_3Step sampling time**

Ideally, the sampling time is equivalent to the cycle time of the calling OB. The PID\_3Step instruction measures the time interval between two calls. This is the current sampling time. On every switchover of operating mode and during the initial startup, the mean value is formed from the first 10 sampling times. Too great a difference between the current sampling time and this mean value triggers an error (ErrorBits = 0800 hex).

PID\_3Step is set to "Inactive" mode during tuning under the following conditions:

- New mean value  $\geq 1.1 \times$  old mean value
- New mean value  $\leq 0.9 \times$  old mean value

In automatic mode, PID\_3Step is set to "Inactive" mode under the following conditions:

- New mean value  $\geq 1.5 \times$  old mean value
- New mean value  $\leq 0.5 \times$  old mean value



## Sampling time of the PID algorithm

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the cycle time. All other functions of PID\_3Step are executed at every call.

## Measuring the motor transition time

The motor transition time is the time in seconds the motor requires to move the actuator from the closed to the opened state. The maximum time that the actuator is moved in one direction is 110% of the motor transition time. PID\_3Step requires the motor transition time to be as accurate as possible for good controller results. The data in the actuator documentation contains average values for this type of actuator. The value for the specific actuator used may differ. You can measure the motor transition time during commissioning. The output value limits are not taken into consideration during the motor transition time measurement. The actuator can travel to the high or the low endstop.

The motor transition time is taken into consideration in the calculation of the analog output value as well as in the calculation of the digital output values. It is mainly required for correct operation during auto-tuning and the anti-windup behavior. You should therefore configure the motor transition time with the value that the motor requires to move the actuator from the closed to the opened state.

If no relevant motor transition time is in effect in your process (e.g. with solenoid valves), so that the output value has a direct and full effect on the process, use PID\_Compact instead.

## Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic. For cooling and discharge control systems, it may be necessary to invert the control logic. PID\_3Step does not work with negative proportional gain. If `InvertControl = TRUE`, an increasing control deviation causes a reduction in the output value. The control logic is also taken into account during pretuning and fine tuning.

## See also

Configuring PID\_3Step V1 (Page 144)

8.2.5.3 PID\_3Step V1 input parameters

Table 8- 10

Parameters	Data type	Default	Description
Setpoint	REAL	0.0	Setpoint of the PID controller in automatic mode
Input	REAL	0.0	A tag of the user program is used as source for the process value. If you are using parameter Input, then Config.InputPerOn = FALSE must be set.
Input_PER	WORD	W#16#0	An analog input is used as the source of the process value. If you are using parameter Input_PER, then Config.InputPerOn = TRUE must be set.
Actuator_H	BOOL	FALSE	Digital position feedback of the valve for the high endstop If Actuator_H = TRUE, the valve is at the high endstop and is no longer moved towards this direction.
Actuator_L	BOOL	FALSE	Digital position feedback of the valve for the low endstop If Actuator_L = TRUE, the valve is at the low endstop and is no longer moved towards this direction.
Feedback	REAL	0.0	Position feedback of the valve If you are using parameter Feedback, then Config.FeedbackPerOn = FALSE must be set.
Feedback_PER	WORD	W#16#0	Analog position feedback of a valve If you are using parameter Feedback_PER, then Config.FeedbackPerOn = TRUE must be set. Feedback_PER is scaled based on the tags: <ul style="list-style-type: none"> <li>• Config.FeedbackScaling.LowerPointIn</li> <li>• Config.FeedbackScaling.UpperPointIn</li> <li>• Config.FeedbackScaling.LowerPointOut</li> <li>• Config.FeedbackScaling.UpperPointOut</li> </ul>
ManualEnable	BOOL	FALSE	<ul style="list-style-type: none"> <li>• A FALSE -&gt; TRUE edge selects "Manual mode", while State = 4, Retain.Mode remains unchanged.</li> <li>• A TRUE -&gt; FALSE edge selects the most recently active operating mode</li> </ul> A change of Retain.Mode will not take effect during ManualEnable = TRUE. The change of Retain.Mode will only be considered upon a TRUE -> FALSE edge at ManualEnable . <b>PID_3Step V1.1</b> If ManualEnable = TRUE when the CPU starts, PID_3Step starts in manual mode. A rising edge (FALSE > TRUE) at ManualEnable is not necessary. <b>PID_3Step V1.0</b> At the start of the CPU, PID_3Step only switches to manual mode with a rising edge (FALSE->TRUE) at ManualEnable . Without rising edge, PID_3Step starts in the last operating mode in which ManualEnable was FALSE.
ManualValue	REAL	0.0	In manual mode, you specify the absolute position of the valve. ManualValue will only be evaluated if you are using OutputPer, or if position feedback is available.

Parameters	Data type	Default	Description
Manual_UP	BOOL	FALSE	In manual mode, every rising edge opens the valve by 5% of the total control range, or for the duration of the minimum motor transition time. Manual_UP is evaluated only if you are not using Output_PER and there is no position feedback available.
Manual_DN	BOOL	FALSE	In manual mode, every rising edge closes the valve by 5% of the total control range, or for the duration of the minimum motor transition time. Manual_DN is evaluated only if you are not using Output_PER and there is no position feedback available.
Reset	BOOL	FALSE	<p>Restarts the controller.</p> <ul style="list-style-type: none"> <li>• FALSE -&gt; TRUE edge <ul style="list-style-type: none"> <li>- Switch to "Inactive" mode</li> <li>- ErrorBits and Warning are reset</li> <li>- Intermediate controller values are reset</li> </ul> <p>(PID parameters are retained)</p> </li> <li>• TRUE -&gt; FALSE edge <ul style="list-style-type: none"> <li>- Change in most recent active mode</li> <li>- If automatic mode was active before, switchover to automatic mode is bumpless.</li> </ul> </li> </ul>

8.2.5.4 PID\_3Step V1 output parameters

Table 8- 11

Parameter	Data type	Default	Description
ScaledInput	REAL	0.0	Scaled process value
ScaledFeedback	REAL	0.0	Scaled position feedback For an actuator without position feedback, the position of the actuator indicated by ScaledFeedback is very imprecise. ScaledFeedback may only be used for rough estimation of the current position in this case.
Output_UP	BOOL	FALSE	Digital output value for opening the valve If Config.OutputPerOn = FALSE, the Output_UP parameter is used.
Output_DN	BOOL	FALSE	Digital output value for closing the valve If Config.OutputPerOn = FALSE, the Output_DN parameter is used.
Output_PER	WORD	W#16#0	Analog output value If Config.OutputPerOn = TRUE, Output_PER is used. Use Output_PER if you are using a valve as actuator which is triggered via an analog output and controlled with a continuous signal, e.g. 0...10 V or 4...20 mA. The value at Output_PER corresponds to the target position of the valve, e.g. Output_PER = 13824, when the valve is to be opened by 50%.
SetpointLimit_H	BOOL	FALSE	If SetpointLimit_H = TRUE, the absolute setpoint high limit is reached. In the CPU, the setpoint is limited to the configured absolute setpoint high limit. The configured absolute process value high limit is the default for the setpoint high limit. If you configure Config.SetpointUpperLimit to a value within the process value limits, this value is used as the setpoint high limit.
SetpointLimit_L	BOOL	FALSE	If SetpointLimit_L = TRUE, the absolute setpoint low limit has been reached. In the CPU, the setpoint is limited to the configured absolute setpoint low limit. The configured absolute process value low limit is the default setting for the setpoint low limit. If you configure Config.SetpointLowerLimit to a value within the process value limits, this value is used as the setpoint low limit.
InputWarning_H	BOOL	FALSE	If InputWarning_H = TRUE, the process value has reached or exceeded the warning high limit.
InputWarning_L	BOOL	FALSE	If InputWarning_L = TRUE, the process value has reached or fallen below the warning low limit.

Parameter	Data type	Default	Description
State	INT	0	<p>The State parameter (Page 374) shows the current operating mode of the PID controller. You change the operating mode with the Retain.Mode tag.</p> <ul style="list-style-type: none"> <li>• State = 0: Inactive</li> <li>• State = 1: Pretuning</li> <li>• State = 2: Fine tuning</li> <li>• State = 3: Automatic mode</li> <li>• State = 4: Manual mode</li> <li>• State = 5: Approach substitute output value</li> <li>• State = 6: Transition time measurement</li> <li>• State = 7: Error monitoring</li> <li>• State = 8: Approach substitute output value with error monitoring</li> </ul>
Error	BOOL	FALSE	If Error = TRUE, at least one error message is pending.
ErrorBits	DWORD	DW#16#0	The ErrorBits parameter (Page 382) indicates the error messages.

**See also**

Parameter State and Retain.Mode V1 (Page 374)

Parameter ErrorBits V1 (Page 382)

**8.2.5.5 PID\_3Step V1 static tags**

You must not change tags that are not listed. These are used for internal purposes only.

Table 8- 12

Tag	Data type	Default	Description
ActivateRecoverMode	BOOL	TRUE	The ActivateRecoverMode tag (Page 385) determines the reaction to error.
RunModeByStartup	BOOL	TRUE	Activate Mode after CPU restart If RunModeByStartup = TRUE, the controller returns to the last active operating mode after a CPU restart. If RunModeByStartup = FALSE, the controller remains inactive after a CPU restart.
PhysicalUnit	INT	0	Unit of measurement of the process value and setpoint, e.g., °C, or °F.
PhysicalQuantity	INT	0	Physical quantity of the process value and setpoint, e.g., temperature.
ErrorBehaviour	INT	0	If ErrorBehaviour = 0 and an error has occurred, the valve stays at its current position and the controller switches directly to "Inactive" or "Error monitoring" mode. If ErrorBehaviour = 1 and an error occurs, the actuator moves to the substitute output value and only then switches to "Inactive" or "Error monitoring" mode. If the following errors occur, you can no longer move the valve to a configured substitute output value. <ul style="list-style-type: none"> <li>• 2000h: Invalid value at Feedback_PER parameter.</li> <li>• 4000h: Invalid value at Feedback parameter.</li> <li>• 8000h: Error during digital position feedback.</li> </ul>
Warning	DWORD	DW#16#0	The Warning tag (Page 374) displays the warnings generated since a Reset or since the last operating mode switchover.  Cyclic warnings (for example, process value warning) are shown until the cause of the warning is removed. They are automatically deleted once their cause has gone. Non-cyclic warnings (for example, point of inflection not found) remain and are deleted like errors.
SavePosition	REAL	0.0	Substitute output value If ErrorBehaviour = 1 and an error occurs, the actuator moves to a safe position for the plant and only then switches to "Inactive" mode.
CurrentSetpoint	REAL	0.0	Currently active setpoint. This value is frozen at the start of tuning.
Progress	REAL	0.0	Progress of tuning as a percentage (0.0 - 100.0)
Config.InputPerOn	BOOL	TRUE	If InputPerOn = TRUE, the Input_PER parameter is used. If InputPerOn = FALSE, the Input parameter is used.
Config.OutputPerOn	BOOL	FALSE	If OutputPerOn = TRUE, the Output_PER parameter is used. If OutputPerOn = FALSE, the Output_UP and Output_DN parameters are used.

Tag	Data type	Default	Description
Config.LoadBackUp	BOOL	FALSE	If LoadBackUp = TRUE, the last set of PID parameters is reloaded. The set was saved prior to the last tuning. LoadBackUp is automatically set back to FALSE.
Config.InvertControl	BOOL	FALSE	Invert control logic If InvertControl = TRUE, an increasing control deviation causes a reduction in the output value.
Config.FeedbackOn	BOOL	FALSE	If FeedbackOn = FALSE, a position feedback is simulated. Position feedback is generally activated when FeedbackOn = TRUE.
Config.FeedbackPerOn	BOOL	FALSE	FeedbackPerOn is only effective when FeedbackOn = TRUE. If FeedbackPerOn = TRUE, the analog input is used for the position feedback (Feedback_PER parameter). If FeedbackPerOn = FALSE, the Feedback parameter is used for the position feedback.
Config.ActuatorEndStopOn	BOOL	FALSE	If ActuatorEndStopOn = TRUE, the digital position feedback Actuator_L and Actuator_H are taken into consideration.
Config.InputUpperLimit	REAL	120.0	High limit of the process value At the I/O input, the process value can be a maximum of 18% higher than the standard range (overrange). An error is no longer signaled due to a violation of the "Process value high limit". Only a wire-break and a short-circuit are recognized and PID_3Step reacts according to the configured reaction to error. InputUpperLimit > InputLowerLimit
Config.InputLowerLimit	REAL	0.0	Low limit of the process value InputLowerLimit < InputUpperLimit
Config.InputUpperWarning	REAL	+3.402822e+38	Warning high limit of the process value If you set InputUpperWarning outside the process value limits, the configured absolute process value high limit is used as the warning high limit. If you configure InputUpperWarning within the process value limits, this value is used as the warning high limit. InputUpperWarning > InputLowerWarning InputUpperWarning ≤ InputUpperLimit
Config.InputLowerWarning	REAL	-3.402822e+38	Warning low limit of the process value If you set InputLowerWarning outside the process value limits, the configured absolute process value low limit is used as the warning low limit. If you configure InputLowerWarning within the process value limits, this value is used as the warning low limit. InputLowerWarning < InputUpperWarning InputLowerWarning ≥ InputLowerLimit
Config.OutputUpperLimit	REAL	100.0	High limit of output value For details, see OutputLowerLimit

Tag	Data type	Default	Description
Config.OutputLowerLimit	REAL	0.0	Low limit of output value If OutputPerOn = TRUE or FeedbackOn = TRUE, the range of values from -100% to +100%, including zero, is valid. At -100%, Output = -27648; at +100%, Output = 27648 If OutputPerOn = FALSE, the range of values from 0% to 100% is valid. The valve is completely closed at 0% and completely opened at 100%.
Config.SetpointUpperLimit	REAL	+3.402822e+38	High limit of setpoint If you set SetpointUpperLimit outside the process value limits, the configured absolute process value high limit is preassigned as the setpoint high limit. If you configure SetpointUpperLimit within the process value limits, this value is used as the setpoint high limit.
Config.SetpointLowerLimit	REAL	- 3.402822e+38	Low limit of the setpoint If you set SetpointLowerLimit outside the process value limits, the configured absolute process value low limit is preassigned as the setpoint low limit. If you set SetpointLowerLimit within the process value limits, this value is used as the setpoint low limit.
Config.MinimumOnTime	REAL	0.0	Minimum ON time Minimum time in seconds for which the servo drive must be switched on. Config.MinimumOnTime is only effective if Output_UP and Output_DN are used (Config.OutputPerOn = FALSE).
Config.MinimumOffTime	REAL	0.0	Minimum OFF time Minimum time in seconds for which the servo drive must be switched off. Config.MinimumOffTime is only effective if Output_UP and Output_DN are used (Config.OutputPerOn = FALSE).
Config.TransitTime	REAL	30.0	Motor transition time Time in seconds the actuating drive requires to move the valve from the closed to the opened state.
Config.InputScaling.UpperPointIn	REAL	27648.0	Scaling Input_PER high Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.InputScaling.LowerPointIn	REAL	0.0	Scaling Input_PER low Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.InputScaling.UpperPointOut	REAL	100.0	Scaled high process value Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.



Tag	Data type	Default	Description
Con-fig.InputScaling.LowerPointOut	REAL	0.0	Scaled low process value Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Con-fig.FeedbackScaling.UpperPointIn	REAL	27648.0	Scaling Feedback_PER high Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.
Con-fig.FeedbackScaling.LowerPointIn	REAL	0.0	Scaling Feedback_PER low Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.
Con-fig.FeedbackScaling.UpperPointOut	REAL	100.0	High endstop Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.
Con-fig.FeedbackScaling.LowerPointOut	REAL	0.0	Low endstop Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.
GetTransitTime.InvertDirection	BOOL	FALSE	If InvertDirection = FALSE, the valve is fully opened, closed, and then reopened in order to determine the valve transition time. If InvertDirection = TRUE, the valve is fully closed, opened, and then closed again.
GetTransitTime.SelectFeedback	BOOL	FALSE	If SelectFeedback = TRUE, then Feedback_PER, or Feedback is taken into consideration in the transition time measurement. If SelectFeedback = FALSE, then Actuator_H and Actuator_L are taken into consideration in the transition time measurement.
GetTransitTime.Start	BOOL	FALSE	If Start = TRUE, the transition time measurement is started.
GetTransitTime.State	INT	0	Current phase of the transition time measurement <ul style="list-style-type: none"> <li>• State = 0: Inactive</li> <li>• State = 1: Open valve completely</li> <li>• State = 2: Close valve completely</li> <li>• State = 3: Move valve to target position (NewOutput)</li> <li>• State = 4: Transition time measurement successfully completed</li> <li>• State = 5: Transition time measurement canceled</li> </ul>
GetTransitTime.NewOutput	REAL	0.0	Target position for transition time measurement with position feedback The target position must be between "High endstop" and "Low endstop". The difference between NewOutput and ScaledFeedback must be at least 50% of the permissible control range.

8.2 PID\_3Step

Tag	Data type	Default	Description
CycleTime.StartEstimation	BOOL	TRUE	If StartEstimation = TRUE, the measurement of the PID_3Step sampling time is started. CycleTime.StartEstimation = FALSE once measurement is complete.
CycleTime.EnEstimation	BOOL	TRUE	If EnEstimation = TRUE, the PID_3Step sampling time is calculated.
CycleTime.EnMonitoring	BOOL	TRUE	If EnMonitoring = TRUE, the PID_3Step sampling time is monitored. If it is not possible to execute PID_3Step within the sampling time, the error 0800h is output and the operating mode is switched. ActivateRecoverMode and ErrorBehaviour determine which operating mode is switched to.  If EnMonitoring = FALSE, the PID_3Step sampling time is not monitored, the error 0800h is not output, and the operating mode is not switched.
CycleTime.Value	REAL	0.1	PID_3Step sampling time in seconds CycleTime.Value is determined automatically and is usually equivalent to the cycle time of the calling OB.
CtrlParamsBackUp.SetByUser	BOOL	FALSE	Saved value of Retain.CtrlParams.SetByUser. You can reload values from the CtrlParamsBackUp structure with Config.LoadBackUp = TRUE.
CtrlParamsBackUp.Gain	REAL	1.0	Saved proportional gain
CtrlParamsBackUp.Ti	REAL	20.0	Saved integral action time
CtrlParamsBackUp.Td	REAL	0.0	Saved derivative action time
CtrlParamsBackUp.TdFiltRatio	REAL	0.0	Saved derivative delay coefficient
CtrlParamsBackUp.PWeighting	REAL	0.0	Saved proportional action weighting
CtrlParamsBackUp.DWeighting	REAL	0.0	Saved derivative action weighting
CtrlParamsBackUp.Cycle	REAL	1.0	Saved sampling time of PID algorithm
CtrlParamsBackUp.InputDeadBand	REAL	0.0	Saved dead zone width of the control deviation
PIDSelfTune.SUT.CalculateSUTParams	BOOL	FALSE	The properties of the controlled system are saved during tuning. If CalculateSUTParams = TRUE, the PID parameters are recalculated on the basis of these properties. The PID parameters are calculated using the method set in TuneRuleSUT. CalculateSUTParams is set to FALSE following calculation.
PIDSelfTune.SUT.TuneRuleSUT	INT	1	Methods used to calculate parameters during pretuning: <ul style="list-style-type: none"> <li>• TuneRuleSUT = 0: PID fast I</li> <li>• TuneRuleSUT = 1: PID slow I</li> <li>• TuneRuleSUT = 2: Chien, Hrones and Reswick PID</li> <li>• TuneRuleSUT = 3: Chien, Hrones, Reswick PI</li> <li>• TuneRuleSUT = 4: PID fast II</li> <li>• TuneRuleSUT = 5: PID slow II</li> </ul>
PIDSelfTune.SUT.State	INT	0	The SUT.State tag indicates the current phase of pretuning:

Tag	Data type	Default	Description
PIDSelfTune.TIR.RunIn	BOOL	FALSE	<ul style="list-style-type: none"> <li>RunIn = FALSE</li> </ul> <p>Pretuning is started when fine tuning is started from inactive or manual mode.</p> <p>If fine tuning is started from automatic mode, the system uses the existing PID parameters to control to the setpoint.</p> <p>Only then will fine tuning start. If pretuning is not possible, PID_3Step switches to "Inactive" mode.</p> <ul style="list-style-type: none"> <li>RunIn = TRUE</li> </ul> <p>The pretuning is skipped. PID_3Step attempts to reach the setpoint with the minimum or maximum output value. This can produce increased overshoot. Only then will fine tuning start.</p> <p>RunIn is set to FALSE after fine tuning.</p>
PIDSelf-Tune.TIR.CalculateTIRParams	BOOL	FALSE	<p>The properties of the controlled system are saved during tuning. If CalculateTIRParams = TRUE, the PID parameters are recalculated on the basis of these properties. The PID parameters are calculated using the method set in TuneRuleTIR. CalculateTIRParams is set to FALSE following calculation.</p>
PIDSelfTune.TIR.TuneRuleTIR	INT	0	<p>Methods used to calculate parameters during fine tuning:</p> <ul style="list-style-type: none"> <li>TuneRuleTIR = 0: PID automatic</li> <li>TuneRuleTIR = 1: PID fast</li> <li>TuneRuleTIR = 2: PID slow</li> <li>TuneRuleTIR = 3: Ziegler-Nichols PID</li> <li>TuneRuleTIR = 4: Ziegler-Nichols PI</li> <li>TuneRuleTIR = 5: Ziegler-Nichols P</li> </ul>
PIDSelfTune.TIR.State	INT	0	<p>The TIR.State tag indicates the current phase of "fine tuning":</p>

Tag	Data type	Default	Description
Retain.Mode	INT	0	<p>A change to the value of Retain.Mode initiates a switch to another operating mode.</p> <p>The following operating mode is enabled upon a change of Mode to:</p> <ul style="list-style-type: none"> <li>• Mode = 0: Inactive</li> <li>• Mode = 1: Pretuning</li> <li>• Mode = 2: Fine tuning</li> <li>• Mode = 3: Automatic mode</li> <li>• Mode = 4: Manual mode</li> <li>• Mode = 5: Approach substitute output value</li> <li>• Mode = 6: Transition time measurement</li> <li>• Mode = 7: Error monitoring</li> <li>• Mode = 8: Approach substitute output value with error monitoring</li> </ul> <p>Mode is retentive.</p>
Retain.CtrlParams.SetByUser	BOOL	FALSE	<p>If SetByUser = FALSE, the PID parameters are determined automatically and PID_3Step operates with a dead zone at the output value. The dead zone width is calculated during tuning on the basis of the standard deviation of the output value and saved in Retain.CtrlParams.OutputDeadBand.</p> <p>If SetByUser = TRUE, the PID parameters are entered manually and PID_3 Step operates without a dead zone at the output value. Retain.CtrlParams.OutputDeadBand = 0.0</p> <p>SetByUser is retentive.</p>
Retain.CtrlParams.Gain	REAL	1.0	<p>Active proportional gain</p> <p>Gain is retentive.</p>
Retain.CtrlParams.Ti	REAL	20.0	<ul style="list-style-type: none"> <li>• Ti &gt; 0.0: Active integral action time</li> <li>• Ti = 0.0: Integral action is deactivated</li> </ul> <p>Ti is retentive.</p>
Retain.CtrlParams.Td	REAL	0.0	<ul style="list-style-type: none"> <li>• Td &gt; 0.0: Active derivative action time</li> <li>• Td = 0.0: Derivative action is deactivated</li> </ul> <p>Td is retentive.</p>
Retain.CtrlParams.TdFiltRatio	REAL	0.0	<p>Active derivative delay coefficient</p> <p>TdFiltRatio is retentive.</p>
Retain.CtrlParams.PWeighting	REAL	0.0	<p>Active proportional action weighting</p> <p>PWeighting is retentive.</p>
Retain.CtrlParams.DWeighting	REAL	0.0	<p>Active derivative action weighting</p> <p>DWeighting is retentive.</p>

Tag	Data type	Default	Description
Retain.CtrlParams.Cycle	REAL	1.0	Active sampling time of PID algorithm in seconds, rounded to an integer multiple of the cycle time of the calling OB. Cycle is retentive.
Re-retain.CtrlParams.InputDeadBand	REAL	0.0	Dead zone width of the control deviation InputDeadBand is retentive.

---

**Note**

Change the tags listed in this table in "Inactive" mode to prevent malfunction of the PID controller. "Inactive" mode is forced by setting the "Retain.Mode" tag to "0".

---

**See also**

Parameter State and Retain.Mode V1 (Page 374)

Tag ActivateRecoverMode V1 (Page 385)

Downloading technology objects to device (Page 72)

### 8.2.5.6 Parameter State and Retain.Mode V1

#### Correlation of the parameters

The State parameter shows the current operating mode of the PID controller. You cannot change the State parameter.

To switch from one operating mode to another, you must change the Retain.Mode tag. This also applies when the value for the new operating mode is already in Retain.Mode. For example, set Retain.Mode = 0 first and then Retain.Mode = 3. Provided the current operating mode of the controller permits this switchover, State will be set to the value of Retain.Mode.

When PID\_3Step automatically switches from one operating mode to another, the following applies: State != Retain.Mode.

Examples:

- After successful pretuning  
State = 3 and Retain.Mode = 1
- In the event of an error  
State = 0 and Retain.Mode remain at the previous value, for example, Retain.Mode = 3
- ManualEnalbe = TRUE  
State = 4 and Retain.Mode remain at the previous value, e.g., Retain.Mode = 3

---

#### Note

You want, for example, to repeat successful fine tuning without exiting automatic mode with Mode = 0.

Setting Retain.Mode to an invalid value such as 9999 for one cycle has no effect on State. Set Mode = 2 in the next cycle. In this way, you can generate a change to Retain.Mode without first switching to "Inactive" mode.

---

## Meaning of values

State / Re- tain.Mode	Description
0	<p>Inactive</p> <p>The controller is switched off and no longer changes the valve position.</p>
1	<p>Pretuning</p> <p>The pretuning determines the process response to a pulse of the output value and searches for the point of inflection. The optimized PID parameters are calculated as a function of the maximum rate of rise and dead time of the controlled system.</p> <p>Pretuning requirements:</p> <ul style="list-style-type: none"> <li>• State = 0 or State = 4</li> <li>• ManualEnable = FALSE</li> <li>• The motor transition time has been configured or measured.</li> <li>• The setpoint and the process value lie within the configured limits.</li> </ul> <p>The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher as compared to the noise.</p> <p>Before the PID parameters are recalculated, they are backed up and can be reactivated with Config.LoadBackUp. The setpoint is frozen in the CurrentSetpoint tag.</p> <p>The controller switches to automatic mode following successful pretuning and to "Inactive" mode following unsuccessful pretuning.</p> <p>The pretuning phase is indicated with the SUT.State tag.</p>

State / Retain.Mode	Description
2	<p>Fine tuning</p> <p>Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are tuned based on the amplitude and frequency of this oscillation. The differences between the process response during pretuning and fine tuning are analyzed. All PID parameters are recalculated from the results. PID parameters from fine tuning usually have better master control and disturbance characteristics than PID parameters from pretuning.</p> <p>PID_3Step automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value.</p> <p>The PID parameters are backed up before fine tuning. They can be reactivated with Config.LoadBackUp. The setpoint is frozen in the CurrentSetpoint tag.</p> <p>Requirements for fine tuning:</p> <ul style="list-style-type: none"> <li>• The motor transition time has been configured or measured.</li> <li>• The setpoint and the process value lie within the configured limits.</li> <li>• ManualEnable = FALSE</li> <li>• Automatic (State = 3), inactive (State = 0) or manual (State = 4) mode</li> </ul> <p>Fine tuning proceeds as follows when started from:</p> <ul style="list-style-type: none"> <li>• Automatic mode (State = 3)                     <p>Start fine tuning from automatic mode if you wish to improve the existing PID parameters through tuning.</p> <p>PID_3Step controls the system using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.</p> </li> <li>• Inactive (State = 0) or manual mode (State = 4)                     <p>Pretuning is always started first. The determined PID parameters will be used for control until the control loop has stabilized and the requirements for fine tuning have been met.</p> <p>If PIDSelfTune.TIR.RunIn = TRUE, pretuning is skipped and an attempt is made to reach the setpoint with the minimum or maximum output value. This can produce increased overshoot. Fine tuning then starts automatically.</p> </li> </ul> <p>The controller switches to automatic mode following successful fine tuning. If fine tuning was not successful, the controller switches to "Inactive" mode.</p> <p>The fine tuning phase is indicated with the TIR.State tag.</p>
3	<p>Automatic mode</p> <p>In automatic mode, PID_3Step controls the controlled system in accordance with the parameters specified. The controller switches to automatic mode if one the following requirements is fulfilled:</p> <ul style="list-style-type: none"> <li>• Pretuning successfully completed</li> <li>• Fine tuning successfully completed</li> <li>• Changing the Retain.Mode tag to the value 3.</li> </ul> <p>When the CPU is switched on or switches from Stop to RUN mode, PID_3Step starts in the most recently active operating mode. To leave PID_3Step in "Inactive" mode, set RunModeByStartup = FALSE.</p> <p>The ActivateRecoverMode tag is taken into consideration in automatic mode.</p>

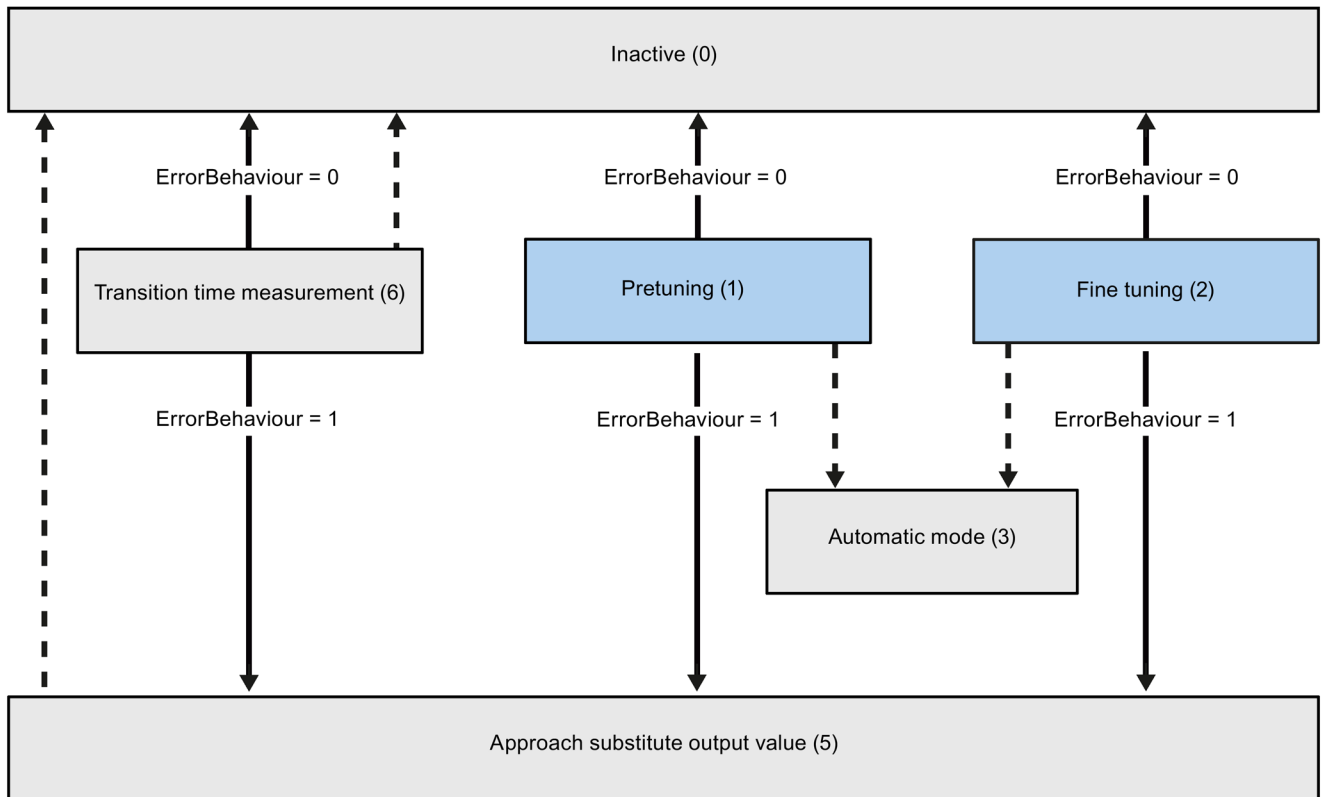


State / Retain.Mode	Description
4	<p>Manual mode</p> <p>In manual mode, you specify manual output values in the Manual_UP and Manual_DN parameters or ManualValue parameter. Whether or not the actuator can be moved to the output value in the event of an error is described in the ErrorBits parameter.</p> <p>This operating mode is enabled if Retain.Mode = 4, or on a rising edge at ManualEnable.</p> <p>If ManualEnable changes to TRUE, only State changes. Retain.Mode retains its current value. On a falling edge at ManualEnable, PID_3Step returns to the previous operating mode.</p> <p>The switchover to automatic mode is bumpless.</p> <p><b>PID_3Step V1.1</b></p> <p>Manual mode is always possible in the event of an error.</p> <p><b>PID_3Step V1.0</b></p> <p>Manual mode is dependent on the ActivateRecoverMode tag in the event of an error.</p>
5	<p>Approach substitute output value</p> <p>This operating mode is activated in the event of an error or when Reset = TRUE if Errorbehaviour = 1 and ActivateRecoverMode = FALSE..</p> <p>PID_3Step moves the actuator to the substitute output value and then switches to "Inactive" mode.</p>
6	<p>Transition time measurement</p> <p>The time that the motor needs to completely open the valve from the closed condition is determined.</p> <p>This operating mode is activated when GetTransitTime.Start = TRUE is set.</p> <p>If endstop signals are used to measure the transition time, the valve will be opened completely from its current position, closed completely, and opened completely again. If GetTransitTime.InvertDirection = TRUE, this behavior is inverted.</p> <p>If position feedback is used to measure the transition time, the actuator will be moved from its current position to a target position.</p> <p>The output value limits are not taken into consideration during the transition time measurement. The actuator can travel to the high or the low endstop.</p>

State / Re- tain.Mode	Description
7	<p>Error monitoring</p> <p>The control algorithm is switched off and no longer changes the valve position.</p> <p>This operating mode is activated instead of "Inactive" mode in the event of an error.</p> <p>All the following conditions must be met:</p> <ul style="list-style-type: none"> <li>• Mode = 3 (automatic mode)</li> <li>• Errorbehaviour = 0</li> <li>• ActivateRecoverMode = TRUE</li> <li>• One or more errors have occurred in which ActivateRecoverMode (Page 385) is effective.</li> </ul> <p>As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.</p>
8	<p>Approach substitute output value with error monitoring</p> <p>This operating mode is activated instead of "Approach substitute output value" mode in the event of an error. PID_3Step moves the actuator to the substitute output value and then switches to "Error monitoring" mode.</p> <p>All the following conditions must be met:</p> <ul style="list-style-type: none"> <li>• Mode = 3 (automatic mode)</li> <li>• Errorbehaviour = 1</li> <li>• ActivateRecoverMode = TRUE</li> <li>• One or more errors have occurred in which ActivateRecoverMode (Page 385) is effective.</li> </ul> <p>As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.</p>

### Automatic switchover of operating mode during commissioning

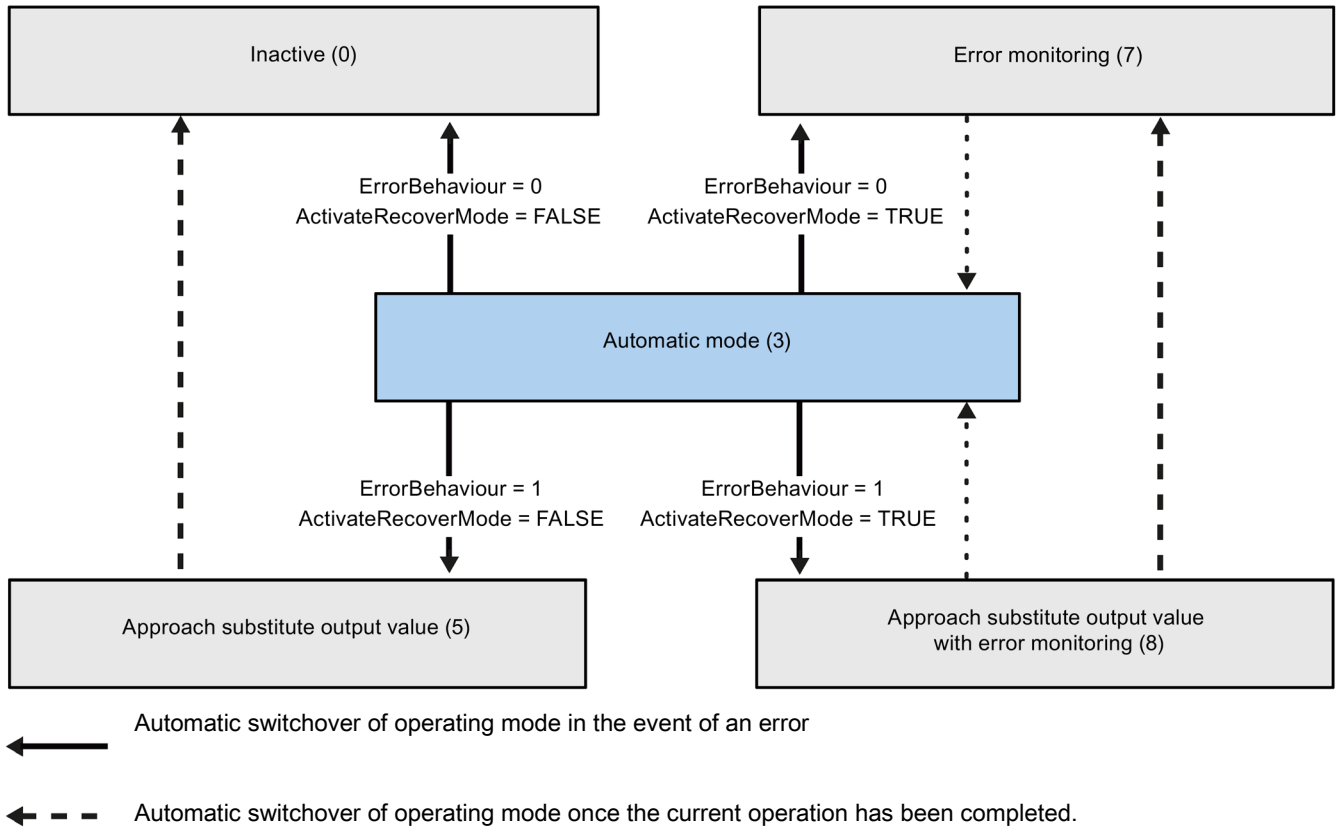
PID\_3Step automatically switches the operating mode in the event of an error. The following diagram illustrates the influence of ErrorBehaviour on the switchover of operating mode from transition time measurement, pretuning, and fine tuning modes.



- ← Automatic switchover of operating mode in the event of an error
- ← - - - Automatic switchover of operating mode once the current operation has been completed.

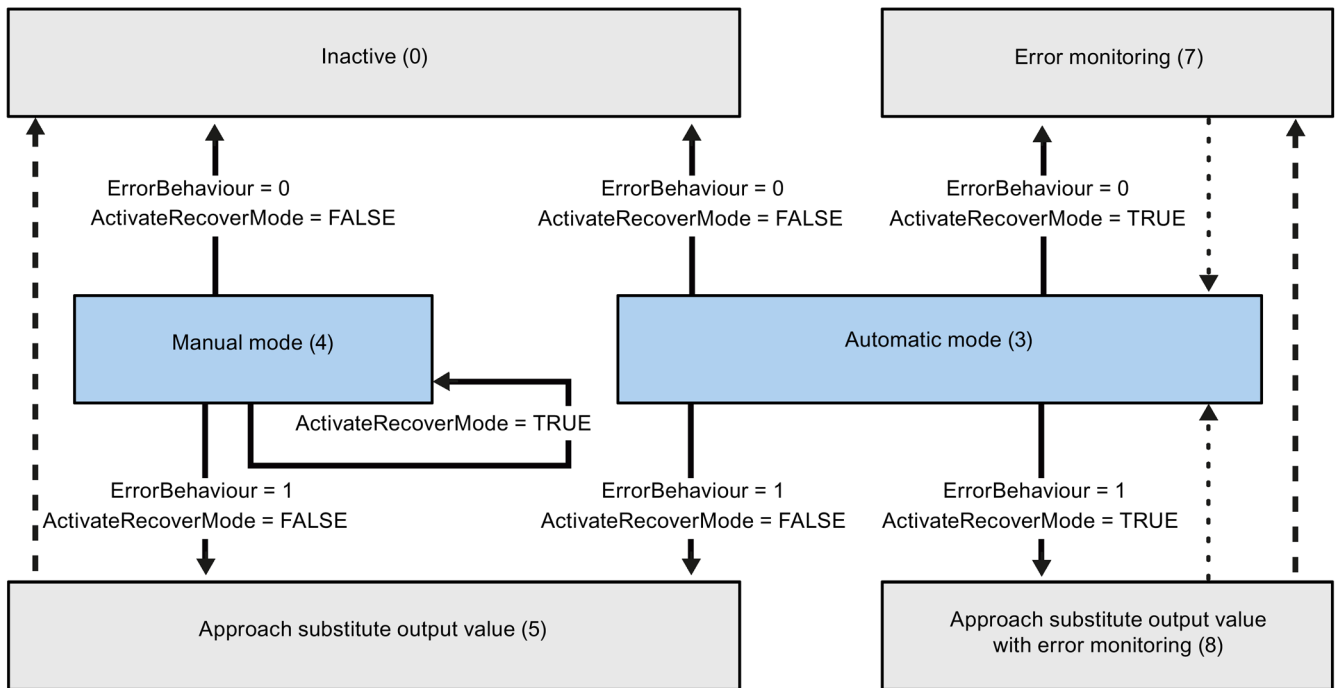
### Automatic switchover of operating mode in automatic mode (PID\_3Step V1.1)

PID\_3Step automatically switches the operating mode in the event of an error. The following diagram illustrates the influence of ErrorBehaviour and ActivateRecoverMode on this switchover of operating mode.



### Automatic switchover of operating mode in automatic and manual modes (PID\_3Step V1.0)

PID\_3Step automatically switches the operating mode in the event of an error. The following diagram illustrates the influence of ErrorBehaviour and ActivateRecoverMode on this switchover of operating mode.



- ← Automatic switchover of operating mode in the event of an error
- ← - - - Automatic switchover of operating mode once the current operation has been completed.
- ← . . . . . Automatic switchover of operating mode when error is no longer pending.

#### See also

- Tag ActivateRecoverMode V1 (Page 385)
- Parameter ErrorBits V1 (Page 382)

### 8.2.5.7 Parameter ErrorBits V1

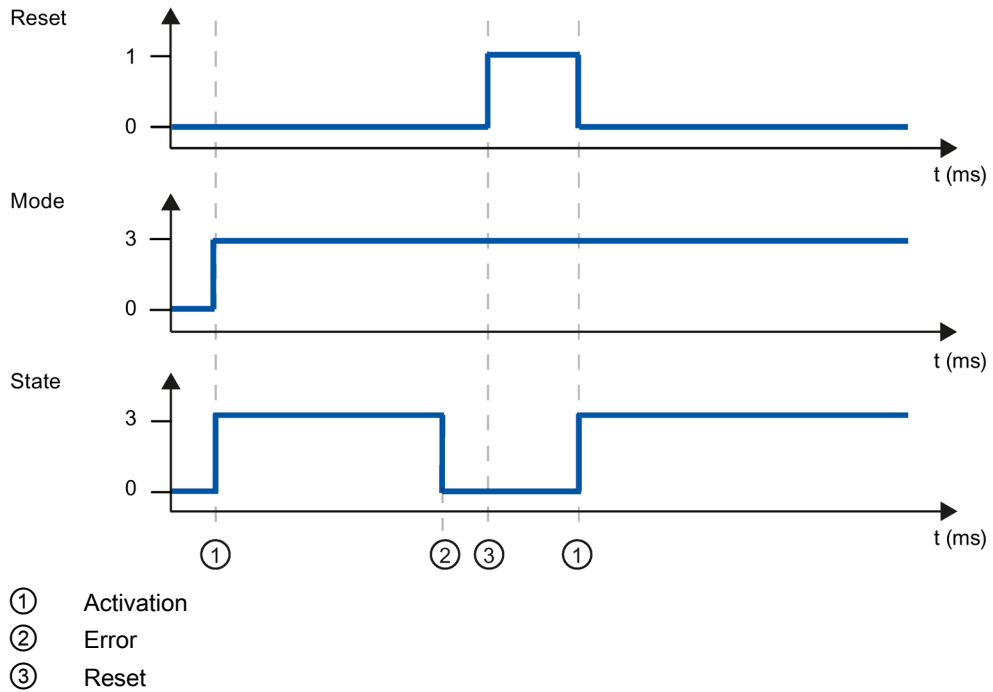
If several errors are pending simultaneously, the values of the error codes are displayed with binary addition. The display of error code 0003, for example, indicates that the errors 0001 and 0002 are pending simultaneously.

ErrorBits (DW#16#...)	Description
0000	There is no error.
0001	<p>The "Input" parameter is outside the process value limits.</p> <ul style="list-style-type: none"> <li>Input &gt; Config.InputUpperLimit or</li> <li>Input &lt; Config.InputLowerLimit</li> </ul> <p>If ActivateRecoverMode = TRUE and ErrorBehaviour = 1, the actuator moves to the substitute output value. If ActivateRecoverMode = TRUE and ErrorBehaviour = 0, the actuator stops in its current position. If ActivateRecoverMode = FALSE, the actuator stops in its current position.</p> <p>PID_3Step V1.1 You can move the actuator in manual mode.</p> <p>PID_3Step V1.0 Manual mode is not possible in this state. You cannot move the actuator again until you eliminate the error.</p>
0002	<p>Invalid value at "Input_PER" parameter. Check whether an error is pending at the analog input.</p> <p>If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.</p>
0004	Error during fine tuning. Oscillation of the process value could not be maintained.
0020	Pretuning is not permitted in automatic mode or during fine tuning.
0080	<p>Error during pretuning. Incorrect configuration of output value limits.</p> <p>Check whether the limits of the output value are configured correctly and match the control logic.</p>
0100	Error during fine tuning resulted in invalid parameters.
0200	<p>Invalid value at "Input" parameter: Value has an invalid number format.</p> <p>If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.</p>
0400	Calculation of output value failed. Check the PID parameters.
0800	<p>Sampling time error: PID_3Step is not called within the sampling time of the cyclic interrupt OB.</p> <p>If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.</p> <p>If this error occurred during simulation with PLCSIM, see the notes under Simulating PID_3Step V1 with PLCSIM (Page 163).</p>
1000	<p>Invalid value at "Setpoint" parameter: Value has an invalid number format.</p> <p>If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.</p>
2000	<p>Invalid value at Feedback_PER parameter.</p> <p>Check whether an error is pending at the analog input.</p> <p>The actuator cannot be moved to the substitute output value and remains in its current position. Manual mode is not possible in this state. You must deactivate position feedback (Config. FeedbackOn = FALSE) to move the actuator from this state.</p> <p>If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.</p>

<b>ErrorBits (DW#16#...)</b>	<b>Description</b>
4000	<p>Invalid value at Feedback parameter. Value has an invalid number format.</p> <p>The actuator cannot be moved to the substitute output value and remains in its current position. Manual mode is not possible in this state. You must deactivate position feedback (Config. FeedbackOn = FALSE) to move the actuator from this state.</p> <p>If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.</p>
8000	<p>Error during digital position feedback. Actuator_H = TRUE and Actuator_L = TRUE.</p> <p>The actuator cannot be moved to the substitute output value and remains in its current position. Manual mode is not possible in this state.</p> <p>In order to move the actuator from this state, you must deactivate the "Actuator endstop" (Config.ActuatorEndStopOn = FALSE).</p> <p>If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.</p>

### 8.2.5.8 Parameter Reset V1

A rising edge at Reset triggers a change to "Inactive" mode, and errors and warnings are reset. A falling edge at Reset triggers a change to the most recently active operating mode. If automatic mode was active before, switchover to automatic mode is bumpless.





### 8.2.5.9 Tag ActivateRecoverMode V1

The effect of the ActivateRecoverMode variable depends on the version of the PID\_3Step.

#### Behavior in version 1.1

The ActivateRecoverMode variable determines the behavior in the event of an error in automatic mode. ActivateRecoverMode is not effective during pretuning, fine tuning and transition time measurement.

ActivateRecoverMode	Description
FALSE	In the event of an error, PID_3Step switches to "Inactive" or "Approach substitute output value" operating mode. The controller is activated by a reset or a change in Retain.Mode.
TRUE	<p>If errors occur frequently in automatic mode, this setting has a negative effect on the control response. In this case, check the ErrorBits parameter and eliminate the cause of the error.</p> <p>If one or more errors occur, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode:</p> <ul style="list-style-type: none"> <li>• 0002h: Invalid value at parameter Input_PER.</li> <li>• 0200h: Invalid value at parameter Input.</li> <li>• 0800h: Sampling time error</li> <li>• 1000h: Invalid value at parameter Setpoint.</li> <li>• 2000h: Invalid value at parameter Feedback_PER.</li> <li>• 4000h: Invalid value at parameter Feedback.</li> <li>• 8000h: Error in digital position feedback.</li> </ul> <p>With errors 2000h, 4000h and 8000h, PID_3Step <b>cannot</b> approach the configured substitute output value.</p> <p>As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.</p>

**Behavior in version 1.0**

The ActivateRecoverMode variable determines the behavior in the event of an error in automatic and manual mode. ActivateRecoverMode is not effective during pretuning, fine tuning and transition time measurement.

ActivateRecoverMode	Description
FALSE	In the event of an error, PID_3Step switches to "Inactive" or "Approach substitute output value" operating mode. The controller is activated by a reset or a change in Retain.Mode.
TRUE	<p><b>Errors in automatic mode</b></p> <p>If errors occur frequently in automatic mode, this setting has a negative effect on the control response. In this case, check the ErrorBits parameter and eliminate the cause of the error.</p> <p>If one or more errors occur, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode:</p> <ul style="list-style-type: none"> <li>• 0002h: Invalid value at parameter Input_PER.</li> <li>• 0200h: Invalid value at parameter Input.</li> <li>• 0800h: Sampling time error</li> <li>• 1000h: Invalid value at parameter Setpoint.</li> <li>• 2000h: Invalid value at parameter Feedback_PER.</li> <li>• 4000h: Invalid value at parameter Feedback.</li> <li>• 8000h: Error in digital position feedback.</li> </ul> <p>With errors 2000h, 4000h and 8000h, PID_3Step <b>cannot</b> approach the configured substitute output value.</p> <p>As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.</p> <p><b>Errors in manual mode</b></p> <p>If one or more of the following errors occur, PID_3Step stays in manual mode:</p> <ul style="list-style-type: none"> <li>• 0002h: Invalid value at parameter Input_PER.</li> <li>• 0200h: Invalid value at parameter Input.</li> <li>• 0800h: Sampling time error</li> <li>• 1000h: Invalid value at parameter Setpoint.</li> <li>• 2000h: Invalid value at parameter Feedback_PER.</li> <li>• 4000h: Invalid value at parameter Feedback.</li> <li>• 8000h: Error in digital position feedback.</li> </ul> <p>With errors 2000h, 4000h and 8000h, you <b>cannot</b> move the valve to a suitable position.</p>

**See also**

PID\_3Step V1 static tags (Page 366)

Parameter State and Retain.Mode V1 (Page 374)

### 8.2.5.10 Tag Warning V1

If several warnings are pending simultaneously, their values are displayed with binary addition. The display of warning 0003, for example, indicates that the warnings 0001 and 0002 are pending simultaneously.

Warning (DW#16#...)	Description
0000	No warning pending.
0001	The point of inflection was not found during pretuning.
0002	Oscillation increased during fine tuning.
0004	The setpoint was limited to the configured limits.
0008	Not all the necessary controlled system properties were defined for the selected method of calculation. The PID parameters were instead calculated using the TuneRuleTIR = 3 method.
0010	The operating mode could not be changed because ManualEnable = TRUE.
0020	The cycle time of the calling OB limits the sampling time of the PID algorithm. Improve results by using shorter OB cycle times.
0040	The process value exceeded one of its warning limits.
0080	Invalid value at Retain.Mode. The operating mode is not switched.
0100	The manual value was limited to the limits of the controller output.
0200	The rule used for tuning produces an incorrect result, or is not supported.
0400	Method selected for transition time measurement not suitable for actuator. The transition time cannot be measured because the actuator settings do not match the selected measuring method.
0800	The difference between the current position and the new output value is too small for transition time measurement. This can produce incorrect results. The difference between the current output value and new output value must be at least 50% of the entire control range.
1000	The substitute output value cannot be reached because it is outside the output value limits.

The following warnings are deleted as soon as the cause is eliminated:

- 0004
- 0020
- 0040
- 0100

All other warnings are cleared with a rising edge at Reset.

**8.2.5.11 Tag SUT.State V1**

SUT.State	Name	Description
0	SUT_INIT	Initialize pretuning
50	SUT_TPDN	Determine start position without position feedback
100	SUT_STDABW	Calculate the standard deviation
200	SUT_GET_POI	Find the point of inflection
300	SUT_GET_RISETM	Determine the rise time
9900	SUT_IO	Pretuning successful
1	SUT_NIO	Pretuning not successful

**8.2.5.12 Tag TIR.State V1**

TIR.State	Name	Description
-100	TIR_FIRST_SUT	Fine tuning is not possible. Pretuning will be executed first.
0	TIR_INIT	Initialize fine tuning
200	TIR_STDABW	Calculate the standard deviation
300	TIR_RUN_IN	Attempt to reach the setpoint with the maximum or minimum output value
400	TIR_CTRLN	Attempt to reach the setpoint with the existing PID parameters (if pretuning has been successful)
500	TIR_OSZIL	Determine oscillation and calculate parameters
9900	TIR_IO	Fine tuning successful
1	TIR_NIO	Fine tuning not successful

## 8.3 PID\_Temp

### 8.3.1 New features of PID\_Temp

#### PID\_Temp V1.1

- **Response of the output value on switchover from "Inactive" operating mode to "Automatic mode"**

The new option IntegralResetMode = 4 was added and defined as default. With IntegralResetMode = 4, the integral action is automatically pre-assigned when switching from "Inactive" operating mode to "Automatic mode" so that a control deviation results in a jump of the PID output value with identical sign.

- **Initialization of the integral action in automatic mode**

The integral action can be initialized in automatic mode with the tags OverwriteInitialOutputValue and PIDCtrl.PIDInit. This simplifies the use of PID\_Temp for override controls.

### 8.3.2 Compatibility with CPU and FW

The following table shows which version of PID\_Temp can be used on which CPU.

CPU	FW	PID_Temp
S7-1200	V4.2 or higher	V1.1 V1.0
	V4.1	V1.0
S7-1500	V2.0 or higher	V1.1 V1.0
	V1.7 to V1.8	V1.0

### 8.3.3 PID\_Temp

#### 8.3.3.1 Description of PID\_Temp

##### Description

The PID\_Temp instruction provides a PID controller with integrated tuning for temperature processes. PID\_Temp can be used for pure heating or heating/cooling applications.

The following operating modes are possible:

- Inactive
- Pretuning
- Fine tuning
- Automatic mode
- Manual mode
- Substitute output value with error monitoring

For a more detailed description of the operating modes, see the State parameter.

##### PID algorithm

PID\_Temp is a PIDT1 controller with anti-windup and weighting of the proportional and derivative actions. The PID algorithm operates according to the following equation (control zone and dead zone deactivated):

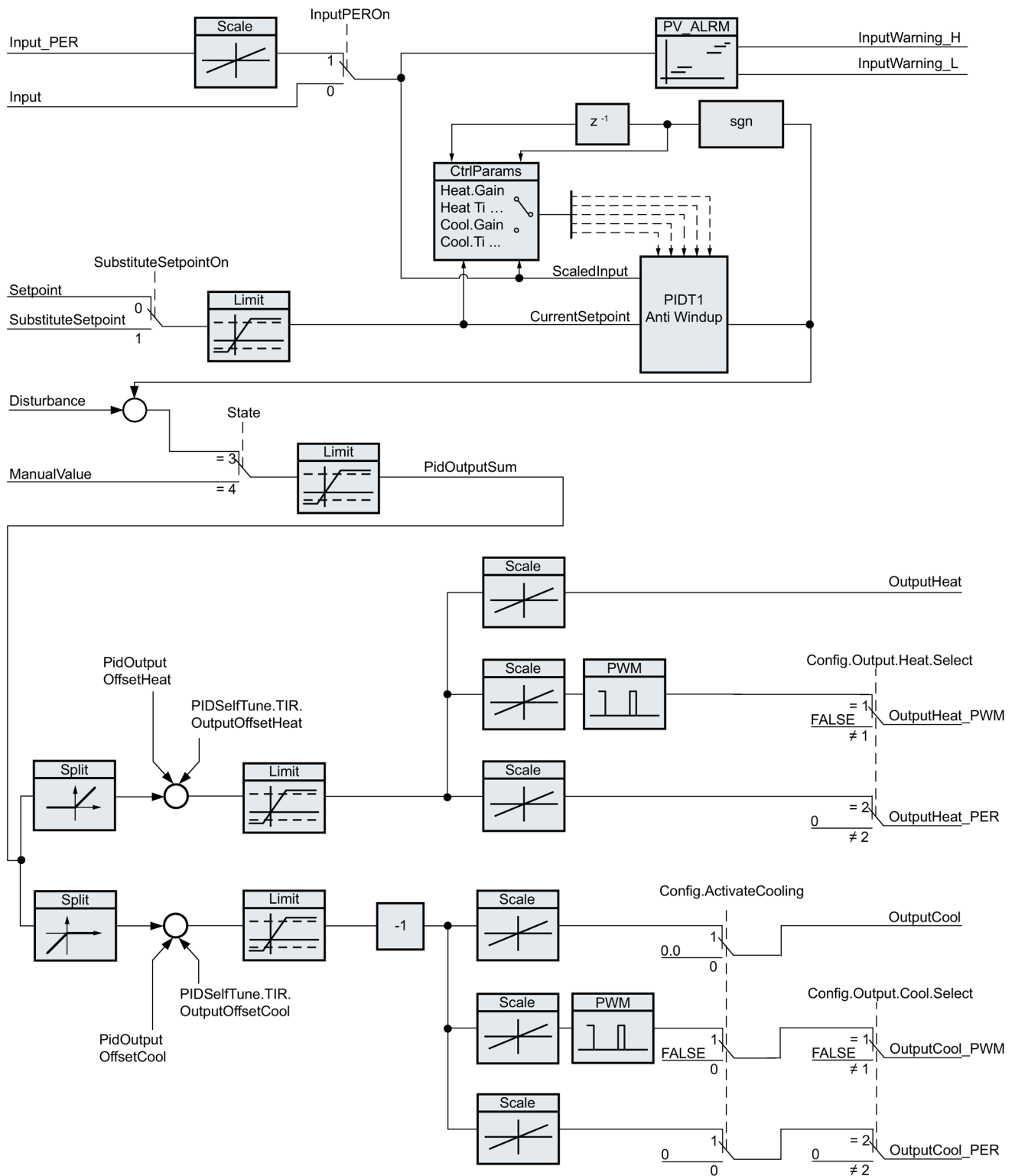
$$y = K_p \left[ (b \cdot w - x) + \frac{1}{T_i \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

The table below shows the meaning of the icons used in the equation and in the subsequent figures.

Icon	Description	Associated parameters of the PID_Temp instruction
y	Output value of the PID algorithm	-
K <sub>p</sub>	Proportional gain	Retain.CtrlParams.Heat.Gain Retain.CtrlParams.Cool.Gain CoolFactor
s	Laplace operator	-
b	Proportional action weighting	Retain.CtrlParams.Heat.PWeighting Retain.CtrlParams.Cool.PWeighting
w	Setpoint	CurrentSetpoint
x	Process value	ScaledInput
T <sub>i</sub>	Integral action time	Retain.CtrlParams.Heat.Ti Retain.CtrlParams.Cool.Ti

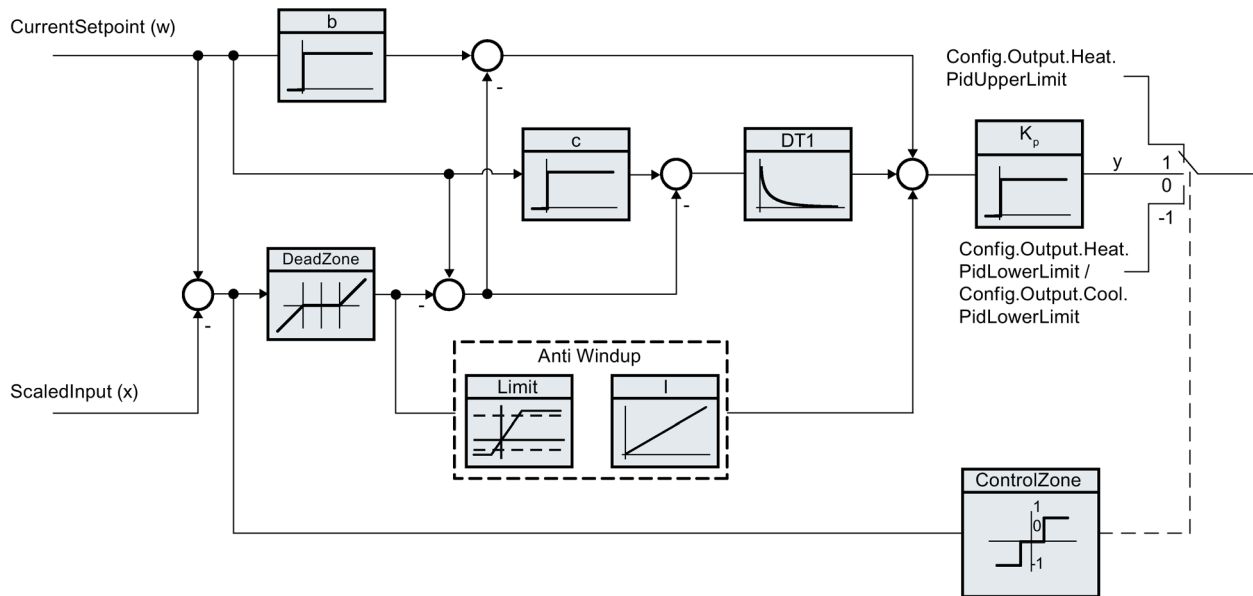
Icon	Description	Associated parameters of the PID_Temp instruction
T <sub>D</sub>	Derivative action time	Retain.CtrlParams.Heat.Td Retain.CtrlParams.Cool.Td
a	Derivative delay coefficient (derivative delay $T_1 = a \times T_D$ )	Retain.CtrlParams.Heat.TdFiltRatio Retain.CtrlParams.Cool.TdFiltRatio
c	Derivative action weighting	Retain.CtrlParams.Heat.DWeighting Retain.CtrlParams.Cool.DWeighting
DeadZone	Dead zone width	Retain.CtrlParams.Heat.DeadZone Retain.CtrlParams.Cool.DeadZone
ControlZone	Control zone width	Retain.CtrlParams.Heat.ControlZone Retain.CtrlParams.Cool.ControlZone

PID\_Temp block diagram





### Block diagram of PIDT1 with anti-windup



### Call

PID\_Temp is called in the constant time scale of a cyclic interrupt OB.

If you call PID\_Temp as a multi-instance DB, no technology object is created. No parameter assignment interface or commissioning interface is available. You must assign parameters for PID\_Temp directly in the multi-instance DB and commission it via a watch table.

### Download to device

The process values of retentive tags are only updated when you download PID\_Temp completely.

Download technology object to device (Page 72)

### Startup

When the CPU starts up, PID\_Temp starts in the operating mode that is saved in the Mode in/out parameter. To switch to "Inactive" operating mode during startup, set RunModeByStartup = FALSE.

**Reaction to error**

The behavior in the case of an error is determined by the tags SetSubstituteOutput and ActivateRecoverMode. If ActivateRecoverMode = TRUE, the behavior also depends on the error that occurred.

SetSubstitute-Output	ActivateRecoverMode	Configuration editor > Basic settings of output > Set PidOutputSum to	Reaction
Not relevant	FALSE	Zero (inactive)	Switch to "Inactive" (State = 0) mode The output value of the PID algorithm and all outputs for heating and cooling are set to 0. The scaling of the outputs for heating and cooling is not active.
FALSE	TRUE	Current value for error while error is pending	Switch to "Substitute output value with error monitoring" mode (State = 5) The current output value is transferred to the actuator while the error is pending.
TRUE	TRUE	Substitute output value while error is pending	Switch to "Substitute output value with error monitoring" mode (State = 5) The value at SubstituteOutput is transferred to the actuator while the error is pending.

In manual mode, PID\_Temp uses ManualValue as output value, unless ManualValue is invalid.

- If ManualValue is invalid, SubstituteOutput is used.
- If ManualValue and SubstituteOutput are invalid, Config.Output.Heat.PidLowerLimit is used.

The Error parameter indicates if an error is pending. When the error is no longer pending, Error = FALSE. The ErrorBits parameter shows which errors have occurred. ErrorBits is reset by a rising edge at Reset or ErrorAck.

### 8.3.3.2 Mode of operation of PID\_Temp

#### Monitoring process value limits

You specify the high limit and low limit of the process value in the Config.InputUpperLimit and Config.InputLowerLimit tags. If the process value is outside these limits, an error occurs (ErrorBits = 0000001h).

You specify a high and low warning limit of the process value in the Config.InputUpperWarning and Config.InputLowerWarning tags. If the process value is outside these warning limits, a warning occurs (Warning = 0000040h), and the InputWarning\_H or InputWarning\_L output parameter changes to TRUE.

#### Limiting the setpoint

You specify a high limit and low limit of the setpoint in the Config.SetpointUpperLimit and Config.SetpointLowerLimit tags. PID\_Temp automatically limits the setpoint to the process value limits. You can limit the setpoint to a smaller area. PID\_Temp checks whether this area is within the process value limits. If the setpoint is outside these limits, the high or low limit is used as the setpoint, and output parameter SetpointLimit\_H or SetpointLimit\_L is set to TRUE.

The setpoint is limited in all operating modes.

#### Substitute setpoint

You can specify a substitute setpoint at the SubstituteSetpoint tag and activate it with SubstituteSetpointOn = TRUE. In this way, you can temporarily specify the setpoint directly, for example for a slave controller in a cascade, without having to change the user program. The limits set for the setpoint also apply to the substitute setpoint.

#### Heating and cooling

With the default setting, PID\_Temp only uses the outputs for heating (OutputHeat, OutputHeat\_PWM, OutputHeat\_PER). The output value of the PID algorithm (PidOutputSum) is scaled and output at the outputs for heating. You specify with Config.Output.Heat.Select if OutputHeat\_PWM or OutputHeat\_PER is calculated. OutputHeat is always calculated.

With Config.ActivateCooling = TRUE, you can also activate the outputs for cooling (OutputCool, OutputCool\_PWM, OutputCool\_PER). Positive output values of the PID algorithm (PidOutputSum) are scaled and output at the outputs for heating. Negative output values of the PID algorithm are scaled and output at the outputs for cooling. You specify with Config.Output.Cool.Select if OutputCool\_PWM or OutputCool\_PER is calculated. OutputCool is always calculated.

Two methods are available to calculate the PID output value with activated cooling:

- Cooling factor (Config.AdvancedCooling = FALSE):  
 The output value calculation for cooling takes place with the PID parameters for heating, taking into consideration the configurable cooling factor Config.CoolFactor. This method is suitable if the heating and cooling actuators have a similar time response but different gains. If this method is selected, pretuning and fine tuning for cooling as well as the PID parameter set for cooling are not available. You can only execute the tuning for heating.
- PID parameter switching (Config.AdvancedCooling = TRUE):  
 The output value calculation for cooling takes place by means of a separate PID parameter set. Based on the calculated output value and the control deviation, the PID algorithm decides whether the PID parameter for heating or cooling is used. This method is suitable if the heating and cooling actuator have different time responses and different gains. Pretuning and fine tuning for cooling are only available if this method is selected.

### Output value limits and scaling

Depending on the operating mode, the PID output value (PidOutputSum) is calculated automatically by the PID algorithm or defined by the manual value (ManualValue) or the configured substitute output value (SubstituteOutput).

The PID output value is limited according to the configuration:

- If cooling is deactivated (Config.ActivateCooling = FALSE), Config.Output.Heat.PidUpperLimit is the high limit and Config.Output.Heat.PidLowerLimit the low limit.
- If cooling is activated (Config.ActivateCooling = TRUE), Config.Output.Heat.PidUpperLimit is the high limit and Config.Output.Cool.PidLowerLimit the low limit.

The PID output value is scaled and output at the outputs for heating and cooling. Scaling can be defined separately for each output and is specified in the structures Config.Output.Heat or Config.Output.Cool with 2 value pairs each:

Output	Value pair	Parameter
OutputHeat	Value pair 1	PID output value high limit (heating) Config.Output.Heat.PidUpperLimit, Scaled high output value (heating) Con- fig.Output.Heat.UpperScaling
	Value pair 2	PID output value low limit (heating) Config.Output.Heat.PidLowerLimit, Scaled low output value (heating) Con- fig.Output.Heat.LowerScaling
OutputHeat_PWM	Value pair 1	PID output value high limit (heating) Config.Output.Heat.PidUpperLimit, Scaled high PWM output value (heating) Config.Output.Heat.PwmUpperScaling

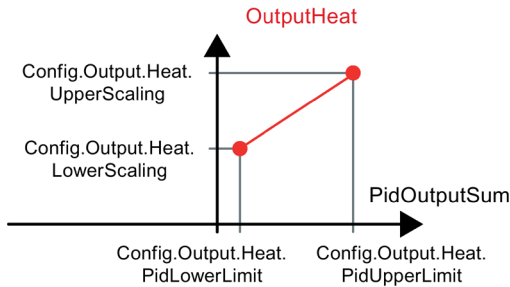
Output	Value pair	Parameter
	Value pair 2	PID output value low limit (heating) Config.Output.Heat.PidLowerLimit, Scaled low PWM output value (heating) Config.Output.Heat.PwmLowerScaling
OutputHeat_PER	Value pair 1	PID output value high limit (heating) Config.Output.Heat.PidUpperLimit, Scaled high analog output value (heating) Config.Output.Heat.PerUpperScaling
	Value pair 2	PID output value low limit (heating) Config.Output.Heat.PidLowerLimit, Scaled low analog output value (heating) Config.Output.Heat.PerLowerScaling
OutputCool	Value pair 1	PID output value low limit (cooling) Config.Output.Cool.PidLowerLimit, Scaled high output value (cooling) Config.Output.Cool.UpperScaling
	Value pair 2	PID output value high limit (cooling) Config.Output.Cool.PidUpperLimit, Scaled low output value (cooling) Config.Output.Cool.LowerScaling
OutputCool_PWM	Value pair 1	PID output value low limit (cooling) Config.Output.Cool.PidLowerLimit, Scaled high PWM output value (cooling) Config.Output.Cool.PwmUpperScaling
	Value pair 2	PID output value high limit (cooling) Config.Output.Cool.PidUpperLimit, Scaled low PWM output value (cooling) Config.Output.Cool.PwmLowerScaling
OutputCool_PER	Value pair 1	PID output value low limit (cooling) Config.Output.Cool.PidLowerLimit, Scaled high analog output value (cooling) Config.Output.Cool.PerUpperScaling
	Value pair 2	PID output value high limit (cooling) Config.Output.Cool.PidUpperLimit, Scaled low analog output value (cooling) Config.Output.Cool.PerLowerScaling

If cooling is activated (Config.ActivateCooling = TRUE), Config.Output.Heat.PidLowerLimit must have the value 0.0.

Config.Output.Cool.PidUpperLimit must always have the value 0.0.

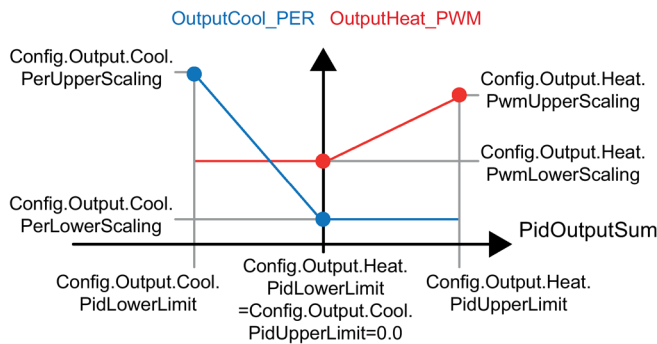
**Example:**

Output scaling when using output OutputHeat (cooling deactivated; Config.Output.Heat.PidLowerLimit may be unequal to 0.0):



**Example:**

Output scaling when using output OutputHeat\_PWM and OutputCool\_PER (cooling activated; Config.Output.Heat.PidLowerLimit must be 0.0):



With the exception of the "Inactive" operating mode, the value at an output is always located between its scaled high output value and scaled low output value, for example, for OutputHeat always between Config.Output.Heat.UpperScaling and Config.Output.Heat.LowerScaling.

If you want to limit the value at the associated output, you therefore have to adapt these scaling values as well.

**Cascading**

PID\_Temp supports you when you use cascade control (see: Program creation (Page 200)).

**Substitute output value**

In the event of an error, PID\_Temp can output a substitute output value that you define at the SubstituteOutput tag. The substitute output value must be within the limits for the PID output value. The values at the outputs for heating and cooling resulting from the substitute output value are the result of the configured output scaling.

## Monitoring signal validity

The values of the following parameters are monitored for validity when used:

- Setpoint
- SubstituteSetpoint
- Input
- Input\_PER
- Disturbance
- ManualValue
- SubstituteOutput
- PID parameters in the structures Retain.CtrlParams.Heat and Retain.CtrlParams.Cool.

## Monitoring the sampling time PID\_Temp

Ideally, the sampling time is equivalent to the cycle time of the cyclic interrupt OB. The PID\_Temp instruction measures the time interval between two calls. This is the current sampling time. On every switchover of operating mode and during the initial startup, the mean value is formed from the first 10 sampling times. Too great a difference between the current sampling time and this mean value triggers an error (Error = 0000800h).

The error occurs during tuning if:

- New mean value  $\geq 1.1 \times$  old mean value
- New mean value  $\leq 0.9 \times$  old mean value

The error occurs in automatic mode if:

- New mean value  $\geq 1.5 \times$  old mean value
- New mean value  $\leq 0.5 \times$  old mean value

If you deactivate the sampling time monitoring (CycleTime.EnMonitoring = FALSE), you can also call PID\_Temp in OB1. You must then accept a lower control quality due to the deviating sampling time.

## Sampling time of the PID algorithm

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the cycle time of the cyclic interrupt OB (sampling time PID\_Temp). All other functions of the PID\_Temp are executed at every call.

If cooling and PID parameter switching are activated, PID\_Temp uses a separate sampling time of the PID algorithm for heating and cooling. In all other configurations, only the sampling time of the PID algorithm for heating is used.

If you use OutputHeat\_PWM or OutputCool\_PWM, the sampling time of the PID algorithm is used as time period of the pulse width modulation. The accuracy of the output signal is determined by the ratio of the PID algorithm sampling time to the cycle time of the OB. The cycle time should be no more than a tenth of the PID algorithm sampling time.

If the PID algorithm sampling time and thus the time period of the pulse width modulation is very high when you use OutputHeat\_PWM or OutputCool\_PWM, you can define a deviating shorter period duration at the Config.Output.Heat.PwmPeriode or Config.Output.Cool.PwmPeriode parameters to improve the smoothness of the process value.

## Control logic

PID\_Temp can be used for heating or heating/cooling applications and always works with normal control logic.

An increase of the PID output value (PidOutputSum) is intended to increase the process value. The values at the outputs for heating and cooling resulting from the PID output value are the result of the configured output scaling.

An inverted control logic or negative proportional gain are not supported.

If you only need an output value for your application in which an increase is to reduce the process value (for example, discharge control), you can use PID\_Compact with inverted control logic.



### 8.3.3.3 Input parameters of PID\_Temp

Parameter	Data type	Default	Description
Setpoint	REAL	0.0	Setpoint of the PID controller in automatic mode Valid range of values: Config.SetpointUpperLimit ≥ Setpoint ≥ Config.SetpointLowerLimit Config.InputUpperLimit ≥ Setpoint ≥ Config.InputLowerLimit
Input	REAL	0.0	A tag of the user program is used as source for the process value. If you are using the Input parameter, Config.InputPerOn = FALSE must be set.
Input_PER	INT	0	An analog input is used as the source of the process value. If you are using the Input_PER parameter, Config.InputPerOn = TRUE must be set.
Disturbance	REAL	0.0	Disturbance variable or precontrol value
ManualEnable	BOOL	FALSE	<ul style="list-style-type: none"> <li>A FALSE -&gt; TRUE edge activates "Manual mode", whileState = 4, Mode remains unchanged.</li> </ul> <p>As long as ManualEnable = TRUE, you cannot change the operating mode via a rising edge at ModeActivate or use the commissioning dialog.</p> <ul style="list-style-type: none"> <li>A TRUE -&gt; FALSE edge activates the operating mode that is specified by Mode.</li> </ul> <p>We recommend that you change the operating mode using Mode and ModeActivate only.</p>
ManualValue	REAL	0.0	Manual value This value is used in manual mode as PID output value (PidOutputSum). The values at the outputs for heating and cooling resulting from this manual value are the result of the configured output scaling (structures Config.Output.Heat and Config.Output.Cool). For controllers with activated cooling output (Config.ActivateCooling = TRUE), define: <ul style="list-style-type: none"> <li>a positive manual value to output the value at the outputs for heating</li> <li>a negative manual value to output the value at the outputs for cooling</li> </ul> <p>The permitted value range is determined by the configuration.</p> <ul style="list-style-type: none"> <li>Cooling output deactivated (Config.ActivateCooling = FALSE): Config.Output.Heat.PidUpperLimit ≥ ManualValue ≥ Config.Output.Heat.PidLowerLimit</li> <li>Cooling output activated (Config.ActivateCooling = TRUE): Config.Output.Heat.PidUpperLimit ≥ ManualValue ≥ Config.Output.Cool.PidLowerLimit</li> </ul>
ErrorAck	BOOL	FALSE	<ul style="list-style-type: none"> <li>FALSE -&gt; TRUE edge</li> </ul> <p>ErrorBits and Warning are reset.</p>

8.3 PID\_Temp

Parameter	Data type	Default	Description
Reset	BOOL	FALSE	<p>Restarts the controller.</p> <ul style="list-style-type: none"> <li>• FALSE -&gt; TRUE edge                             <ul style="list-style-type: none"> <li>- Switch to "Inactive" mode</li> <li>- ErrorBits and Warning are reset.</li> </ul> </li> <li>• As long as Reset = TRUE,                             <ul style="list-style-type: none"> <li>- PID_Temp remains in "Inactive" mode (State = 0).</li> <li>- you cannot change the operating mode with Mode and ModeActivate or ManualEnable</li> <li>- You cannot use the commissioning dialog.</li> </ul> </li> <li>• TRUE -&gt; FALSE edge                             <ul style="list-style-type: none"> <li>- If ManualEnable = FALSE, PID_Temp switches to the operating mode that is saved in Mode.</li> <li>- If Mode = 3 (automatic mode), the integral action is treated as configured with the tag IntegralResetMode.</li> </ul> </li> </ul>
ModeActivate	BOOL	FALSE	<ul style="list-style-type: none"> <li>• FALSE -&gt; TRUE edge</li> </ul> <p>PID_Temp switches to the operating mode that is saved at the Mode input.</p>

### 8.3.3.4 Output parameters of PID\_Temp

Parameter	Data type	Default	Description
ScaledInput	REAL	0.0	Scaled process value
OutputHeat	REAL	0.0	Output value (heating) in REAL format The PID output value (PidOutputSum) is scaled with the two value pairs Config.Output.Heat.PidUpperLimit, Config.Output.Heat.UpperScaling and Config.Output.Heat.PidLowerLimit, Config.Output.Heat.LowerScaling and output in REAL format at OutputHeat. OutputHeat is always calculated.
OutputCool	REAL	0.0	Output value (cooling) in REAL format The PID output value (PidOutputSum) is scaled with the two value pairs Config.Output.Cool.PidUpperLimit, Config.Output.Cool.LowerScaling and Config.Output.Cool.PidLowerLimit, Config.Output.Cool.UpperScaling and output in REAL format at OutputCool. OutputCool is only calculated if the cooling output is activated (Config.ActivateCooling = TRUE).
OutputHeat_PER	INT	0	Analog output value (heating) The PID output value (PidOutputSum) is scaled with the two value pairs Config.Output.Heat.PidUpperLimit, Config.Output.Heat.PerUpperScaling and Config.Output.Heat.PidLowerLimit, Config.Output.Heat.PerLowerScaling and output as analog value at OutputHeat_PER. OutputHeat_PER is only calculated if Config.Output.Heat.Select = 2.
OutputCool_PER	INT	0	Analog output value (cooling) The PID output value (PidOutputSum) is scaled with the two value pairs Config.Output.Cool.PidUpperLimit, Config.Output.Cool.PerLowerScaling and Config.Output.Cool.PidLowerLimit, Config.Output.Cool.PerUpperScaling and output as analog value at OutputCool_PER. OutputCool_PER is only calculated if the cooling output is activated (Config.ActivateCooling = TRUE) and Config.Output.Cool.Select = 2.
OutputHeat_PWM	BOOL	FALSE	Pulse-width modulated output value (heating) The PID output value (PidOutputSum) is scaled with the two value pairs Config.Output.Heat.PidUpperLimit, Config.Output.Heat.PwmUpperScaling and Config.Output.Heat.PidLowerLimit, Config.Output.Heat.PwmLowerScaling and output as pulse-width modulated value (variable switch on and switch off times) at OutputHeat_PWM. OutputHeat_PWM is only calculated if Config.Output.Heat.Select = 1.
OutputCool_PWM	BOOL	FALSE	Pulse-width modulated output value (cooling) The PID output value (PidOutputSum) is scaled with the two value pairs Config.Output.Cool.PidUpperLimit, Config.Output.Cool.PwmLowerScaling and Config.Output.Cool.PidLowerLimit, Config.Output.Cool.PwmUpperScaling and output as pulse-width modulated value (variable switch on and switch off times) at OutputCool_PWM. OutputCool_PWM is only calculated if the cooling output is activated (Config.ActivateCooling = TRUE) and Config.Output.Cool.Select = 1.
SetpointLimit_H	BOOL	FALSE	If SetpointLimit_H = TRUE, the absolute setpoint high limit is reached (Setpoint ≥ Config.SetpointUpperLimit) or Setpoint ≥ Config.InputUpperLimit. The setpoint high limit is the minimum of Config.SetpointUpperLimit and Config.InputUpperLimit.

8.3 PID\_Temp

Parameter	Data type	Default	Description
SetpointLimit_L	BOOL	FALSE	If SetpointLimit_L = TRUE, the absolute setpoint low limit is reached (Setpoint ≤ Config.SetpointLowerLimit) or Setpoint ≤ Config.InputLowerLimit. The setpoint low limit is the maximum of Config.SetpointLowerLimit and Config.InputLowerLimit.
InputWarning_H	BOOL	FALSE	If InputWarning_H = TRUE, the process value has reached or exceeded the warning high limit (ScaledInput ≥ Config.InputUpperWarning).
InputWarning_L	BOOL	FALSE	If InputWarning_L = TRUE, the process value has reached or fallen below the warning low limit (ScaledInput ≤ Config.InputLowerWarning).
State	INT	0	The PID_Temp state and mode parameters (Page 440) shows the current operating mode of the PID controller. You can change the operating mode using the input parameter Mode and a rising edge at ModeActivate. For pre-tuning and fine tuning, you specify with Heat.EnableTuning and Cool.EnableTuning whether tuning takes place for heating or cooling. <ul style="list-style-type: none"> <li>• State = 0: Inactive</li> <li>• State = 1: Pretuning</li> <li>• State = 2: Fine tuning</li> <li>• State = 3: Automatic mode</li> <li>• State = 4: Manual mode</li> <li>• State = 5: Substitute output value with error monitoring</li> </ul>
Error	BOOL	FALSE	If Error = TRUE, at least one error message is pending in this cycle.
ErrorBits	DWORD	DW#16#0	The PID_Temp ErrorBits parameter (Page 450) shows the pending error messages. ErrorBits is retentive and is reset with a rising edge at Reset or ErrorAck.

### 8.3.3.5 In/out parameters of PID\_Temp V2

Parameter	Data type	Default	Description
Mode	INT	4	<p>At Mode, specify the operating mode to which PID_Temp is to switch. Options are:</p> <ul style="list-style-type: none"> <li>• Mode = 0: Inactive</li> <li>• Mode = 1: Pretuning</li> <li>• Mode = 2: Fine tuning</li> <li>• Mode = 3: Automatic mode</li> <li>• Mode = 4: Manual mode</li> </ul> <p>The operating mode is activated by:</p> <ul style="list-style-type: none"> <li>• Rising edge at ModeActivate</li> <li>• Falling edge at Reset</li> <li>• Falling edge at ManualEnable</li> <li>• Cold restart of CPU if RunModeByStartup = TRUE</li> </ul> <p>For pretuning and fine tuning, you specify with Heat.EnableTuning and Cool.EnableTuning whether tuning takes place for heating or cooling. Mode is retentive.</p> <p>A detailed description of the operating modes can be found in State and Mode parameters (Page 440).</p>
Master	DWORD	DW#16#0	<p>Interface for cascade control</p> <p>If this PID_Temp instance is used as slave controller in a cascade (Config.Cascade.IsSlave = TRUE), assign the Master parameter at the instruction call with the Slave parameter of the master controller.</p> <p>Example: Call of a slave controller "PID_Temp_2" with master controller "PID_Temp_1" in SCL:</p> <pre>----- "PID_Temp_2"(Master := "PID_Temp_1".Slave, Setpoint := "PID_Temp_1".OutputHeat); -----</pre> <p>You use this interface to exchange slave controller information about operating mode, limit and substitute setpoint with your master controller. Keep in mind that the call of the master controller has to take place before the call of the slave controller in the same cyclic interrupt OB.</p> <p>Assignment:</p> <ul style="list-style-type: none"> <li>• Bits 0 to 15: Unassigned</li> <li>• Bits 16 to 23 – Limit counter: A slave controller whose output value is limited increments this counter. Depending on the configured number of slaves (Config.Cascade.CountSlaves) and of the anti-windup mode (Config.Cascade.AntiWindUpMode), the master controller reacts accordingly.</li> <li>• Bit 24 – Automatic mode of the slave controllers: TRUE, if all slave controllers are in automatic mode</li> <li>• Bit 25 – Substitute setpoint of the slave controllers: TRUE, if a slave controller has activated the substitute setpoint (SubstituteSetpointOn = TRUE)</li> </ul>

Parameter	Data type	Default	Description
Slave	DWORD	DW#16#0	Interface for cascade control You use this interface to exchange slave controller information about operating mode, limit and substitute setpoint with your master controller. See description of Master parameter

**See also**

PID\_Temp state and mode parameters (Page 440)

Program creation (Page 200)

Cascade control with PID\_Temp (Page 198)

### 8.3.3.6 PID\_Temp static tags

You must not change tags that are not listed. These are used for internal purposes only.

Tag	Data type	Default	Description
IntegralResetMode	Int	V1.0: 1, V1.1 or higher: 4	<p>The IntegralResetMode tag (Page 458) determines how the integral action PIDCtrl.IOutputOld is pre-assigned when switching from "Inactive" operating mode to "Automatic mode".</p> <p>This setting only works for one cycle.</p> <ul style="list-style-type: none"> <li>• IntegralResetMode = 0: Smooth</li> <li>• IntegralResetMode = 1: Delete</li> <li>• IntegralResetMode = 2: Hold</li> <li>• IntegralResetMode = 3: Pre-assign</li> <li>• IntegralResetMode = 4: Like setpoint change (only for PID_Temp with version ≥ 1.1)</li> </ul>
OverwriteInitialOutputValue	REAL	0.0	<p>If one of the following conditions is met, the integral action PIDCtrl.IOutputOld is pre-assigned automatically as if PIDOutputSum = OverwriteInitialOutputValue in the previous cycle:</p> <ul style="list-style-type: none"> <li>• IntegralResetMode = 3 when switching from "Inactive" operating mode to "Automatic mode"</li> <li>• TRUE -&gt; FALSE edge at parameter Reset and parameter Mode = 3</li> <li>• PIDCtrl.PIDInit = TRUE in "Automatic mode" (available as of PID_Temp version 1.1)</li> </ul>
RunModeByStartup	BOOL	TRUE	<p>Activate operating mode at Mode parameter after CPU restart</p> <ul style="list-style-type: none"> <li>• If RunModeByStartup = TRUE, PID_Temp starts in the operating mode saved in the Mode parameter after CPU startup.</li> <li>• If RunModeByStartup = FALSE, PID_Temp remains in "Inactive" mode after CPU startup.</li> </ul>
LoadBackUp	BOOL	FALSE	<p>If LoadBackUp = TRUE, the last set of PID parameters is reloaded from the CtrlParamsBackUp structure. The set was saved prior to the last tuning. LoadBackUp is automatically set back to FALSE. The acceptance is bumpless.</p>

Tag	Data type	Default	Description
SetSubstituteOutput	BOOL	TRUE	<p>Selection of the output value while an error is pending (State = 5):</p> <ul style="list-style-type: none"> <li>• If SetSubstituteOutput = TRUE and ActivateRecoverMode = TRUE, the configured substitute output value SubstituteOutput is output as PID output value as long as an error is pending.</li> <li>• If SetSubstituteOutput = FALSE and ActivateRecoverMode = TRUE, the actuator remains at the current PID output value as long as an error is pending.</li> <li>• If ActivateRecoverMode = FALSE, SetSubstituteOutput is not effective.</li> <li>• If SubstituteOutput is invalid (ErrorBits = 0020000h), the substitute output value cannot be output. In this case, the low limit of the PID output value for heating (Config.Output.Heat.PidLowerLimit) is used as PID output value.</li> </ul>
PhysicalUnit	INT	0	<p>Unit of measurement of the process value and setpoint, e.g., °C, or °F.</p> <p>This parameter is used for display in the editors and does not influence the control algorithm.</p>
PhysicalQuantity	INT	0	<p>Physical quantity of the process value and setpoint, e.g., temperature.</p> <p>This parameter is used for display in the editors and does not influence the control algorithm.</p>
ActivateRecoverMode	BOOL	TRUE	<p>The ActivateRecoverMode tag determines the reaction to error.</p>
Warning	DWORD	0	<p>The Warning tag shows the warnings since Reset = TRUE or ErrorAck = TRUE. Warning is retentive.</p>
Progress	REAL	0.0	<p>Progress of current tuning phase as a percentage (0.0 - 100.0)</p>
CurrentSetpoint	REAL	0.0	<p>CurrentSetpoint always displays the currently effective setpoint. This value is frozen during tuning.</p>
CancelTuningLevel	REAL	10.0	<p>Permissible fluctuation of setpoint during tuning. Tuning is not canceled until:</p> <ul style="list-style-type: none"> <li>• Setpoint &gt; CurrentSetpoint + CancelTuningLevel</li> <li>or</li> <li>• Setpoint &lt; CurrentSetpoint - CancelTuningLevel</li> </ul>



Tag	Data type	Default	Description
SubstituteOutput	REAL	0.0	<p>The substitute output value is used as PID output value as long as the following conditions are met:</p> <ul style="list-style-type: none"> <li>• One or more errors are pending in automatic mode for which ActivateRecoverMode is in effect</li> <li>• SetSubstituteOutput = TRUE</li> <li>• ActivateRecoverMode = TRUE</li> </ul> <p>The values at the outputs for heating and cooling resulting from the substitute output value are the result of the configured output scaling (structures Config.Output.Heat and Config.Output.Cool).</p> <p>For controllers with activated cooling output (Config.ActivateCooling = TRUE), define:</p> <ul style="list-style-type: none"> <li>• a positive substitute output value to output the value at the outputs for heating</li> <li>• a negative substitute output value to output the value at the outputs for cooling</li> </ul> <p>The permitted value range is determined by the configuration.</p> <ul style="list-style-type: none"> <li>• Cooling output deactivated (Config.ActivateCooling = FALSE): Config.Output.Heat.PidUpperLimit ≥ SubstituteOutput ≥ Config.Output.Heat.PidLowerLimit</li> <li>• Cooling output activated (Config.ActivateCooling = TRUE): Config.Output.Heat.PidUpperLimit ≥ SubstituteOutput ≥ Config.Output.Cool.PidLowerLimit</li> </ul>
PidOutputSum	REAL	0.0	<p>PID output value</p> <p>PidOutputSum displays the output value of the PID algorithm. Depending on the operating mode, it is either calculated automatically or defined by the manual value or the configured substitute output value.</p> <p>The values at the outputs for heating and cooling resulting from the PID output value are the result of the configured output scaling (structures Config.Output.Heat and Config.Output.Cool).</p> <p>The PidOutputSum is limited as defined in the configuration.</p> <ul style="list-style-type: none"> <li>• Cooling output deactivated (Config.ActivateCooling = FALSE): Config.Output.Heat.PidUpperLimit ≥ PidOutputSum ≥ Config.Output.Heat.PidLowerLimit</li> <li>• Cooling output activated (Config.ActivateCooling = TRUE): Config.Output.Heat.PidUpperLimit ≥ PidOutputSum ≥ Config.Output.Cool.PidLowerLimit</li> </ul>

Tag	Data type	Default	Description
PidOutputOffsetHeat	REAL	0.0	<p>Offset of the PID output value heating</p> <p>PidOutputOffsetHeat is added to the value that results from PidOutputSum for the heating branch. Enter a positive value for PidOutputOffsetHeat to receive a positive offset at the outputs for heating.</p> <p>The resulting values at the outputs for heating are the result of the configured output scaling (Config.Output.Heat structure).</p> <p>This offset can be used for actuators which need a fixed minimum value, for example, fans with minimum speed.</p>
PidOutputOffsetCool	REAL	0.0	<p>Offset of the PID output value cooling</p> <p>PidOutputOffsetCool is added to the value that results from PidOutputSum for the cooling branch. Enter a negative value for PidOutputOffsetCool to receive a positive offset at the outputs for cooling.</p> <p>The resulting values at the outputs for cooling are the result of the configured output scaling ( Config.Output.Cool structure).</p> <p>This offset can be used for actuators which need a fixed minimum value, for example, fans with minimum speed.</p>
SubstituteSetpointOn	BOOL	FALSE	<p>Activates the substitute setpoint as controller setpoint.</p> <ul style="list-style-type: none"> <li>FALSE = the Setpoint parameter is used.</li> <li>TRUE = the SubstituteSetpoint parameter is used as setpoint</li> </ul> <p>SubstituteSetpointOn can be used to specify the setpoint of a slave controller in a cascade directly without having to change the user program.</p>
SubstituteSetpoint	REAL	0.0	<p>Substitute setpoint</p> <p>If SubstituteSetpointOn = TRUE, the SubstituteSetpoint parameter is used as setpoint.</p> <p>Valid range of values:            Config.SetpointUpperLimit ≥ SubstituteSetpoint ≥ Config.SetpointLowerLimit, Config.InputUpperLimit ≥ SubstituteSetpoint ≥ Config.InputLowerLimit</p>
DisableCooling	BOOL	FALSE	<p>DisableCooling = TRUE deactivates the cooling branch for heating/cooling controllers (Config.ActivateCooling = TRUE) in Automatic mode by setting PidOutputSum to 0.0 as low limit.</p> <p>PidOutputOffsetCool and the output scaling for the cooling outputs remain active.</p> <p>DisableCooling can be used for tuning of multi-zone applications to temporarily deactivate the cooling branch as long as all controllers have not completed their tuning yet.</p> <p>This parameter is set/reset by the user manually and is not automatically reset by the PID_Temp instruction.</p>

Tag	Data type	Default	Description
AllSlaveAutomaticState	BOOL	FALSE	<p>If this PID_Temp instance is used as master controller in a cascade (Config.Cascade.IsMaster = TRUE), AllSlaveAutomaticState = TRUE indicates that all slave controllers are in automatic mode.</p> <p>Tuning, manual mode or automatic mode of the master controller can only be executed accurately if all slave controllers are in automatic mode.</p> <p>AllSlaveAutomaticState is only determined if you interconnect the master controller and slave controller with the Master and Slave parameters.</p> <p>For details, see the Master parameter.</p>
NoSlaveSubstituteSetpoint	BOOL	FALSE	<p>If this PID_Temp instance is used as master controller in a cascade (Config.Cascade.IsMaster = TRUE), NoSlaveSubstituteSetpoint = TRUE indicates that no slave controller has activated its substitute setpoint.</p> <p>Tuning, manual mode or automatic mode of the master controller can only be executed accurately if no slave controller has activated its substitute setpoint.</p> <p>NoSlaveSubstituteSetpoint is only determined if you interconnect the master controller and slave controller with the Master and Slave parameters.</p> <p>For details, see the Master parameter.</p>
Heat.EnableTuning	BOOL	TRUE	<p>Enabling of tuning for heating</p> <p>Heat.EnableTuning must be set for the following tunings (at the same time or prior to the start with Mode and ModeActivate):</p> <ul style="list-style-type: none"> <li>• Pretuning heating</li> <li>• Pretuning heating and cooling</li> <li>• Fine tuning heating</li> </ul> <p>This parameter is not automatically reset by the PID_Temp instruction.</p>
Cool.EnableTuning	BOOL	FALSE	<p>Enabling of tuning for cooling</p> <p>Cool.EnableTuning must be set for the following tunings (simultaneously with or prior to the start with Mode and ModeActivate):</p> <ul style="list-style-type: none"> <li>• Pretuning cooling</li> <li>• Pretuning heating and cooling</li> <li>• Fine tuning cooling</li> </ul> <p>Only effective if the cooling output and PID parameter switching are activated ("Config.ActivateCooling" = TRUE and "Config.AdvancedCooling" = TRUE).</p> <p>This parameter is not automatically reset by the PID_Temp instruction.</p>
Config.InputPerOn	BOOL	TRUE	<p>If InputPerOn = TRUE, the Input_PER parameter is used for detecting the process value. If InputPerOn = FALSE, the Input parameter is used.</p>

Tag	Data type	Default	Description
Config.InputUpperLimit	REAL	120.0	<p>High limit of the process value</p> <p>Input and Input_PER are monitored to ensure adherence to this limit. If the limit is exceeded, an error is output and the reaction is determined by ActivateRecoverMode. At the I/O input, the process value can be a maximum of 18% higher than the nominal range (overrange). This means the limit cannot be exceeded when you use an I/O input with the pre-setting for high limit and process value scaling.</p> <p>When pretuning is started, the difference between high and low limit of the process value is checked to determine whether the distance between setpoint and process value meets the necessary requirements.</p> <p>InputUpperLimit &gt; InputLowerLimit</p>
Config.InputLowerLimit	REAL	0.0	<p>Low limit of the process value</p> <p>Input and Input_PER are monitored to ensure adherence to this limit. If the limit is undershot, an error is output and the reaction is determined by ActivateRecoverMode.</p> <p>InputLowerLimit &lt; InputUpperLimit</p>
Config.InputUpperWarning	REAL	3.402822e+38	<p>Warning high limit of the process value</p> <p>Input and Input_PER are monitored to ensure adherence to this limit. If the limit is exceeded, a warning is output at the Warning parameter.</p> <ul style="list-style-type: none"> <li>• If you set InputUpperWarning outside the process value limits, the configured absolute process value high limit is used as the warning high limit.</li> <li>• If you configure InputUpperWarning within the process value limits, this value is used as the warning high limit.</li> </ul> <p>InputUpperWarning &gt; InputLowerWarning</p>
Config.InputLowerWarning	REAL	-3.402822e+38	<p>Warning low limit of the process value</p> <p>Input and Input_PER are monitored to ensure adherence to this limit. If the limit is undershot, a warning is output at the Warning parameter.</p> <ul style="list-style-type: none"> <li>• If you set InputLowerWarning outside the process value limits, the configured absolute process value low limit is used as the warning low limit.</li> <li>• If you configure InputLowerWarning within the process value limits, this value is used as the warning low limit.</li> </ul> <p>InputLowerWarning &lt; InputUpperWarning</p>

Tag	Data type	Default	Description
Config.SetpointUpperLimit	REAL	3.402822e+38	<p>High limit of setpoint</p> <p>Setpoint and SubstituteSetpoint are monitored to ensure adherence to this limit. If the limit is exceeded, a warning is output at the Warning parameter.</p> <ul style="list-style-type: none"> <li>If you configure SetpointUpperLimit outside the process value limits, the configured absolute process value high limit is used as the setpoint high limit.</li> <li>If you configure SetpointUpperLimit within the process value limits, this value is used as the setpoint high limit.</li> </ul> <p>SetpointUpperLimit &gt; SetpointLowerLimit</p>
Config.SetpointLowerLimit	REAL	-3.402822e+38	<p>Low limit of the setpoint</p> <p>Setpoint and SubstituteSetpoint are monitored to ensure adherence to this limit. If the limit is undershot, a warning is output at the Warning parameter.</p> <ul style="list-style-type: none"> <li>If you set SetpointLowerLimit outside the process value limits, the configured process value absolute low limit is used as the setpoint low limit.</li> <li>If you configure SetpointLowerLimit within the process value limits, this value is used as the setpoint low limit.</li> </ul> <p>SetpointLowerLimit &lt; SetpointUpperLimit</p>
Config.ActivateCooling	BOOL	FALSE	<p>Activate cooling output</p> <ul style="list-style-type: none"> <li>Config.ActivateCooling = FALSE Only the outputs for heating are used.</li> <li>Config.ActivateCooling = TRUE The outputs for heating and cooling are used.</li> </ul> <p>If you are using the cooling output, the controller must not be configured as master controller (Config.Cascade.IsMaster must be FALSE) .</p>

Tag	Data type	Default	Description
Config.AdvancedCooling	BOOL	TRUE	<p>Method for heating/cooling</p> <ul style="list-style-type: none"> <li>• Cooling factor (Config.AdvancedCooling = FALSE) The output value calculation for cooling takes place with the PID parameters for heating (Retain.CtrlParams.Heat structure) taking into consideration the configurable cooling factor Config.CoolFactor. This method is suitable if the heating and cooling actuators have a similar time response but different gains. Pretuning and fine tuning for cooling are not available when you select this method. You can only execute the tuning for heating.</li> <li>• PID parameter switching (Config.AdvancedCooling = TRUE) The output value calculation for cooling takes place by means of a separate PID parameter set (Retain.CtrlParams.Cool structure). This method is suitable if the heating and cooling actuator have different time responses and different gains. Pretuning and fine tuning for cooling are only available when you select this method (Mode = 1 or 2, Cool.EnableTuning = TRUE). Config.AdvancedCooling is only calculated if the cooling output is activated (Config.ActivateCooling = TRUE).</li> </ul>
Config.CoolFactor	REAL	1.0	<p>Cooling factor</p> <p>If Config.AdvancedCooling = FALSE, Config.CoolFactor is considered as factor in the calculation of the output value for cooling. This allows different gains of heating and cooling actuators to be taken into account. Config.CoolFactor is not set automatically or adjusted during tuning. You must correctly configure Config.CoolFactor manually with the ratio "heating actuator gain/cooling actuator gain". Example: Config.CoolFactor = 2.0 means that the gain of the heating actuator is twice as high as the gain of the cooling actuator. Config.CoolFactor is only effective if the cooling output is activated (Config.ActivateCooling = TRUE) and cooling factor is selected as method for heating/cooling (Config.AdvancedCooling = FALSE). Config.CoolFactor &gt; 0.0</p>

Tag	Data type	Default	Description
Config.InputScaling.UpperPointIn	REAL	27648.0	Scaling Input_PER high Input_PER is scaled based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn. Only effective if Input_PER is used for process value detection (Config.InputPerOn = TRUE). UpperPointIn > LowerPointIn
Config.InputScaling.LowerPointIn	REAL	0.0	Scaling Input_PER low Input_PER is scaled based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn. Only effective if Input_PER is used for process value detection (Config.InputPerOn = TRUE). LowerPointIn < UpperPointIn
Config.InputScaling.UpperPointOut	REAL	100.0	Scaled high process value Input_PER is scaled based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn. Only effective if Input_PER is used for process value detection (Config.InputPerOn = TRUE). UpperPointOut > LowerPointOut
Config.InputScaling.LowerPointOut	REAL	0.0	Scaled low process value Input_PER is scaled based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn. Only effective if Input_PER is used for process value detection (Config.InputPerOn = TRUE). LowerPointOut < UpperPointOut
Config.Output.Heat.Select	INT	1	Selecting the output value for heating Config.Output.Heat.Select specifies which outputs are used for heating: <ul style="list-style-type: none"> <li>• Heat.Select = 0 - OutputHeat is used</li> <li>• Heat.Select = 1 - OutputHeat and OutputHeat_PWM are used</li> <li>• Heat.Select = 2 - OutputHeat and OutputHeat_PER are used</li> </ul> Outputs that are not used are not calculated and remain at their default value.

Tag	Data type	Default	Description
Config.Output.Heat.PwmPeriode	REAL	0.0	<p>Period duration of the pulse width modulation (PWM) for heating (OutputHeat_PWM output) in seconds:</p> <ul style="list-style-type: none"> <li>Heat.PwmPeriode = 0.0 The sampling time of the PID algorithm for heating (Retain.CtrlParams.Heat.Cycle) is used as period duration of the PWM.</li> <li>Heat.PwmPeriode &gt; 0.0 The value is rounded off to an integer multiple of the PID_Temp sampling time (CycleTime.Value) and used as period duration of the PWM. This setting can be used to improve the smoothing of the process value with a long sampling time of the PID algorithm. The value must meet the following conditions:                             <ul style="list-style-type: none"> <li>Heat.PwmPeriode ≤ Retain.CtrlParams.Heat.Cycle,</li> <li>Heat.PwmPeriode &gt; Config.Output.Heat.MinimumOnTime</li> <li>Heat.PwmPeriode &gt; Config.Output.Heat.MinimumOffTime</li> </ul> </li> </ul>
Config.Output.Heat.PidUpperLimit	REAL	100.0	<p>High limit of the PID output value for heating The PID output value (PidOutputSum) is limited to the high limit. Heat.PidUpperLimit forms a value pair together with the following parameters for scaling of the PID output value (PidOutputSum) to the outputs for heating:</p> <ul style="list-style-type: none"> <li>Heat.UpperScaling for OutputHeat</li> <li>Heat.PwmUpperScaling for OutputHeat_PWM</li> <li>Heat.PerUpperScaling for OutputHeat_PER</li> </ul> <p>If you want to limit the value at the associated output, you must also adjust these scaling values. Heat.PidUpperLimit &gt; Heat.PidLowerLimit</p>



Tag	Data type	Default	Description
Config.Output.Heat.PidLowerLimit	REAL	0.0	<p>Low limit of the PID output value for heating</p> <p>For controllers with <b>deactivated</b> cooling output (Config.ActivateCooling = FALSE), the PID output value (PidOutputSum) is limited to this low limit.</p> <p>For controllers with <b>activated</b> cooling output (Config.ActivateCooling = TRUE), the value must be 0.0.</p> <p>Heat.PidLowerLimit forms a value pair together with the following parameters for scaling of the PID output value (PidOutputSum) to the outputs for heating:</p> <ul style="list-style-type: none"> <li>• Heat.LowerScaling for OutputHeat</li> <li>• Heat.PwmLowerScaling for OutputHeat_PWM</li> <li>• Heat.PerLowerScaling for OutputHeat_PER</li> </ul> <p>If you want to limit the value at the associated output, you must also adjust these scaling values.</p> <p>The permitted value range is determined by the configuration.</p> <ul style="list-style-type: none"> <li>• Cooling output deactivated (Config.ActivateCooling = FALSE): Heat.PidLowerLimit &lt; Heat.PidUpperLimit</li> <li>• Cooling output activated (Config.ActivateCooling = TRUE): Heat.PidLowerLimit = 0.0</li> </ul>
Config.Output.Heat.UpperScaling	REAL	100.0	<p>Scaled high output value for heating</p> <p>Heat.UpperScaling and Heat.PidUpperLimit form a value pair for scaling of the PID output value (PidOutputSum) to the output value for heating (OutputHeat).</p> <p>The OutputHeat value is always located between Heat.UpperScaling and Heat.LowerScaling.</p> <p>Heat.UpperScaling ≠ Heat.LowerScaling</p>
Config.Output.Heat.LowerScaling	REAL	0.0	<p>Scaled low output value for heating</p> <p>Heat.LowerScaling and Heat.PidLowerLimit form a value pair for scaling of the PID output value (PidOutputSum) to the output value for heating (OutputHeat).</p> <p>The OutputHeat value is always located between Heat.UpperScaling and Heat.LowerScaling.</p> <p>Heat.UpperScaling ≠ Heat.LowerScaling</p>
Config.Output.Heat.PwmUpperScaling	REAL	100.0	<p>Scaled high PWM output value for heating</p> <p>Heat.PwmUpperScaling and Heat.PidUpperLimit form a value pair for scaling of the PID output value (PidOutputSum) to the pulse-width modulated output value for heating (OutputHeat_PWM).</p> <p>The OutputHeat_PWM value is always located between Heat.PwmUpperScaling and Heat.PwmLowerScaling.</p> <p>Heat.PwmUpperScaling is only effective if OutputHeat_PWM is selected as output for heating (Heat.Select = 1)</p> <p>100.0 ≥ Heat.PwmUpperScaling ≥ 0.0</p> <p>Heat.PwmUpperScaling ≠ Heat.PwmLowerScaling</p>

Tag	Data type	Default	Description
Con-fig.Output.Heat.PwmLowerScaling	REAL	0.0	<p>Scaled low PWM output value for heating Heat.PwmLowerScaling and Heat.PidLowerLimit form a value pair for scaling of the PID output value (PidOutputSum) to the pulse-width modulated output value for heating (OutputHeat_PWM).</p> <p>The OutputHeat_PWM value is always located between Heat.PwmUpperScaling and Heat.PwmLowerScaling. Heat.PwmLowerScaling is only effective if OutputHeat_PWM is selected as output for heating (Heat.Select = 1)</p> <p><math>100.0 \geq \text{Heat.PwmLowerScaling} \geq 0.0</math> Heat.PwmUpperScaling <math>\neq</math> Heat.PwmLowerScaling</p>
Con-fig.Output.Heat.PerUpperScaling	REAL	27648.0	<p>Scaled high analog output value for heating Heat.PerUpperScaling and Heat.PidUpperLimit form a value pair for scaling of the PID output value (PidOutputSum) to the analog output value for heating (OutputHeat_PER).</p> <p>The OutputHeat_PER value is always located between Heat.PerUpperScaling and Heat.PerLowerScaling. Heat.PerUpperScaling is only effective if OutputHeat_PER is selected as output for heating (Heat.Select = 2)</p> <p><math>32511.0 \geq \text{Heat.PerUpperScaling} \geq -32512.0</math> Heat.PerUpperScaling <math>\neq</math> Heat.PerLowerScaling</p>
Con-fig.Output.Heat.PerLowerScaling	REAL	0.0	<p>Scaled low analog output value for heating Heat.PerLowerScaling and Heat.PidLowerLimit form a value pair for scaling of the PID output value (PidOutputSum) to the analog output value for heating (OutputHeat_PER).</p> <p>The OutputHeat_PER value is always located between Heat.PerUpperScaling and Heat.PerLowerScaling. Heat.PerLowerScaling is only effective if OutputHeat_PER is selected as output for heating (Heat.Select = 2)</p> <p><math>32511.0 \geq \text{Heat.PerLowerScaling} \geq -32512.0</math> Heat.PerUpperScaling <math>\neq</math> Heat.PerLowerScaling</p>
Con-fig.Output.Heat.MinimumOnTime	REAL	0.0	<p>Minimum on time of the pulse width modulation for heating (OutputHeat_PWM output) A PWM pulse is never shorter than this value. The value is rounded off to: Heat.MinimumOnTime = <math>n \times \text{CycleTime.Value}</math> Heat.MinimumOnTime is only effective if the output for heating OutputHeat_PWM is selected (Heat.Select = 1)".</p> <p><math>100000.0 \geq \text{Heat.MinimumOnTime} \geq 0.0</math></p>

Tag	Data type	Default	Description
Config.Output.Heat.MinimumOffTime	REAL	0.0	<p>Minimum off time of the pulse width modulation for heating (OutputHeat_PWM output)            A PWM pause is never shorter than this value.            The value is rounded off to:  <math>\text{Heat.MinimumOffTime} = n \times \text{CycleTime.Value}</math>            Heat.MinimumOffTime is only effective if the output for heating OutputHeat_PWM is selected (Heat.Select = 1)".  <math>100000.0 \geq \text{Heat.MinimumOffTime} \geq 0.0</math></p>
Config.Output.Cool.Select	INT	1	<p>Selecting the output value for cooling            Config.Output.Cool.Select specifies which outputs are used for cooling:</p> <ul style="list-style-type: none"> <li>• Cool.Select = 0 - OutputCool is used</li> <li>• Cool.Select = 1 -OutputCool and OutputCool_PWM are used</li> <li>• Cool.Select = 2 - OutputCool and OutputCool_PER are used</li> </ul> <p>Outputs that are not used are not calculated and remain at their default value.            Only effective if the cooling output is activated (Config.ActivateCooling = TRUE).</p>

Tag	Data type	Default	Description
Config.Output.Cool.PwmPeriode	REAL	0.0	<p>Period duration of the pulse width modulation for cooling (OutputCool_PWM output) in seconds:</p> <ul style="list-style-type: none"> <li>• Cool.PwmPeriode = 0.0 and Config.AdvancedCooling = FALSE: sampling time of the PID algorithm for heating (Retain.CtrlParams.Heat.Cycle) is used as period duration of the PWM.</li> <li>• Cool.PwmPeriode = 0.0 and Config.AdvancedCooling = TRUE: The sampling time of the PID algorithm for cooling (Retain.CtrlParams.Cool.Cycle) is used as period duration of the PWM.</li> <li>• Cool.PwmPeriode &gt; 0.0: The value is rounded off to an integer multiple of the PID_Temp sampling time (CycleTime.Value) and used as period duration of the PWM. This setting can be used to improve the smoothing of the process value with a long sampling time of the PID algorithm. The value must meet the following conditions:                             <ul style="list-style-type: none"> <li>– Cool.PwmPeriode ≤ Retain.CtrlParams.Cool.Cycle or Retain.CtrlParams.Heat.Cycle</li> <li>– Cool.PwmPeriode &gt; Config.Output.Cool.MinimumOnTime</li> <li>– Cool.PwmPeriode &gt; Config.Output.Cool.MinimumOffTime</li> </ul> </li> </ul> <p>Only effective if the cooling output is activated (Config.ActivateCooling = TRUE).</p>
Config.Output.Cool.PidUpperLimit	REAL	0.0	<p>High limit of the PID output value for cooling The value must be 0.0.</p> <p>Cool.PidUpperLimit forms a value pair together with the following parameters for scaling of the PID output value (PidOutputSum) to the outputs for cooling:</p> <ul style="list-style-type: none"> <li>• Cool.LowerScaling for OutputCool</li> <li>• Cool.PwmLowerScaling for OutputCool_PWM</li> <li>• Cool.PerLowerScaling for OutputCool_PER</li> </ul> <p>If you want to limit the value at the associated output, you must also adjust these scaling values.</p> <p>Only effective if the cooling output is activated (Config.ActivateCooling = TRUE).</p> <p>Cool.PidUpperLimit = 0.0</p>

Tag	Data type	Default	Description
Config.Output.Cool.PidLowerLimit	REAL	-100.0	<p>Low limit of the PID output value for cooling</p> <p>For controllers with activated cooling output (Config.ActivateCooling = TRUE), the PID output value (PidOutputSum) is limited to this low limit.</p> <p>Cool.PidLowerLimit forms a value pair together with the following parameters for scaling of the PID output value (PidOutputSum) to the outputs for cooling:</p> <ul style="list-style-type: none"> <li>• Cool.UpperScaling for OutputCool</li> <li>• Cool.PwmUpperScaling for OutputCool_PWM</li> <li>• Cool.PerUpperScaling for OutputCool_PER</li> </ul> <p>If you want to limit the value at the associated output, you must also adjust these scaling values.</p> <p>Only effective if the cooling output is activated (Config.ActivateCooling = TRUE).</p> <p>Cool.PidLowerLimit &lt; Cool.PidUpperLimit</p>
Config.Output.Cool.UpperScaling	REAL	100.0	<p>Scaled high output value for cooling</p> <p>Cool.UpperScaling and Cool.PidLowerLimit form a value pair for scaling of the PID output value (PidOutputSum) to the output value for cooling (OutputCool).</p> <p>The OutputCool value is always located between Cool.UpperScaling and Cool.LowerScaling.</p> <p>Only effective if the cooling output is activated (Config.ActivateCooling = TRUE).</p> <p>Cool.UpperScaling ≠ Cool.LowerScaling</p>
Config.Output.Cool.LowerScaling	REAL	0.0	<p>Scaled low output value for cooling</p> <p>Cool.LowerScaling and Cool.PidUpperLimit form a value pair for scaling of the PID output value (PidOutputSum) to the output value for cooling (OutputCool).</p> <p>The OutputCool value is always located between Cool.UpperScaling and Cool.LowerScaling.</p> <p>Only effective if the cooling output is activated (Config.ActivateCooling = TRUE).</p> <p>Cool.UpperScaling ≠ Cool.LowerScaling</p>
Config.Output.Cool.PwmUpperScaling	REAL	100.0	<p>Scaled high PWM output value for cooling</p> <p>Cool.PwmUpperScaling and Cool.PidLowerLimit form a value pair for scaling of the PID output value (PidOutputSum) to the pulse-width modulated output value for cooling (OutputCool_PWM).</p> <p>The OutputCool_PWM value is always located between Cool.PwmUpperScaling and Cool.PwmLowerScaling.</p> <p>Cool.PwmUpperScaling is only effective if the cooling output is activated (Config.ActivateCooling = TRUE) and OutputCool_PWM is selected as output for cooling (Cool.Select = 1).</p> <p><math>100.0 \geq \text{Cool.PwmUpperScaling} \geq 0.0</math></p> <p>Cool.PwmUpperScaling ≠ Cool.PwmLowerScaling</p>

8.3 PID\_Temp

Tag	Data type	Default	Description
Con-fig.Output.Cool.PwmLowerScaling	REAL	0.0	<p>Scaled low PWM output value for cooling</p> <p>Cool.PwmLowerScaling and Cool.PidUpperLimit form a value pair for scaling of the PID output value (PidOutputSum) to the pulse-width modulated output value for cooling (OutputCool_PWM).</p> <p>The OutputCool_PWM value is always located between Cool.PwmUpperScaling and CoolPwm.LowerScaling.</p> <p>Cool.PwmLowerScaling is only effective if the cooling output is activated (Config.ActivateCooling = TRUE) and OutputCool_PWM is selected as output for cooling (Cool.Select = 1).</p> <p><math>100.0 \geq \text{Cool.PwmLowerScaling} \geq 0.0</math></p> <p><math>\text{Cool.PwmUpperScaling} \neq \text{Cool.PwmLowerScaling}</math></p>
Con-fig.Output.Cool.PerUpperScaling	REAL	27648.0	<p>Scaled high analog output value for cooling</p> <p>Cool.PerUpperScaling and Cool.PidLowerLimit form a value pair for scaling of the PID output value (PidOutputSum) to the analog output value for cooling (OutputCool_PER).</p> <p>The OutputCool_PER value is always located between Cool.PerUpperScaling and Cool.PerLowerScaling.</p> <p>Cool.PerUpperScaling is only effective if the cooling output is activated (Config.ActivateCooling = TRUE) and OutputCool_PER is selected as output for cooling (Cool.Select = 2).</p> <p><math>32511.0 \geq \text{Cool.PerUpperScaling} \geq -32512.0</math></p> <p><math>\text{Cool.PerUpperScaling} \neq \text{Cool.PerLowerScaling}</math></p>
Con-fig.Output.Cool.PerLowerScaling	REAL	0.0	<p>Scaled low analog output value for cooling</p> <p>Cool.PerLowerScaling and Cool.PidUpperLimit form a value pair for scaling of the PID output value (PidOutputSum) to the analog output value for cooling (OutputCool_PER).</p> <p>The OutputCool_PER value is always located between Cool.PerUpperScaling and Cool.PerLowerScaling.</p> <p>Cool.PerLowerScaling is only effective if the cooling output is activated (Config.ActivateCooling = TRUE) and OutputCool_PER is selected as output for cooling (Cool.Select = 2).</p> <p><math>32511.0 \geq \text{Cool.PerLowerScaling} \geq -32512.0</math></p> <p><math>\text{Cool.PerUpperScaling} \neq \text{Cool.PerLowerScaling}</math></p>
Con-fig.Output.Cool.MinimumOnTime	REAL	0.0	<p>Minimum on time of the pulse width modulation for cooling (OutputCool_PWM output)</p> <p>A PWM pulse is never shorter than this value.</p> <p>The value is rounded off to:</p> <p><math>\text{Cool.MinimumOnTime} = n \times \text{CycleTime.Value}</math></p> <p>Cool.MinimumOnTime is only effective if the output for cooling OutputCool_PWM is selected (Cool.Select = 1).</p> <p>Only effective if the cooling output is activated (Config.ActivateCooling = TRUE).</p> <p><math>100000.0 \geq \text{Cool.MinimumOnTime} \geq 0.0</math></p>

Tag	Data type	Default	Description
Config.Output.Cool.MinimumOffTime	REAL	0.0	<p>Minimum off time of the pulse width modulation for cooling (OutputCool_PWM output) A PWM pause is never shorter than this value. The value is rounded off to: Cool.MinimumOffTime = n × CycleTime.Value Cool.MinimumOffTime is only effective if the output for cooling OutputCool_PWM is selected (Cool.Select = 1). Only effective if the cooling output is activated (Config.ActivateCooling = TRUE). 100000.0 ≥ Cool.MinimumOffTime ≥ 0.0</p>
<p>If you are using PID_Temp in a cascade, the master controller and slave controller exchange information via the Master and Slave parameters. You need to make the interconnection. For details, see the Master parameter.</p>			
Config.Cascade.IsMaster	BOOL	FALSE	<p>The controller is master in a cascade and provides the slave setpoint. Set IsMaster = TRUE if you are using this PID_Temp instance as master controller in a cascade. A master controller defines the setpoint of a slave controller with its output. A PID_Temp instance can be master controller and slave controller at the same time. If the controller is used as master controller, the cooling output must be deactivated (Config.ActivateCooling = FALSE).</p>
Config.Cascade.IsSlave	BOOL	FALSE	<p>The controller is slave in a cascade and receives its setpoint from the master. Set IsSlave = TRUE if you are using this PID_Temp instance as slave controller in a cascade. A slave controller receives its setpoint (Setpoint parameter) from the output of its master controller (OutputHeat parameter). A PID_Temp instance can be master controller and slave controller at the same time.</p>
Config.Cascade.AntiWindUpMode	INT	1	<p>Anti-windup behavior in the cascade Options are:</p> <ul style="list-style-type: none"> <li>• Anti-windup = 0 The AntiWindUp functionality is deactivated. The master controller does not respond to the limit of its slave controllers.</li> <li>• Anti-windup = 1 The integral action of the master controller is reduced in the ratio "Slaves in limit" to "Number of slaves" ("CountSlaves" parameter). This reduces the effects of the limitation on the control behavior.</li> <li>• Anti-windup = 2 The integral action of the master controller is held as soon as a slave controller is in the limit.</li> </ul> <p>Only effective if the controller is configured as master controller (Config.Cascade.IsMaster = TRUE).</p>

8.3 PID\_Temp

Tag	Data type	Default	Description
Config.Cascade.CountSlaves	INT	1	Number of subordinate slaves Here you enter the number of directly subordinate slave controllers which receive their setpoint from this master controller. Only effective if the controller is configured as master controller (Config.Cascade.IsMaster = TRUE). 255 ≥ CountSlaves ≥ 1
CycleTime.StartEstimation	BOOL	TRUE	If CycleTime.EnEstimation = TRUE, CycleTime.StartEstimation = TRUE starts automatic determination of the PID_Temp sampling time (cycle time of the calling OB). CycleTime.StartEstimation = FALSE is set once measurement is complete.
CycleTime.EnEstimation	BOOL	TRUE	If CycleTime.EnEstimation = TRUE, the PID_Temp sampling time is determined automatically. If CycleTime.EnEstimation = FALSE, the sampling time PID_Temp is not determined automatically and must be configured correctly manually with CycleTime.Value.
CycleTime.EnMonitoring	BOOL	TRUE	If CycleTime.EnMonitoring = FALSE, the PID_Temp sampling time is not monitored. If PID_Temp cannot be executed within the sampling time, no error (Error-Bits=0000800h) is output and PID_Temp does not respond as configured with ActivateRecoverMode.
CycleTime.Value	REAL	0.1	PID_Temp sampling time (cycle time of the calling OB) in seconds CycleTime.Value is determined automatically and is usually equivalent to the cycle time of the calling OB.
You can reload values from the CtrlParamsBackUp structure with LoadBackUp = TRUE.			
CtrlParamsBackUp.SetByUser	BOOL	FALSE	Saved value of Retain.CtrlParams.SetByUser
CtrlParamsBackUp.Heat.Gain	REAL	1.0	Saved proportional gain for heating
CtrlParamsBackUp.Heat.Ti	REAL	20.0	Saved integral action time for heating in seconds
CtrlParamsBackUp.Heat.Td	REAL	0.0	Saved derivative action time for heating in seconds
CtrlParamsBackUp.Heat.TdFiltRatio	REAL	0.2	Saved derivative delay coefficient for heating
CtrlParamsBackUp.Heat.PWeighting	REAL	1.0	Saved weighting of the proportional action for heating
CtrlParamsBackUp.Heat.DWeighting	REAL	1.0	Saved weighting of the derivative action for heating
CtrlParamsBackUp.Heat.Cycle	REAL	1.0	Saved sampling time of the PID algorithm for heating in seconds
CtrlParamsBackUp.Heat.ControlZone	REAL	3.402822e+38	Saved control zone width for heating
CtrlParamsBackUp.Heat.DeadZone	REAL	0.0	Saved dead zone width for heating
CtrlParamsBackUp.Cool.Gain	REAL	1.0	Saved proportional gain for cooling
CtrlParamsBackUp.Cool.Ti	REAL	20.0	Saved integral action time for cooling in seconds
CtrlParamsBackUp.Cool.Td	REAL	0.0	Saved derivative action time for cooling in seconds
CtrlParamsBackUp.Cool.TdFiltRatio	REAL	0.2	Saved derivative delay coefficient for cooling



Tag	Data type	Default	Description
CtrlParamsBack-Up.Cool.PWeighting	REAL	1.0	Saved proportional action weighting factor for cooling
CtrlParamsBack-Up.Cool.DWeighting	REAL	1.0	Saved derivative action weighting factor for cooling
CtrlParamsBackUp.Cool.Cycle	REAL	1.0	Saved sampling time of the PID algorithm for cooling in seconds
CtrlParamsBack-Up.Cool.ControlZone	REAL	3.402822e+38	Saved control zone width for cooling
CtrlParamsBack-Up.Cool.DeadZone	REAL	0.0	Saved dead zone width for cooling
PIDSelf-Tune.SUT.CalculateParamsHeat	BOOL	FALSE	<p>The properties of the heating branch of the controlled system are saved during pretuning for heating. If SUT.CalculateParamsHeat = TRUE, the PID parameters for heating (Retain.CtrlParams.Heat structure) are recalculated on the basis of these properties. This enables you to change the parameter calculation method (PID-SelfTune.SUT.TuneRuleHeat parameter) without having to repeat the tuning.</p> <p>SUT.CalculateParamsHeat is set to FALSE after the calculation.</p> <p>Only possible if the pretuning was successful (SUT.ProcParHeatOk = TRUE).</p>
PIDSelf-Tune.SUT.CalculateParamsCool	BOOL	FALSE	<p>The properties of the cooling branch of the controlled system are saved during tuning for cooling. If SUT.CalculateParamsCool = TRUE, the PID parameters for cooling (Retain.CtrlParams.Cool structure) are recalculated on the basis of these properties. This enables you to change the parameter calculation method (PID-SelfTune.SUT.TuneRuleCool parameter) without having to repeat the tuning.</p> <p>SUT.CalculateParamsCool is set to FALSE after the calculation.</p> <p>Only possible if the pretuning was successful (SUT.ProcParCoolOk = TRUE).</p> <p>Only effective if Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE.</p>
PIDSelfTune.SUT.TuneRuleHeat	INT	2	<p>Method for PID parameter calculation with pretuning for heating</p> <p>Options are:</p> <ul style="list-style-type: none"> <li>• SUT.TuneRuleHeat = 0: PID according to CHR</li> <li>• SUT.TuneRuleHeat = 1: PI according to CHR</li> <li>• SUT.TuneRuleHeat = 2: PID for temperature processes according to CHR (results in a slower and rather asymptomatic control response with lower overshoot than SUT.TuneRuleHeat = 0)</li> </ul> <p>(CHR = Chien, Hrones and Reswick)</p> <p>Only with SUT.TuneRuleHeat = 2 is the control zone Retain.CtrlParams.Heat.ControlZone automatically set during pretuning for heating.</p>

Tag	Data type	Default	Description
PIDSelfTune.SUT.TuneRuleCool	INT	2	<p>Method for PID parameter calculation with pretuning for cooling</p> <p>Options are:</p> <ul style="list-style-type: none"> <li>• SUT.TuneRuleCool = 0: PID according to CHR</li> <li>• SUT.TuneRuleCool = 1: PI according to CHR</li> <li>• SUT.TuneRuleCool = 2: PID for temperature processes according to CHR (results in a slower and rather asymptomatic control response with lower overshoot than SUT.TuneRuleCool = 0)</li> </ul> <p>(CHR = Chien, Hrones and Reswick)</p> <p>Only with SUT.TuneRuleCool = 2 is the control zone Retain.CtrlParams.Cool.ControlZone automatically set during pretuning for cooling.</p> <p>SUT.TuneRuleCool is only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE, Config.AdvancedCooling = TRUE).</p>
PIDSelfTune.SUT.State	INT	0	<p>The SUT.State tag indicates the current phase of pretuning:</p> <ul style="list-style-type: none"> <li>• State = 0: Initialize pretuning</li> <li>• State = 100: Calculate standard deviation for heating</li> <li>• State = 200: Calculate standard deviation for cooling</li> <li>• State = 300: Determine point of inflection for heating</li> <li>• State = 400: Determine point of inflection for cooling</li> <li>• State = 500: Set heating to setpoint after reaching point of inflection</li> <li>• State = 600: Set cooling to setpoint after reaching point of inflection</li> <li>• State = 700: Compare efficiency of the heating actuator and cooling actuator</li> <li>• State = 800: Heating and cooling activated</li> <li>• State = 900: Cooling activated</li> <li>• State = 1000: Determine delay time after switching off heating</li> <li>• State = 9900: Pretuning successful</li> <li>• State = 1: Pretuning not successful</li> </ul>
PIDSelfTune.SUT.ProcParHeatOk	BOOL	FALSE	<p>TRUE: The calculation of the process parameters for pretuning heating was successful.</p> <p>This tag is set during tuning.</p> <p>It must be TRUE for calculation of the PID parameters for heating.</p>

Tag	Data type	Default	Description
PIDSelfTune.SUT.ProcParCoolOk	BOOL	FALSE	TRUE: The calculation of the process parameters for pretuning cooling was successful. This tag is set during tuning. It must be TRUE for calculation of the PID parameters for cooling.
PIDSelf-Tune.SUT.AdaptDelayTime	INT	0	The AdaptDelayTime tag determines the adaptation of the delay time for heating at the operating point (for "Pretuning heating" and "Pretuning heating and cooling"). Options are: <ul style="list-style-type: none"> <li>• SUT.AdaptDelayTime = 0: No adaptation of delay time. The SUT.State = 1000 phase is skipped. This option results in a shorter tuning time than with SUT.AdaptDelayTime = 1.</li> <li>• SUT.AdaptDelayTime = 1: Adaptation of the delay time to the setpoint in SUT.State = 1000 phase by switching off heating temporarily. This option results in a longer tuning time than with SUT.AdaptDelayTime = 0. It can improve the control response if the process behavior depends significantly on the operating point (non-linearity). This option should not be used for multi-zone applications with strong thermal connections.</li> </ul>

Tag	Data type	Default	Description
PIDSelfTune.SUT.CoolingMode	INT	0	<p>The CoolingMode tag determines the manipulated variable output to determine the cooling parameters (for pre-tuning heating and cooling).</p> <p>Options are:</p> <ul style="list-style-type: none"> <li>• SUT.CoolingMode = 0: Switch off heating and switch on cooling after reaching the setpoint. The SUT.State = 700 phase is skipped. Phase SUT.State = 500 is followed by phase SUT.State = 900. This option can improve the control response if the gain of the cooling actuator is low compared to the gain of the heating actuator. It results in a shorter tuning time than with SUT.CoolingMode = 1 or 2.</li> <li>• SUT.CoolingMode = 1: Switch on cooling in addition to heating after reaching the setpoint. The SUT.State = 700 phase is skipped. Phase SUT.State = 500 is followed by phase SUT.State = 800. This option can improve the control response if the gain of the cooling actuator is high compared to the gain of the heating actuator.</li> <li>• SUT.CoolingMode = 2: After heating up to the setpoint, a decision is automatically made in phase SUT.State = 700 as to whether heating is switched off. Phase SUT.State = 500 is followed by phase SUT.State = 700 and then SUT.State = 800 or SUT.State = 900. This option requires more time than options 0 and 1.</li> </ul>
PIDSelfTune.TIR.RunIn	BOOL	FALSE	<p>Use the RunIn tag to specify the sequence of fine tuning during start from automatic mode.</p> <ul style="list-style-type: none"> <li>• RunIn = FALSE If fine tuning is started from automatic mode, the system uses the existing PID parameters to control to the setpoint (TIR.State = 500 or 600). Only then will fine tuning start.</li> <li>• RunIn = TRUE PID_Temp tries to reach the setpoint with minimum or maximum output value (TIR.State = 300 or 400). This can produce increased overshoot. Fine tuning then starts automatically. RunIn is set to FALSE after fine tuning. During start of fine tuning from Inactive or Manual mode, PID_Temp reacts as described under RunIn = TRUE.</li> </ul>

Tag	Data type	Default	Description
PIDSelf-Tune.TIR.CalculateParamsHeat	BOOL	FALSE	<p>The properties of the heating branch of the controlled system are saved during fine tuning for heating. If TIR.CalculateParamsHeat= TRUE, the PID parameters for heating (Retain.CtrlParams.Heat structure) are recalculated on the basis of these properties. This enables you to change the parameter calculation method (PIDSelfTune.TIR.TuneRuleHeat parameter) without having to repeat the tuning.</p> <p>TIR.CalculateParamsHeat is set to FALSE after the calculation.</p> <p>Only possible if fine tuning heating was successful beforehand (TIR.ProcParHeatOk = TRUE).</p>
PIDSelf-Tune.TIR.CalculateParamsCool	BOOL	FALSE	<p>The properties of the cooling branch of the controlled system are saved during fine tuning for cooling. If TIR.CalculateParamsCool= TRUE, the PID parameters for cooling (Retain.CtrlParams.Cool structure) are recalculated on the basis of these properties. This enables you to change the parameter calculation method (PIDSelfTune.TIR.TuneRuleCool parameter) without having to repeat the tuning.</p> <p>TIR.CalculateParamsCool is set to FALSE after the calculation.</p> <p>Only possible if fine tuning cooling was successful beforehand (TIR.ProcParCoolOk = TRUE).</p> <p>Only effective if Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE</p>
PIDSelfTune.TIR.TuneRuleHeat	INT	0	<p>Method for parameter calculation during fine tuning for heating</p> <p>Options are:</p> <ul style="list-style-type: none"> <li>• TIR.TuneRuleHeat = 0: PID automatic</li> <li>• TIR.TuneRuleHeat = 1: PID fast (faster control response with higher amplitudes of the output value than with TIR.TuneRuleHeat = 2)</li> <li>• TIR.TuneRuleHeat = 2: PID slow (slower control response with lower amplitudes of the output value than with TIR.TuneRuleHeat = 1)</li> <li>• TIR.TuneRuleHeat = 3: ZN PID</li> <li>• TIR.TuneRuleHeat = 4: ZN PI</li> <li>• TIR.TuneRuleHeat = 5: ZN P</li> </ul> <p>(ZN=Ziegler-Nichols)</p> <p>To be able to repeat the calculation of the PID parameters for heating with TIR.CalculateParamsHeat and TIR.TuneRuleHeat = 0, 1 or 2, the previous fine tuning also has to have been executed with TIR.TuneRuleHeat = 0, 1 or 2. If this is not the case, TIR.TuneRuleHeat = 3 is used.</p> <p>The recalculation of the PID parameters for heating with TIR.CalculateParamsHeat and TIR.TuneRuleHeat = 3, 4 or 5 is always possible.</p>

Tag	Data type	Default	Description
PIDSelfTune.TIR.TuneRuleCool	INT	0	<p>Method for parameter calculation during fine tuning for cooling</p> <p>Options are:</p> <ul style="list-style-type: none"> <li>• TIR.TuneRuleCool = 0: PID automatic</li> <li>• TIR.TuneRuleCool = 1: PID fast (faster control response with higher amplitudes of the output value than with TIR.TuneRuleCool = 2)</li> <li>• TIR.TuneRuleCool = 2: PID slow (slower control response with lower amplitudes of the output value than with TIR.TuneRuleCool = 1)</li> <li>• TIR.TuneRuleCool = 3: ZN PID</li> <li>• TIR.TuneRuleCool = 4: ZN PI</li> <li>• TIR.TuneRuleCool = 5: ZN P</li> </ul> <p>(ZN=Ziegler-Nichols)</p> <p>To be able to repeat the calculation of the PID parameters for cooling with TIR.CalculateParamsCool and TIR.TuneRuleCool = 0, 1 or 2, the previous fine tuning also has to have been executed with TIR.TuneRuleCool = 0, 1 or 2. If this is not the case, TIR.TuneRuleCool = 3 is used.</p> <p>The recalculation of the PID parameters for cooling with TIR.CalculateParamsCool and TIR.TuneRuleCool = 3, 4 or 5 is always possible.</p> <p>Only effective if the cooling output and PID parameter switching are activated (ConfigActivateCooling = TRUE and Config.AdvancedCooling = TRUE).</p>

Tag	Data type	Default	Description
PIDSelfTune.TIR.State	INT	0	<p>The TIR.State tag indicates the current phase of "fine tuning":</p> <ul style="list-style-type: none"> <li>• State = 0: Initialize fine tuning</li> <li>• State = 100: Calculate standard deviation for heating</li> <li>• State = 200: Calculate standard deviation for cooling</li> <li>• State = 300: Attempting to reach setpoint for heating with two-step control using heating</li> <li>• State = 400: Attempting to reach setpoint for cooling with two-step control using cooling</li> <li>• State = 500: Attempting to reach setpoint for heating with PID control</li> <li>• State = 600: Attempting to reach setpoint for cooling with PID control</li> <li>• State = 700: Calculate standard deviation for heating</li> <li>• State = 800: Calculate standard deviation for cooling</li> <li>• State = 900: Determine oscillation and calculate parameters for heating</li> <li>• State = 1000: Determine oscillation and calculate parameters for cooling</li> <li>• State = 9900: Fine tuning successful</li> <li>• State = 1: Fine tuning not successful</li> </ul>
PIDSelfTune.TIR.ProcParHeatOk	BOOL	FALSE	<p>TRUE: The calculation of the process parameters for fine tuning heating was successful.</p> <p>This tag is set during tuning.</p> <p>It must be met for calculation of the PID parameters for heating.</p>
PIDSelfTune.TIR.ProcParCoolOk	BOOL	FALSE	<p>TRUE: The calculation of the process parameters for fine tuning cooling was successful.</p> <p>This tag is set during tuning.</p> <p>It must be met for calculation of the PID parameters for cooling.</p>

Tag	Data type	Default	Description
PIDSelf-Tune.TIR.OutputOffsetHeat	REAL	0.0	<p>Tuning offset heating of the PID output value                      TIR.OutputOffsetHeat is added to the value that results from PidOutputSum for the heating branch.</p> <p>To receive a positive offset at the outputs for heating, define a positive value for TIR.OutputOffsetHeat.</p> <p>The resulting values at the outputs for heating are the result of the configured output scaling (Struktur Config.Output.Heat).</p> <p>This tuning offset can be used in controllers with activated cooling output and PID parameter switching (Config.ActivateCooling = TRUE, Config.AdvancedCooling = TRUE) for fine tuning cooling. If the outputs for cooling are not active at the setpoint that is to be tuned (PidOutputSum &gt; 0.0), fine tuning cooling is not possible. In this case, define a positive tuning offset heating which is greater than the PID output value (PidOutputSum) at the setpoint in the steady state before you start tuning. This step increases the values at the outputs for heating and activates the outputs for cooling (PidOutputSum &lt; 0.0). Fine tuning cooling is now possible.</p> <p>When fine tuning is complete, TIR.OutputOffsetHeat is reset to 0.0.</p> <p>Major changes at TIR.OutputOffsetHeat in one step can result in temporary overshoots.</p> <p>Config.Output.Heat.PidUpperLimit ≥ PIDSelf-Tune.TIR.OutputOffsetHeat ≥ Config.Output.Heat.PidLowerLimit</p>



Tag	Data type	Default	Description
PIDSelf-Tune.TIR.OutputOffsetCool	REAL	0.0	<p>Tuning offset cooling of the PID output value</p> <p>TIR.OutputOffsetCool is added to the value that results from PidOutputSum for the cooling branch.</p> <p>To receive a positive offset at the outputs for cooling, define a negative value for TIR.OutputOffsetCool.</p> <p>The resulting values at the outputs for cooling are the result of the configured output scaling (Struktur Config.Output.Cool).</p> <p>This tuning offset can be used in controllers with activated cooling output (Config.ActivateCooling = TRUE) for fine tuning heating. If the outputs for heating are not active at the setpoint that is to be tuned (PidOutputSum &lt; 0.0), fine tuning heating is not possible. In this case, define a negative tuning offset cooling which is less than the PID output value (PidOutputSum) at the setpoint in the steady state before you start tuning. This step increases the values at the outputs for cooling and activates the outputs for heating (PidOutputSum &gt; 0.0). Fine tuning heating is now possible.</p> <p>When fine tuning is complete, TIR.OutputOffsetCool is reset to 0.0.</p> <p>Major changes at TIR.OutputOffsetCool in one step can result in temporary overshoots.</p> <p>Config.Output.Cool.PidUpperLimit ≥ PIDSelf-Tune.TIR.OutputOffsetCool ≥ Config.Output.Cool.PidLowerLimit</p>
PIDSelf-Tune.TIR.WaitForControlln	BOOL	FALSE	<p>Waiting with fine tuning after reaching the setpoint</p> <p>If TIR.WaitForControlln = TRUE, fine tuning waits in between reaching the setpoint (TIR.State = 500 or 600) and calculation of the standard deviation (TIR.State = 700 or 800) until a FALSE -&gt; TRUE edge is given at TIR.FinishControlln.</p> <p>TIR.WaitForControlln can be used for simultaneous fine tuning of several controllers in multi-zone applications to synchronize tuning of the individual zones. It ensures that all zones have reached their setpoints before the actual tuning starts. The influence of thermal connections between the zones on tuning can be reduced in this way.</p> <p>TIR.WaitForControlln is only effective if fine tuning is started from automatic mode with PIDSelf-Tune.TIR.RunIn = FALSE.</p>
PIDSelfTune.TIR.ControllnReady	BOOL	FALSE	<p>If TIR.WaitForControlln = TRUE, PID_Temp sets TIR.ControllnReady = TRUE as soon as the setpoint has been reached and waits with additional tuning steps until a FALSE -&gt; TRUE edge is given at TIR.FinishControlln.</p>
PIDSelfTune.TIR.FinishControlln	BOOL	FALSE	<p>If TIR.ControllnReady = TRUE, a FALSE -&gt; TRUE edge at TIR.FinishControlln stops the wait and fine tuning resumes.</p>
PIDCtrl.IOutputOld	REAL	0.0	Integral action in last cycle

Tag	Data type	Default	Description
PIDCtrl.PIDInit	BOOL	FALSE	PIDCtrl.PIDInit is available as of PID_Temp version 1.1. If PIDCtrl.PIDInit = TRUE in "Automatic mode", the integral action PIDCtrl.IOutputOld is pre-assigned automatically as if PidOutputSum = OverwriteInitialOutputValue in the previous cycle. This can be used for a Override control with PID_Temp (Page 209).
Retain.CtrlParams.SetByUser	BOOL	FALSE	If the PID parameters are entered manually in the configuration editor, SetByUser = TRUE. This parameter is used for display in the editors and does not influence the control algorithm. SetByUser is retentive.
Retain.CtrlParams.Heat.Gain	REAL	1.0	Active proportional gain for heating Heat.Gain is retentive. Heat.Gain ≥ 0.0
Retain..CtrlParams.Heat.Ti	REAL	20.0	Active integral action time for heating in seconds The integral action for heating is switched off with Heat.CtrlParams.Ti = 0.0. Heat.Ti is retentive. 100000.0 ≥ Heat.Ti ≥ 0.0
Retain.CtrlParams.Heat.Td	REAL	0.0	Active derivative action time for heating in seconds The derivative action for heating is switched off with Heat.CtrlParams.Td = 0.0. Heat.Td is retentive. 100000.0 ≥ Heat.Td ≥ 0.0
Re- tain.CtrlParams.Heat.TdFiltRatio	REAL	0.2	Active derivative delay coefficient for heating The derivative delay coefficient delays the effect of the derivative action. Derivative delay = derivative action time × derivative delay coefficient <ul style="list-style-type: none"> <li>• 0.0: Derivative action is effective for one cycle only and therefore almost not effective.</li> <li>• 0.5: This value has proved useful in practice for controlled systems with <b>one</b> dominant time constant.</li> <li>• &gt; 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed.</li> </ul> Heat.TdFiltRatio is retentive. Heat.TdFiltRatio ≥ 0.0

Tag	Data type	Default	Description
Re-retain.CtrlParams.Heat.PWeighting	REAL	1.0	<p>Active weighting of the proportional action for heating</p> <p>The proportional action may weaken with changes to the setpoint.</p> <p>Values from 0.0 to 1.0 are applicable.</p> <ul style="list-style-type: none"> <li>• 1.0: Proportional action for setpoint change is fully effective</li> <li>• 0.0: Proportional action for setpoint change is not effective</li> </ul> <p>The proportional action is always fully effective when the process value is changed.</p> <p>Heat.PWeighting is retentive.</p> <p><math>1.0 \geq \text{Heat.PWeighting} \geq 0.0</math></p>
Re-retain.CtrlParams.Heat.DWeighting	REAL	1.0	<p>Active weighting of the derivative action for heating</p> <p>The derivative action may weaken with changes to the setpoint.</p> <p>Values from 0.0 to 1.0 are applicable.</p> <ul style="list-style-type: none"> <li>• 1.0: Derivative action is fully effective upon setpoint change</li> <li>• 0.0: Derivative action is not effective upon setpoint change</li> </ul> <p>The derivative action is always fully effective when the process value is changed.</p> <p>Heat.DWeighting is retentive.</p> <p><math>1.0 \geq \text{Heat.DWeighting} \geq 0.0</math></p>
Retain.CtrlParams.Heat.Cycle	REAL	1.0	<p>Active sampling time of the PID algorithm for heating in seconds</p> <p>CtrlParams.Heat.Cycle is calculated during tuning and rounded to an integer multiple of CycleTime.Value.</p> <p>If Config.Output.Heat.PwmPeriode = 0.0, Heat.Cycle is used as period duration of the pulse width modulation for heating.</p> <p>If Config.Output.Cool.PwmPeriode = 0.0 and Config.AdvancedCooling = FALSE, Heat.Cycle is used as period duration of the pulse width modulation for cooling.</p> <p>Heat.Cycle is retentive.</p> <p><math>100000.0 \geq \text{Heat.Cycle} &gt; 0.0</math></p>

Tag	Data type	Default	Description
Re-retain.CtrlParams.Heat.ControlZone	REAL	3.402822e+38	<p>Active control zone width for heating</p> <p>The control zone for heating is switched off with Heat.ControlZone = 3.402822e+38.</p> <p>Heat.ControlZone is only set automatically during pre-tuning heating or pretuning heating and cooling if PID-SelfTune.SUT.TuneRuleHeat = 2 is selected as method of the parameter calculation.</p> <p>For controllers with deactivated cooling output (Config.ActivateCooling = FALSE) or controllers with activated cooling output and cooling factor (Config.AdvancedCooling = FALSE), the control zone is symmetrically located between Setpoint – Heat.ControlZone and Setpoint + Heat.ControlZone.</p> <p>For controllers with activated cooling output and PID parameter switching (Config.ActivateCooling = TRUE, Config.AdvancedCooling = TRUE), the control zone is located between Setpoint – Heat.ControlZone and Setpoint + Cool.ControlZone.</p> <p>Heat.ControlZone is retentive. Heat.ControlZone &gt; 0.0</p>
Re-retain.CtrlParams.Heat.DeadZone	REAL	0.0	<p>Active dead zone width for heating (see PID parameters (Page 181))</p> <p>The dead zone for heating is switched off with Heat.DeadZone = 0.0.</p> <p>Heat.DeadZone is not set automatically or adjusted during tuning. You must correctly configure Heat.DeadZone manually.</p> <p>When the dead zone is switched on, the result can be a permanent control deviation (deviation between setpoint and process value). This can have a negative effect on fine tuning.</p> <p>For controllers with deactivated cooling output (Config.ActivateCooling = FALSE) or controllers with activated cooling output and cooling factor (Config.AdvancedCooling = FALSE), the dead zone is symmetrically located between Setpoint – Heat.DeadZone and Setpoint + Heat.DeadZone.</p> <p>For controllers with activated cooling output and PID parameter switching (Config.ActivateCooling = TRUE, Config.AdvancedCooling = TRUE), the dead zone is located between Setpoint – Heat.DeadZone and Setpoint + Cool.DeadZone.</p> <p>Heat.DeadZone is retentive. Heat.DeadZone ≥ 0.0</p>
Retain.CtrlParams.Cool.Gain	REAL	1.0	<p>Active proportional gain for cooling</p> <p>Cool.Gain is retentive.</p> <p>Only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE).</p> <p>Cool.Gain ≥ 0.0</p>

Tag	Data type	Default	Description
Retain.CtrlParams.Cool.Ti	REAL	20.0	<p>Active integral action time for cooling in seconds</p> <p>The integral action for cooling is switched off with Cool.CtrlParams.Ti = 0.0.</p> <p>Cool.Ti is retentive.</p> <p>Only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE).</p> <p><math>100000.0 \geq \text{Cool.Ti} \geq 0.0</math></p>
Retain.CtrlParams.Cool.Td	REAL	0.0	<p>Active derivative action time for cooling in seconds</p> <p>The derivative action for cooling is switched off with Cool.CtrlParams.Td = 0.0.</p> <p>Cool.Td is retentive.</p> <p>Only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE).</p> <p><math>100000.0 \geq \text{Cool.Td} \geq 0.0</math></p>
Re-retain.CtrlParams.Cool.TdFiltRatio	REAL	0.2	<p>Active derivative delay coefficient for cooling</p> <p>The derivative delay coefficient delays the effect of the derivative action.</p> <p>Derivative delay = derivative action time × derivative delay coefficient</p> <ul style="list-style-type: none"> <li>• 0.0: Derivative action is effective for one cycle only and therefore almost not effective.</li> <li>• 0.5: This value has proved useful in practice for controlled systems with one dominant time constant.</li> <li>• &gt; 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed.</li> </ul> <p>Cool.TdFiltRatio is retentive.</p> <p>Only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE).</p> <p><math>\text{Cool.TdFiltRatio} \geq 0.0</math></p>
Re-retain.CtrlParams.Cool.PWeighting	REAL	1.0	<p>Active weighting of the proportional action for cooling</p> <p>The proportional action may weaken with changes to the setpoint.</p> <p>Values from 0.0 to 1.0 are applicable.</p> <ul style="list-style-type: none"> <li>• 1.0: Proportional action for setpoint change is fully effective</li> <li>• 0.0: Proportional action for setpoint change is not effective</li> </ul> <p>The proportional action is always fully effective when the process value is changed.</p> <p>Cool.PWeighting is retentive.</p> <p>Only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE).</p> <p><math>1.0 \geq \text{Cool.PWeighting} \geq 0.0</math></p>

Tag	Data type	Default	Description
Retain.CtrlParams.Cool.DWeighting	REAL	1.0	<p>Active weighting of the derivative action for cooling The derivative action may weaken with changes to the setpoint. Values from 0.0 to 1.0 are applicable.</p> <ul style="list-style-type: none"> <li>• 1.0: Derivative action is fully effective upon setpoint change</li> <li>• 0.0: Derivative action is not effective upon setpoint change</li> </ul> <p>The derivative action is always fully effective when the process value is changed. Cool.DWeighting is retentive. Only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE). <math>1.0 \geq \text{Cool.DWeighting} \geq 0.0</math></p>
Retain.CtrlParams.Cool.Cycle	REAL	1.0	<p>Active sampling time of the PID algorithm for cooling in seconds CtrlParams.Cool.Cycle is calculated during tuning and rounded off to an integer multiple of CycleTime.. If Config.Output.Cool.PwmPeriode = 0.0 and Config.AdvancedCooling = TRUE, Cool.Cycle is used as period duration of the pulse width modulation for cooling. If Config.Output.Cool.PwmPeriode = 0.0 and Config.AdvancedCooling = FALSE, Heat.Cycle is used as period duration of the pulse width modulation for cooling. Cool.Cycle is retentive. Only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE). <math>100000.0 \geq \text{Cool.Cycle} &gt; 0.0</math></p>

Tag	Data type	Default	Description
Re-tain.CtrlParams.Cool.ControlZone	REAL	3.402822e+38	<p>Active control zone width for cooling</p> <p>The control zone for cooling is switched off with Cool.ControlZone = 3.402822e+38.</p> <p>Cool.ControlZone is only set automatically during pretuning cooling or pretuning heating and cooling if PIDSelf-Tune.SUT.TuneRuleCool = 2 is selected as method of the parameter calculation.</p> <p>Cool.ControlZone is retentive.</p> <p>Only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE).</p> <p>Cool.ControlZone &gt; 0.0</p>
Re-tain.CtrlParams.Cool.DeadZone	REAL	0.0	<p>Active dead zone width for cooling (see PID parameters (Page 181))</p> <p>The dead zone for cooling is switched off with Cool.DeadZone = 0.0.</p> <p>Cool.DeadZone is not set automatically or adjusted during tuning. You must correctly configure Cool.DeadZone manually.</p> <p>When the dead zone is switched on, the result can be a permanent control deviation (deviation between setpoint and process value). This can have a negative effect on fine tuning.</p> <p>Cool.DeadZone is retentive.</p> <p>Only effective if the cooling output and PID parameter switching are activated (Config.ActivateCooling = TRUE and Config.AdvancedCooling = TRUE).</p> <p>Cool.DeadZone ≥ 0.0</p>

---

**Note**

Change the tags listed in this table in "Inactive" mode to prevent malfunction of the PID controller.

---

**See also**

PID\_Temp ActivateRecoverMode tag (Page 453)

PID\_Temp Warning tag (Page 455)

Multi-zone controlling with PID\_Temp (Page 206)

### 8.3.3.7 PID\_Temp state and mode parameters

#### Correlation of the parameters

The State parameter shows the current operating mode of the PID controller. You cannot change the State parameter.

With a rising edge at ModeActivate, PID\_Temp switches to the operating mode saved in the Mode in-out parameter.

Heat.EnableTuning and Cool.EnableTuning specify for pretuning and fine tuning, if tuning takes place for heating or cooling.

If the CPU is switched on or switches from Stop to RUN mode, PID\_Temp starts in the operating mode that is saved in the Mode parameter. To leave PID\_Temp in "Inactive" mode, set RunModeByStartup = FALSE.

#### Meaning of values

State / Mode	Description of operating mode
0	<p><b>Inactive</b></p> <p>The following output values are output in "Inactive" mode:</p> <ul style="list-style-type: none"><li>• 0.0 as PID output value (PidOutputSum)</li><li>• 0.0 as output value for heating (OutputHeat) and output value for cooling (OutputCool)</li><li>• 0 as analog output value for heating (OutputHeat_PER) and analog output value for cooling (OutputCool_PER)</li><li>• FALSE as PWM output value for heating (OutputHeat_PWM) and PWM output value for cooling (OutputCool_PWM)</li></ul> <p>This does not depend on the configured output value limits and scaling in the structures Config.Output.Heat and Config.Output.Cool.</p>



State / Mode	Description of operating mode
1	<p><b>Pretuning</b></p> <p>The pretuning determines the process response to a jump change of the output value and searches for the point of inflection. The PID parameters are calculated from the maximum rate of rise and dead time of the controlled system. You obtain the best PID parameters when you perform pretuning and fine tuning.</p> <p>PID_Temp offers different pretuning types depending on the configuration:</p> <ul style="list-style-type: none"> <li>• Pretuning heating: <p>A jump change is output at the output value heating, the PID parameters for heating are calculated (Retain.CtrlParams.Heat structure), and control to the setpoint then takes place in automatic mode.</p> <p>If the process behavior strongly depends on the operating point, an adaptation of the delay time can be activated at the setpoint with PIDSelfTune.SUT.AdaptDelayTime.</p> </li> <li>• Pretuning heating and cooling: <p>A jump is output at the output value heating. As soon as the process value is close to the setpoint, a jump change is output at the output value cooling. The PID parameters for heating (Retain.CtrlParams.Heat structure) and cooling (Retain.CtrlParams.Cool structure) are calculated. Then, control to the setpoint takes place in automatic mode.</p> <p>If the process behavior strongly depends on the operating point, an adaptation of the delay time can be activated at the setpoint with PIDSelfTune.SUT.AdaptDelayTime.</p> <p>Depending on the effect of the cooling actuator compared to the heating actuator, the quality of tuning can be influenced by whether or not the heating and cooling outputs are operated simultaneously during tuning. You can specify this with PIDSelfTune.SUT.CoolingMode.</p> </li> <li>• Pretuning cooling: <p>A jump change is output at the output value cooling and the PID parameters for cooling are calculated (Struktur Retain.CtrlParams.Cool). Then, control to the setpoint takes place in automatic mode.</p> </li> </ul> <p>If you want to tune the PID parameters for heating and cooling, you can expect a better control response with "Pretuning heating" followed by "Pretuning cooling" rather than with "Pretuning heating and cooling". However, carrying out pretuning in two steps takes more time.</p> <p><b>General requirements for pretuning:</b></p> <ul style="list-style-type: none"> <li>• The PID_Temp instruction is called in a cyclic interrupt OB.</li> <li>• Inactive (State = 0), manual mode (State = 4), or automatic mode (State = 3)</li> <li>• ManualEnable = FALSE</li> <li>• Reset = FALSE</li> <li>• The setpoint and the process value lie within the configured limits.</li> </ul>

State / Mode	Description of operating mode
1	<p><b>Pretuning</b></p> <p>The pretuning determines the process response to a jump change of the output value and searches for the point of inflection. The PID parameters are calculated from the maximum rate of rise and dead time of the controlled system. You obtain the best PID parameters when you perform pretuning and fine tuning.</p> <p>PID_Temp offers different pretuning types depending on the configuration:</p> <ul style="list-style-type: none"> <li>• Pretuning heating:                     <p>A jump change is output at the output value heating, the PID parameters for heating are calculated (Retain.CtrlParams.Heat structure), and control to the setpoint then takes place in automatic mode.</p> <p>If the process behavior strongly depends on the operating point, an adaptation of the delay time can be activated at the setpoint with PIDSelfTune.SUT.AdaptDelayTime.</p> </li> <li>• Pretuning heating and cooling:                     <p>A jump is output at the output value heating. As soon as the process value is close to the setpoint, a jump change is output at the output value cooling. The PID parameters for heating (Retain.CtrlParams.Heat structure) and cooling (Retain.CtrlParams.Cool structure) are calculated. Then, control to the setpoint takes place in automatic mode.</p> <p>If the process behavior strongly depends on the operating point, an adaptation of the delay time can be activated at the setpoint with PIDSelfTune.SUT.AdaptDelayTime.</p> <p>Depending on the effect of the cooling actuator compared to the heating actuator, the quality of tuning can be influenced by whether or not the heating and cooling outputs are operated simultaneously during tuning. You can specify this with PIDSelfTune.SUT.CoolingMode.</p> </li> <li>• Pretuning cooling:                     <p>A jump change is output at the output value cooling and the PID parameters for cooling are calculated (Struktur Retain.CtrlParams.Cool). Then, control to the setpoint takes place in automatic mode.</p> </li> </ul> <p>If you want to tune the PID parameters for heating and cooling, you can expect a better control response with "Pretuning heating" followed by "Pretuning cooling" rather than with "Pretuning heating and cooling". However, carrying out pretuning in two steps takes more time.</p> <p><b>General requirements for pretuning:</b></p> <ul style="list-style-type: none"> <li>• The PID_Temp instruction is called in a cyclic interrupt OB.</li> <li>• Inactive (State = 0), manual mode (State = 4), or automatic mode (State = 3)</li> <li>• ManualEnable = FALSE</li> <li>• Reset = FALSE</li> <li>• The setpoint and the process value lie within the configured limits.</li> </ul>

State / Mode	Description of operating mode
1	<p><b>Requirements for pretuning heating:</b></p> <ul style="list-style-type: none"> <li>• Heat.EnableTuning = TRUE</li> <li>• Cool.EnableTuning = FALSE</li> <li>• The process value must not be too close to the setpoint.  <math> \text{Setpoint} - \text{Input}  &gt; 0.3 *  \text{Config.InputUpperLimit} - \text{Config.InputLowerLimit} </math> and  <math> \text{Setpoint} - \text{Input}  &gt; 0.5 *  \text{Setpoint} </math></li> <li>• The setpoint is greater than the process value.  Setpoint &gt; Input</li> </ul> <p><b>Requirements for pretuning heating and cooling:</b></p> <ul style="list-style-type: none"> <li>• Heat.EnableTuning = TRUE</li> <li>• Cool.EnableTuning = TRUE</li> <li>• The cooling output is activated (Config.ActivateCooling = TRUE).</li> <li>• The PID parameter switching is activated (Config.AdvancedCooling = TRUE).</li> <li>• The process value must not be too close to the setpoint.  <math> \text{Setpoint} - \text{Input}  &gt; 0.3 *  \text{Config.InputUpperLimit} - \text{Config.InputLowerLimit} </math> and  <math> \text{Setpoint} - \text{Input}  &gt; 0.5 *  \text{Setpoint} </math></li> <li>• The setpoint is greater than the process value.  Setpoint &gt; Input</li> </ul>

State / Mode	Description of operating mode
1	<p><b>Requirements for pretuning cooling:</b></p> <ul style="list-style-type: none"> <li>• Heat.EnableTuning = FALSE</li> <li>• Cool.EnableTuning = TRUE</li> <li>• The cooling output is activated (Config.ActivateCooling = TRUE).</li> <li>• The PID parameter switching is activated (Config.AdvancedCooling = TRUE).</li> <li>• A "pretuning heating" or "pretuning heating and cooling" has been successful (PIDSelfTune.SUT.ProcParHeatOk = TRUE), if possible at the same setpoint.</li> <li>• The process value must be close to the setpoint.  <math display="block"> \text{Setpoint} - \text{Input}  &lt; 0.05 *  \text{Config.InputUpperLimit} - \text{Config.InputLowerLimit} </math> </li> </ul> <p>The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. This is most likely the case in operating modes "Inactive" or "Manual mode".</p> <p>The setpoint is frozen in the CurrentSetpoint tag. Tuning is canceled when:</p> <ul style="list-style-type: none"> <li>• Setpoint &gt; CurrentSetpoint + CancelTuningLevel</li> <li style="padding-left: 20px;">or</li> <li>• Setpoint &lt; CurrentSetpoint - CancelTuningLevel</li> </ul> <p>The method for calculation of the PID parameters can be specified separately for heating and cooling with PIDSelfTune.SUT.TuneRuleHeat and PIDSelfTune.SUT.TuneRuleCool.</p> <p>Before the PID parameters are recalculated, they are backed up in the CtrlParamsBackUp structure and can be reactivated with LoadBackUp.</p> <p>After successful pretuning, the switch is made to automatic mode.</p> <p>After unsuccessful pretuning, the switch to the mode is determined by ActivateRecoverMode.</p> <p>The phase of pretuning is indicated with PIDSelfTune.SUT.State.</p>

State / Mode	Description of operating mode
2	<p><b>Fine tuning</b></p> <p>Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are tuned for the operating point from the amplitude and frequency of this oscillation. PID parameters from fine tuning usually have better master control and disturbance characteristics than PID parameters from pretuning. You obtain the best PID parameters when you perform pretuning and fine tuning.</p> <p>PID_Temp automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value.</p> <p>PID_Temp offers different fine tuning types depending on the configuration:</p> <ul style="list-style-type: none"> <li>• Fine tuning heating: <p style="margin-left: 20px;">PID_Temp generates an oscillation of the process value with periodic changes at the output value heating and calculates the PID parameters for heating (Struktur Retain.CtrlParams.Heat).</p> </li> <li>• Fine tuning cooling: <p style="margin-left: 20px;">PID_Temp generates an oscillation of the process value with periodic changes at the output value cooling and calculates the PID parameters for cooling (Struktur Retain.CtrlParams.Cool).</p> </li> </ul> <p><b>Temporary tuning offset for heating/cooling controllers</b></p> <p>If PID_Temp is used as heating/cooling controller (Config.ActivateCooling = TRUE), the PID output value (PidOutputSum) at the setpoint must meet the following requirements for a process value oscillation to be generated and fine tuning to be successful:</p> <ul style="list-style-type: none"> <li>• Positive PID output value for fine tuning heating</li> <li>• Negative PID output value for fine tuning cooling</li> </ul> <p>If this requirement is not met, you can define a temporary offset for fine tuning which is output at the output with the opposite effect:</p> <ul style="list-style-type: none"> <li>• Offset for cooling output (PIDSelfTune.TIR.OutputOffsetCool) with fine tuning heating. <p style="margin-left: 20px;">Define a negative tuning offset cooling which is less than the PID output value (PidOutputSum) at the setpoint in the steady state before you start tuning.</p> </li> <li>• Offset for heating output (PIDSelfTune.TIR.OutputOffsetHeat) with fine tuning cooling. <p style="margin-left: 20px;">Define a positive tuning offset heating which is greater than the PID output value (PidOutputSum) at the setpoint in the steady state before you start tuning.</p> </li> </ul> <p>The defined offset is balanced by the PID algorithm so that the process value remains at the setpoint. The height of the offset allows the PID output value to be adapted correspondingly so that it fulfills the requirement mentioned above.</p> <p>To avoid larger overshoots of the process value when defining the offset, it can also be increased in several steps.</p> <p>If PID_Temp exits the fine tuning mode, the tuning offset is reset.</p>

State / Mode	Description of operating mode
2	<p>Example for definition of an offset for fine tuning cooling:</p> <ul style="list-style-type: none"> <li>• Without offset:                             <ul style="list-style-type: none"> <li>– Setpoint = Process value (ScaledInput) = 80°C</li> <li>– PID output value (PidOutputSum) = 30.0</li> <li>– Output value heating (OutputHeat) = 30.0</li> <li>– Output value cooling (OutputCool) = 0.0</li> </ul> <p>Oscillation of the process value around the setpoint cannot be generated with the cooling output alone.</p> <p>Fine tuning would fail here.</p> </li> <li>• With definition of an offset for heating output (PIDSelfTune.TIR.OutputOffsetHeat) = 80.0                             <ul style="list-style-type: none"> <li>– Setpoint = process value (ScaledInput) = 80°C</li> <li>– PID output value (PidOutputSum) = -50.0</li> <li>– Output value heating (OutputHeat) = 80.0</li> <li>– Output value cooling (OutputCool) = -50.0</li> </ul> <p>By defining an offset for the heating output, the cooling output can now create an oscillation of the process value around the setpoint.</p> <p>Fine tuning can now be carried out successfully.</p> </li> </ul> <p><b>General requirements for fine tuning:</b></p> <ul style="list-style-type: none"> <li>• The PID_Temp instruction is called in a cyclic interrupt OB.</li> <li>• No disturbances are expected.</li> <li>• The setpoint and the process value lie within the configured limits.</li> <li>• The control loop has stabilized at the operating point. The operating point is reached when the process value corresponds to the setpoint.</li> </ul> <p>When the dead zone is switched on, the result can be a permanent control deviation (deviation between setpoint and actual value). This can have a negative effect on fine tuning.</p> <ul style="list-style-type: none"> <li>• ManualEnable = FALSE</li> <li>• Reset = FALSE</li> <li>• Automatic (State = 3), inactive (State = 0) or manual (State = 4) mode</li> </ul> <p><b>Requirements for fine tuning heating:</b></p> <ul style="list-style-type: none"> <li>• Heat.EnableTuning = TRUE</li> <li>• Cool.EnableTuning = FALSE</li> <li>• If PID_Temp is configured as heating/cooling controller (Config.ActivateCooling = TRUE), the heating output must be active at the operating point at which tuning is to take place (PidOutputSum &gt; 0.0 (see tuning offset)).</li> </ul>

State / Mode	Description of operating mode
2	<p><b>Requirements for fine tuning cooling:</b></p> <ul style="list-style-type: none"> <li>• Heat.EnableTuning = FALSE</li> <li>• Cool.EnableTuning = TRUE</li> <li>• The cooling output is activated (Config.ActivateCooling = TRUE).</li> <li>• The PID parameter switching is activated (Config.AdvancedCooling = TRUE)</li> <li>• The cooling output must be active at the operating point at which tuning is to take place (PidOutputSum &lt; 0.0 (see tuning offset)).</li> </ul> <p><b>The course of fine tuning is determined by the mode from which it is started:</b></p> <ul style="list-style-type: none"> <li>• Automatic mode (State = 3) with PIDSelfTune.TIR.RunIn = FALSE (default) Start fine tuning from automatic mode if you wish to improve the existing PID parameters through tuning. PID_Temp controls the system using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.</li> <li>• Inactive (State = 0), manual mode (State = 4), or automatic mode (State = 3) with PIDSelfTune.TIR.RunIn = TRUE Attempts are made to reach the setpoint with the minimum or maximum output value: <ul style="list-style-type: none"> <li>– with minimum or maximum output value heating for fine tuning heating</li> <li>– With minimum or maximum output value cooling for fine tuning cooling.</li> </ul> This can produce increased overshoot. Fine tuning starts when the setpoint is reached. If the setpoint cannot be reached, PID_Temp does not automatically abort tuning.</li> </ul> <p>The setpoint is frozen in the CurrentSetpoint tag. Tuning is canceled when:</p> <ul style="list-style-type: none"> <li>• Setpoint &gt; CurrentSetpoint + CancelTuningLevel</li> </ul> <p style="text-align: center;">or</p> <ul style="list-style-type: none"> <li>• Setpoint &lt; CurrentSetpoint - CancelTuningLevel</li> </ul> <p>The method for calculation of the PID parameters can be specified separately for heating and cooling with PIDSelfTune.TIR.TuneRuleHeat and PIDSelfTune.TIR.TuneRuleCool. Before the PID parameters are recalculated, they are backed up in the CtrlParamsBackUp structure and can be reactivated with LoadBackUp. The controller changes to automatic mode after successful fine tuning. After unsuccessful fine tuning, the switch to the mode is determined by ActivateRecoverMode. The "Fine tuning" phase is indicated with PIDSelfTune.TIR.State.</p>
3	<p><b>Automatic mode</b></p> <p>In automatic mode, PID_Temp corrects the controlled system in accordance with the parameters specified. The controller switches to automatic mode if one the following requirements is met:</p> <ul style="list-style-type: none"> <li>• Pretuning successfully completed</li> <li>• Fine tuning successfully completed</li> <li>• Changing of the Mode in-out parameter to the value 3 and a rising edge at ModeActivate.</li> </ul> <p>The switchover from automatic mode to manual mode is only bumpless if carried out in the commissioning editor. The ActivateRecoverMode tag is taken into consideration in automatic mode.</p>

State / Mode	Description of operating mode
4	<p><b>Manual mode</b></p> <p>In manual mode, you specify a manual PID output value in the ManualValue parameter. The values at the outputs for heating and cooling resulting from this manual value are the result of the configured output scaling.</p> <p>You can also activate this operating mode using ManualEnable = TRUE. We recommend that you change the operating mode using Mode and ModeActivate only.</p> <p>The switchover from manual mode to automatic mode is bumpless.</p> <p>The ActivateRecoverMode tag is taken into consideration in manual mode.</p>
5	<p><b>Substitute output value with error monitoring</b></p> <p>The control algorithm is deactivated. The SetSubstituteOutput tag determines which PID output value (Pi-dOutputSum) is output in this operating mode.</p> <ul style="list-style-type: none"> <li>• SetSubstituteOutput = FALSE: Last valid PID output value</li> <li>• SetSubstituteOutput = TRUE: Substitute output value (SubstituteOutput)</li> </ul> <p>You cannot activate this operating mode using Mode = 5.</p> <p>In the event of an error, it is activated instead of "Inactive" operating mode if all the following conditions are met:</p> <ul style="list-style-type: none"> <li>• Automatic mode (State = 3)</li> <li>• ActivateRecoverMode = TRUE</li> <li>• One or more errors have occurred in which ActivateRecoverMode is effective.</li> </ul> <p>As soon as the errors are no longer pending, PID_Temp switches back to automatic mode.</p>

**ENO characteristics**

If State = 0, then ENO = FALSE.

If State ≠ 0, then ENO = TRUE.



### Automatic switchover of operating mode during commissioning

Automatic mode is activated following successful pretuning or fine tuning. The following table shows how Mode and State change during successful pretuning.

Cycle no.	Mode	State	Action
0	4	4	Set Mode = 1
1	1	4	Set ModeActivate = TRUE
1	4	1	Value of State is saved in Mode parameter Pretuning is started
n	4	1	Pretuning successfully completed
n	3	3	Automatic mode is started

PID\_Temp automatically switches the operating mode in the event of an error.

The following table shows how Mode and State change during pretuning with errors.

Cycle no.	Mode	State	Action
0	4	4	Set Mode = 1
1	1	4	Set ModeActivate = TRUE
1	4	1	Value of State is saved in Mode parameter Pretuning is started
n	4	1	Pretuning canceled
n	4	4	Manual mode is started

If ActivateRecoverMode = TRUE, the operating mode that is saved in the Mode parameter is activated. When you start pretuning or fine tuning, PID\_Temp has saved the value of State in the Mode in-out parameter. This means PID\_Temp switches to the mode from which tuning was started.

If ActivateRecoverMode = FALSE, the system switches to "Inactive" operating mode.

### See also

Output parameters of PID\_Temp (Page 403)

In/out parameters of PID\_Temp V2 (Page 405)

**8.3.3.8 PID\_Temp ErrorBits parameter**

If several errors are pending simultaneously, the values of the ErrorBits are displayed with binary addition. The display of ErrorBits = 0000003h, for example, indicates that the errors 0000001h and 0000002h are pending simultaneously.

ErrorBits (DW#16#...)	Description
0000000	There is no error.
0000001	<p>The "Input" parameter is outside the process value limits.</p> <ul style="list-style-type: none"> <li>• Input &gt; Config.InputUpperLimit or</li> <li>• Input &lt; Config.InputLowerLimit</li> </ul> <p>If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in automatic mode.</p> <p>If manual mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in manual mode.</p> <p>If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp switches to the operating mode that is saved in the Mode parameter.</p>
0000002	<p>Invalid value at "Input_PER" parameter. Check whether an error is pending at the analog input.</p> <p>If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp outputs the configured substitute output value. As soon as the error is no longer pending, PID_Temp switches back to automatic mode.</p> <p>If manual mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in manual mode.</p> <p>If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp switches to the operating mode that is saved in the Mode parameter.</p>
0000004	<p>Error during fine tuning. Oscillation of the process value could not be maintained.</p> <p>If PID_Temp is used as heating-cooling controller (Config.ActivateCooling = TRUE), to be able to generate actual value oscillation, the PID output value (PidOutputSum) at the setpoint must be</p> <ul style="list-style-type: none"> <li>• positive for fine tuning heating</li> <li>• negative for fine tuning cooling</li> </ul> <p>If this requirement is not met, use the tuning offsets ( PIDSelfTune.TIR.OutputOffsetCool and PIDSelfTune.TIR.OutputOffsetHeat tags), see Fine tuning (Page 192).</p> <p>If ActivateRecoverMode was = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.</p>
0000008	<p>Error at start of pretuning. The process value is too close to the setpoint or greater than the setpoint. Start fine tuning.</p> <p>If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.</p>
0000010	<p>The setpoint was changed during tuning.</p> <p>You can set the permitted fluctuation of the setpoint at the CancelTuningLevel tag.</p> <p>If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.</p>
0000020	<p>Pretuning is not permitted during fine tuning.</p> <p>If ActivateRecoverMode = TRUE before the error occurred, PID_Temp remains in fine tuning mode.</p>
0000040	<p>Error during pretuning. Cooling could not reduce the process value.</p> <p>If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.</p>

ErrorBits (DW#16#...)	Description
0000100	<p>Error during fine tuning resulted in invalid parameters.</p> <p>If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.</p>
0000200	<p>Invalid value at "Input" parameter: Value has an invalid number format.</p> <p>If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp outputs the configured substitute output value. As soon as the error is no longer pending, PID_Temp switches back to automatic mode.</p> <p>If manual mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in manual mode.</p> <p>If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp switches to the operating mode that is saved in the Mode parameter.</p>
0000400	<p>Calculation of output value failed. Check the PID parameters.</p> <p>If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp outputs the configured substitute output value. As soon as the error is no longer pending, PID_Temp switches back to automatic mode.</p> <p>If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp switches to the operating mode that is saved in the Mode parameter.</p>
0000800	<p>Sampling time error: PID_Temp is not called within the sampling time of the cyclic interrupt OB.</p> <p>If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in automatic mode.</p> <p>If manual mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in manual mode.</p> <p>If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp switches to the operating mode that is saved in the Mode parameter.</p> <p>If this error occurred during simulation with PLCSIM, see the notes under Simulating PID_Temp with PLCSIM (Page 213).</p>
0001000	<p>Invalid value at "Setpoint" parameter or "SubstituteSetpoint": Value has an invalid number format.</p> <p>If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp outputs the configured substitute output value. As soon as the error is no longer pending, PID_Temp switches back to automatic mode.</p> <p>If manual mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in manual mode.</p> <p>If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp switches to the operating mode that is saved in the Mode parameter.</p>
0010000	<p>Invalid value at ManualValue parameter. Value has an invalid number format.</p> <p>If ActivateRecoverMode = TRUE before the error occurred, PID_Temp remains in manual mode and uses SubstituteOutput as PID output value. As soon as you specify a valid value in ManualValue, PID_Temp uses it as the PID output value.</p>
0020000	<p>Invalid value at SubstituteOutput tag. Value has an invalid number format.</p> <p>PID_Temp remains in the "Substitute output value with error monitoring" mode or manual mode and uses the low limit of the PID output value for heating (Config.Output.Heat.PidLowerLimit) as PID output value.</p> <p>As soon as you specify a valid value in SubstituteOutput, PID_Temp uses it as the PID output value.</p>

8.3 PID\_Temp

ErrorBits (DW#16#...)	Description
0040000	<p>Invalid value at Disturbance parameter. Value has an invalid number format.</p> <p>If automatic mode was active and ActivateRecoverMode = TRUE before the error occurred, Disturbance is set to zero. PID_Temp remains in automatic mode.</p> <p>If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp switches to the operating mode that is saved in the Mode parameter. If Disturbance in the current phase has no effect on the output value, tuning is not be canceled.</p>
0200000	<p>Error in master in the cascade: Slaves are not in automatic mode or have activated substitute setpoint and prevent tuning of the master.</p> <p>If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.</p>
0400000	<p>Pretuning heating is not permitted while cooling is active.</p> <p>If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.</p>
0800000	<p>The process value must be close to the setpoint to start pretuning cooling.</p> <p>If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.</p>
1000000	<p>Error at start of tuning: Heat.EnableTuning and Cool.EnableTuning are not set or do not match the configuration.</p> <p>If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.</p>
2000000	<p>Pretuning cooling requires successful pretuning heating.</p> <p>If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.</p>
4000000	<p>Error at start of fine tuning: Heat.EnableTuning and Cool.EnableTuning must not be set simultaneously.</p> <p>If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.</p>
8000000	<p>Error during calculation of the PID parameters resulted in invalid parameters.</p> <p>The invalid parameters are discarded and the original PID parameters are retained unchanged.</p> <p>We can distinguish between the following cases:</p> <ul style="list-style-type: none"> <li>• If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in automatic mode.</li> <li>• If manual mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in manual mode.</li> <li>• If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp switches to the operating mode that is saved in the Mode parameter.</li> </ul>

### 8.3.3.9 PID\_Temp ActivateRecoverMode tag

The ActivateRecoverMode tag determines the reaction to error. The Error parameter indicates if an error is pending. When the error is no longer pending, Error = FALSE. The ErrorBits parameter shows which errors have occurred.

#### Automatic mode and manual mode

**NOTICE**

**Your system may be damaged.**

If ActivateRecoverMode = TRUE, PID\_Temp remains in automatic mode or in manual mode even if there is an error and the process limit values are exceeded.

This may damage your system.

It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

ActivateRecoverMode	Description
FALSE	PID_Temp switches to "Inactive" mode in the event of an error. The controller is only activated by a falling edge at Reset or a rising edge at ModeActivate.
TRUE	<p><b>Automatic mode</b></p> <p>If errors occur frequently in automatic mode, this setting has a negative effect on the control response, because PID_Temp switches between the calculated PID output value and the substitute output value at each error. In this case, check the ErrorBits parameter and eliminate the cause of the error.</p> <p>If one or several of the following errors occur and automatic mode was active before the error occurred, PID_Temp remains in automatic mode:</p> <ul style="list-style-type: none"> <li>• 0000001h: The "Input" parameter is outside the process value limits.</li> <li>• 0000800h: Sampling time error</li> <li>• 0040000h: Invalid value at Disturbance parameter.</li> <li>• 8000000h: Error during calculation of the PID parameters</li> </ul> <p>If one or several of the following errors occur and automatic mode was active before the error occurred, PID_Temp switches to "Substitute output value with error monitoring" mode:</p> <ul style="list-style-type: none"> <li>• 0000002h: Invalid value at Input_PER parameter.</li> <li>• 0000200h: Invalid value at Input parameter.</li> <li>• 0000400h: Calculation of output value failed.</li> <li>• 0001000h: Invalid value at Setpoint parameter or SubstituteSetpoint.</li> </ul> <p>As soon as the errors are no longer pending, PID_Temp switches back to automatic mode.</p> <p>If the following error occurs in "Substitute output value with error monitoring" mode, PID_Temp sets the PID output value to Config.Output.Heat.PidLowerLimit as long as this error is pending:</p> <ul style="list-style-type: none"> <li>• 0020000h: Invalid value at SubstituteOutput tag. Value has an invalid number format.</li> </ul> <p>This behavior is independent of SetSubstituteOutput.</p>

8.3 PID\_Temp

ActivateRecoverMode	Description
TRUE	<p><b>Manual mode</b></p> <p>If one or several errors occur and manual mode was active before the error occurred, PID_Temp remains in manual mode.</p> <p>If the following error occurs in manual mode, as long as this error is pending, PID_Temp sets the PID output value to SubstituteOutput:</p> <ul style="list-style-type: none"> <li>• 0010000h: Invalid value at ManualValue parameter. Value has an invalid number format.</li> </ul> <p>If the error 0010000h is pending in manual mode and the following error occurs, PID_Temp sets the PID output value to Config.Output.Heat.PidLowerLimit as long as this error is pending:</p> <ul style="list-style-type: none"> <li>• 0020000h: Invalid value at SubstituteOutput tag. Value has an invalid number format.</li> </ul> <p>This behavior is independent of SetSubstituteOutput.</p>

Pretuning and fine tuning

ActivateRecoverMode	Description
FALSE	<p>PID_Temp switches to "Inactive" mode in the event of an error. The controller is only activated by a falling edge at Reset or a rising edge at ModeActivate.</p>
TRUE	<p>If the following error occurs, PID_Temp remains in the active mode:</p> <ul style="list-style-type: none"> <li>• 0000020h: Pretuning is not permitted during fine tuning.</li> </ul> <p>The following errors are ignored:</p> <ul style="list-style-type: none"> <li>• 0010000h: Invalid value at ManualValue parameter.</li> <li>• 0020000h: Invalid value at SubstituteOutput tag.</li> </ul> <p>When any other error occurs, PID_Temp cancels the tuning and switches to the mode from which tuning was started.</p>

### 8.3.3.10 PID\_Temp Warning tag

If several warnings are pending simultaneously, the values of the Warning tag are displayed with binary addition. If the warning 0000003h is displayed, for example, the warnings 0000001h and 0000002h are pending simultaneously.

Warning (DW#16#....)	Description
0000000	No warning pending.
0000001	The point of inflection was not found during pretuning.
0000004	The setpoint was limited to the configured limits.
0000008	Not all the necessary controlled system properties were defined for the selected method of calculation. Instead, the PID parameters were calculated using the method TIR.TuneRuleHeat = 3 or TIR.TuneRuleCool = 3.
0000010	The operating mode could not be changed because Reset = TRUE or ManualEnable = TRUE.
0000020	The cycle time of the calling OB limits the sampling time of the PID algorithm. Improve results by using shorter OB cycle times.
0000040	The process value exceeded one of its warning limits.
0000080	Invalid value at Mode. The operating mode is not switched.
0000100	The manual value was limited to the limits of the PID output value.
0000200	The specified rule for tuning is not supported. No PID parameters are calculated.
0001000	The substitute output value cannot be reached because it is outside the output value limits.
0004000	The specified selection of the output value for heating and/or cooling is not supported. Only the output OutputHeat or OutputCool is used.
0008000	Invalid value at PIDSelfTune.SUT.AdaptDelayTime. The default value 0 is used.
0010000	Invalid value at PIDSelfTune.SUT.CoolingMode. The default value 0 is used.
0020000	The activation of cooling (Config.ActivateCooling tag) is not supported by the controller that is used as master (Config.Cascade.IsMaster tag). PID_Temp works as heating controller. Set the Config.ActivateCooling tag to FALSE.
0040000	Invalid value at Retain.CtrlParams.Heat.Gain, Retain.CtrlParams.Cool.Gain oder Config.CoolFactor. PID_Temp supports only positive values for proportional gain (heating and cooling) and cooling factor. Automatic mode remains active with PID output value 0.0. The integral component is stopped.

The following warnings are deleted as soon as the cause has been remedied or you repeat the action with valid parameters:

- 0000001h
- 0000004h
- 0000008h
- 0000040h
- 0000100h

All other warnings are cleared with a rising edge at Reset or ErrorAck.

### 8.3.3.11 PwmPeriode tag

If the PID algorithm sampling time (Retain.CtrlParams.Heat.Cycle or Retain.CtrlParams.Heat.Cycle) and thus the time period of the pulse width modulation is very high when you use OutputHeat\_PWM or OutputCool\_PWM, you can define a deviating shorter time period at the Config.Output.Heat.PwmPeriode or Config.Output.Cool.PwmPeriode parameters to improve the smoothness of the process value.

#### Time period of the pulse width modulation at OutputHeat\_PWM

Time period of the PWM at output OutputHeat\_PWM depending on Config.Output.Heat.PwmPeriode:

- Heat.PwmPeriode = 0.0 (default)

The sampling time of the PID algorithm for heating (Retain.CtrlParams.Heat.Cycle) is used as time period of the PWM.

- Heat.PwmPeriode > 0.0

The value is rounded off to an integer multiple of the PID\_Temp sampling time (CycleTime.Value) and used as time period of the PWM.

The value must meet the following conditions:

- Heat.PwmPeriode ≤ Retain.CtrlParams.Heat.Cycle
- Heat.PwmPeriode > Config.Output.Heat.MinimumOnTime
- Heat.PwmPeriode > Config.Output.Heat.MinimumOffTime

#### Time period of the pulse width modulation at OutputCool\_PWM

Time period of the PWM at output OutputCool\_PWM depending on Config.Output.Cool.PwmPeriode and the method for heating/cooling:

- Cool.PwmPeriode = 0.0 and cooling factor (Config.AdvancedCooling = FALSE):

The sampling time of the PID algorithm for heating (Retain.CtrlParams.Heat.Cycle) is used as time period of the PWM.

- Cool.PwmPeriode = 0.0 and PID parameter switching (Config.AdvancedCooling = TRUE):

The sampling time of the PID algorithm for cooling (Retain.CtrlParams.Cool.Cycle) is used as time period of the PWM.

- Cool.PwmPeriode > 0.0:

The value is rounded off to an integer multiple of the PID\_Temp sampling time (CycleTime.Value) and used as time period of the PWM.

The value must meet the following conditions:

- Cool.PwmPeriode ≤ Retain.CtrlParams.Cool.Cycle or Retain.CtrlParams.Heat.Cycle
- Cool.PwmPeriode > Config.Output.Cool.MinimumOnTime
- Cool.PwmPeriode > Config.Output.Cool.MinimumOffTime

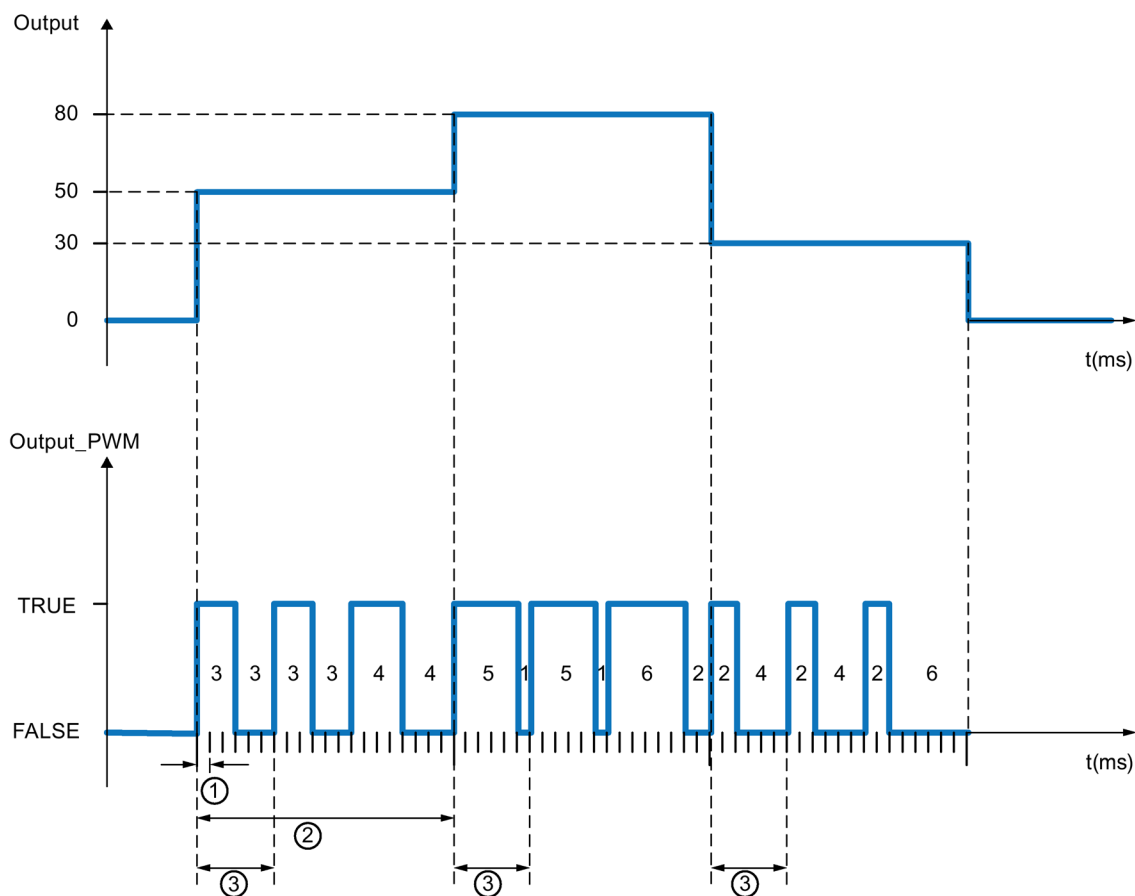


Config.Output.Cool.PwmPeriode is only effective if the cooling output is activated (Config.ActivateCooling =TRUE).

When you use PwmPeriode, the accuracy of the PWM output signal is determined by the relationship of PwmPeriode to the PID\_Temp sampling time (cycle time of the OB). PwmPeriode should be at least 10 times the PID\_Temp sampling time.

If the sampling time of the PID algorithm is not an integer multiple of PwmPeriode, each last period of the PWM within the sampling time of the PID algorithm is extended accordingly.

### Example for OutputHeat\_PWM



- ① PID\_Temp sampling time = 100.0 ms (cycle time of the calling cyclic interrupt OB, CycleTime.Value tag)
- ② PID algorithm sampling time = 2000.0 ms (Retain.CtrlParams.Heat.Cycle tag)
- ③ Time period of the PWM for heating = 600.0 ms (Config.Output.Heat.PwmPeriode tag)

**8.3.3.12 IntegralResetMode tag**

The IntegralResetMode tag determines how the integral action PIDCtrl.IOutputOld is pre-assigned:

- When switching from "Inactive" operating mode to "Automatic mode"
- With edge TRUE -> FALSE at parameter Reset and parameter Mode = 3

This setting only works for one cycle and is only effective if the integral action is activated (Retain.CtrlParams.Heat.Ti and Retain.CtrlParams.Cool.Ti > 0.0 tags).

IntegralReset-Mode	Description
0	<p>Smooth</p> <p>The value of PIDCtrl.IOutputOld is pre-assigned so that the switchover is bumpless, which means "Automatic mode" starts with the output value = 0.0 (parameter PidOutputSum) and there is no jump of the output value regardless of the control deviation (setpoint – process value).</p>
1	<p>Delete</p> <p>We recommend setting the weighting of the proportional action (Retain.CtrlParams.Heat.PWeighting and Retain.CtrlParams.Cool.PWeighting tags) to 1.0 if this option is used.</p> <p>The value of PIDCtrl.IOutputOld is deleted. Any control deviation will cause a jump of the PID output value. The direction of the output value jump depends on the active weighting of the proportional action (Retain.CtrlParams.Heat.PWeighting and Retain.CtrlParams.Cool.PWeighting tags) and the control deviation:</p> <ul style="list-style-type: none"> <li>• Active proportional action weighting = 1.0:                      Output value jump and control deviation have identical signs.                      Example: If the process value value is smaller than the setpoint (positive control deviation), the PID output value jumps to a positive value.</li> <li>• Active proportional action weighting &lt; 1.0:                      For large control deviations, the PID output value jump and control deviation have identical signs.                      Example: If the process value is much smaller than the setpoint (positive control deviation), the PID output value jumps to a positive value.                      For small control deviations, the PID output value jump and control deviation have different signs.                      Example: If the process value is just below the setpoint (positive control deviation), the PID output value jumps to a negative value. This is usually not desirable, because it results in a temporary increase in the control deviation.                      The smaller the configured weighting of the proportional action, the greater the control deviation must be to receive a PID output value jump with identical sign.</li> </ul> <p>We recommend setting the weighting of the proportional action (Retain.CtrlParams.Heat.PWeighting and Retain.CtrlParams.Cool.PWeighting tags) to 1.0 when this option is used. Otherwise, you may experience the undesirable behavior described for small control deviations. Alternatively, you can also use IntegralResetMode = 4. This option guarantees identical signs of the PID output value jump and control deviation independent of the configured weighting of the proportional action and the control deviation.</p>
2	<p>Hold</p> <p>The value of PIDCtrl.IOutputOld is not changed. You can define a new value using the user program.</p>

IntegralReset-Mode	Description
3	<p>Pre-assign</p> <p>The value of PIDCtrl.IOutputOld is automatically pre-assigned as if PidOutputSum = OverwriteInitialOutputValue in the last cycle.</p>
4	<p>Like setpoint change (only for PID_Temp with version <math>\geq 1.1</math>)</p> <p>The value of PIDCtrl.IOutputOld is automatically pre-assigned so that a similar PID output value jump results as for a PI controller in automatic mode in case of a setpoint change from the current process value to the current setpoint.</p> <p>Any control deviation will cause a jump of the PID output value. The PID output value jump and control deviation have identical signs.</p> <p>Example: If the process value value is smaller than the setpoint (positive control deviation), the PID output value jumps to a positive value. This is independent of the configured weighting of the proportional action and the control deviation.</p>

If IntegralResetMode is assigned a value outside the valid value range, PID\_Temp behaves as with the pre-assignment of IntegralResetMode:

- PID\_Temp up to V1.0: IntegralResetMode = 1
- PID\_Temp as of V1.1: IntegralResetMode = 4

### 8.3.4 CPU processing time and memory requirement PID\_Temp V1

#### CPU processing time

Typical CPU processing times of the PID\_Temp technology object as of Version 1.0, depending on CPU type.

CPU	Typ. CPU processing time PID_Temp V1
CPU 1211C ≥ V4.1	580 μs
CPU 1215C ≥ V4.1	580 μs
CPU 1217C ≥ V4.1	580 μs
CPU 1505S ≥ V1.0	50 μs
CPU 1510SP-1 PN ≥ V1.7	130 μs
CPU 1511-1 PN ≥ V1.7	130 μs
CPU 1512SP-1 PN ≥ V1.7	130 μs
CPU 1516-3 PN/DP ≥ V1.7	75 μs
CPU 1518-4 PN/DP ≥ V1.7	6 μs

#### Memory requirement

Memory requirement of an instance DB of the PID\_Temp technology object as of Version V1.0.

	Memory requirement of the instance DB of PID_Temp V1
Load memory requirement	Approx. 17000 bytes
Total work memory requirement	1280 bytes
Retentive work memory requirement	100 bytes

## 8.4 PID basic functions

### 8.4.1 CONT\_C

#### 8.4.1.1 Description CONT\_C

The CONT\_C instruction is used on SIMATIC S7 automation systems to control technical processes with continuous input and output variables. You can assign parameters to enable or disable sub-functions of the PID controller and adapt it to the process. In addition to the functions in the setpoint and process value branches, the instruction implements a complete PID controller with continuous output value output and the option of manually influencing the value of the output value.

#### Application

You can use the controller as a PID fixed setpoint controller, or in multi-loop control systems, also as a cascade, blending or ratio controller. The functions of the controller are based on the PID control algorithm of the sampling controller with an analog signal, if necessary extended by including a pulse shaper stage to generate pulse-width modulated output signals for two or three step controllers with proportional actuators.

#### Call

The CONT\_C instruction has an initialization routine that is run through when input parameter COM\_RST = TRUE is set. During initialization, the integral action is set to the initialization value I\_ITVAL. All the signal outputs are set to zero. COM\_RST = FALSE has to be set after the initialization routine has been completed.

The calculation of the values in the control blocks is only correct if the block is called at regular intervals. You should therefore call the control blocks in a cyclic interrupt OB (OB 30 to OB 38). Enter the sampling time in the CYCLE parameter.

If you call the instruction CONT\_C as a multiple instance DB, no technology object is created. No parameter assignment interface or commissioning interface is available. You must assign parameters for CONT\_C directly in the multiple instance DB and commission it via a watch table.

#### Error information

The error message word RET\_VAL is not evaluated by the block.

### 8.4.1.2 How CONT\_C works

#### Setpoint branch

The setpoint is entered in floating-point format at the SP\_INT input.

#### Process value branch

The process value can be input in I/O or floating-point format. The function CRP\_IN converts the I/O value PV\_PER to a floating-point format -100 to +100 % in accordance with the following rule:

$$\text{Output of CRP\_IN} = \text{PV\_PER} * 100 / 27648$$

The PV\_NORM function scales the output of CRP\_IN according to the following rule:

$$\text{Output of PV\_NORM} = (\text{output of CRP\_IN}) * \text{PV\_FAC} + \text{PV\_OFF}$$

PV\_FAC has a default of 1 and PV\_OFF a default of 0.

#### Forming the error signal

The difference between the setpoint and process value is the error signal. To suppress a minor sustained oscillation due to manipulated variable quantization (e.g. with a pulse width modulation with PULSEGEN), the error signal is applied to a dead band (DEADBAND). With DEADB\_W = 0, the dead band is switched off.

#### PID Algorithm

The PID algorithm operates as a position algorithm. The proportional, integral (INT), and differential (DIF) actions are connected in parallel and can be activated or deactivated individually. This allows P, PI, PD, and PID controllers to be configured. Pure I controllers are also possible.

#### Manual value processing

It is possible to switch over between manual and automatic mode. In manual mode, the manipulated variable is corrected to a manually selected value.

The integral action (INT) is set internally to LMN - LMN\_P - DISV and the derivative action (DIF) is set to 0 and synchronized internally. Changeover to automatic mode is therefore bumpless.

### Manipulated value processing

You can use the LMNLIMIT function to limit the manipulated value to selected values. Alarm bits indicate when a limit is exceeded by the input variable.

The LMN\_NORM function normalizes the output of LMNLIMIT according to the following rule:

$$\text{LMN} = (\text{output of LMNLIMIT}) * \text{LMN\_FAC} + \text{LMN\_OFF}$$

LMN\_FAC has a default of 1 and LMN\_OFF a default of 0.

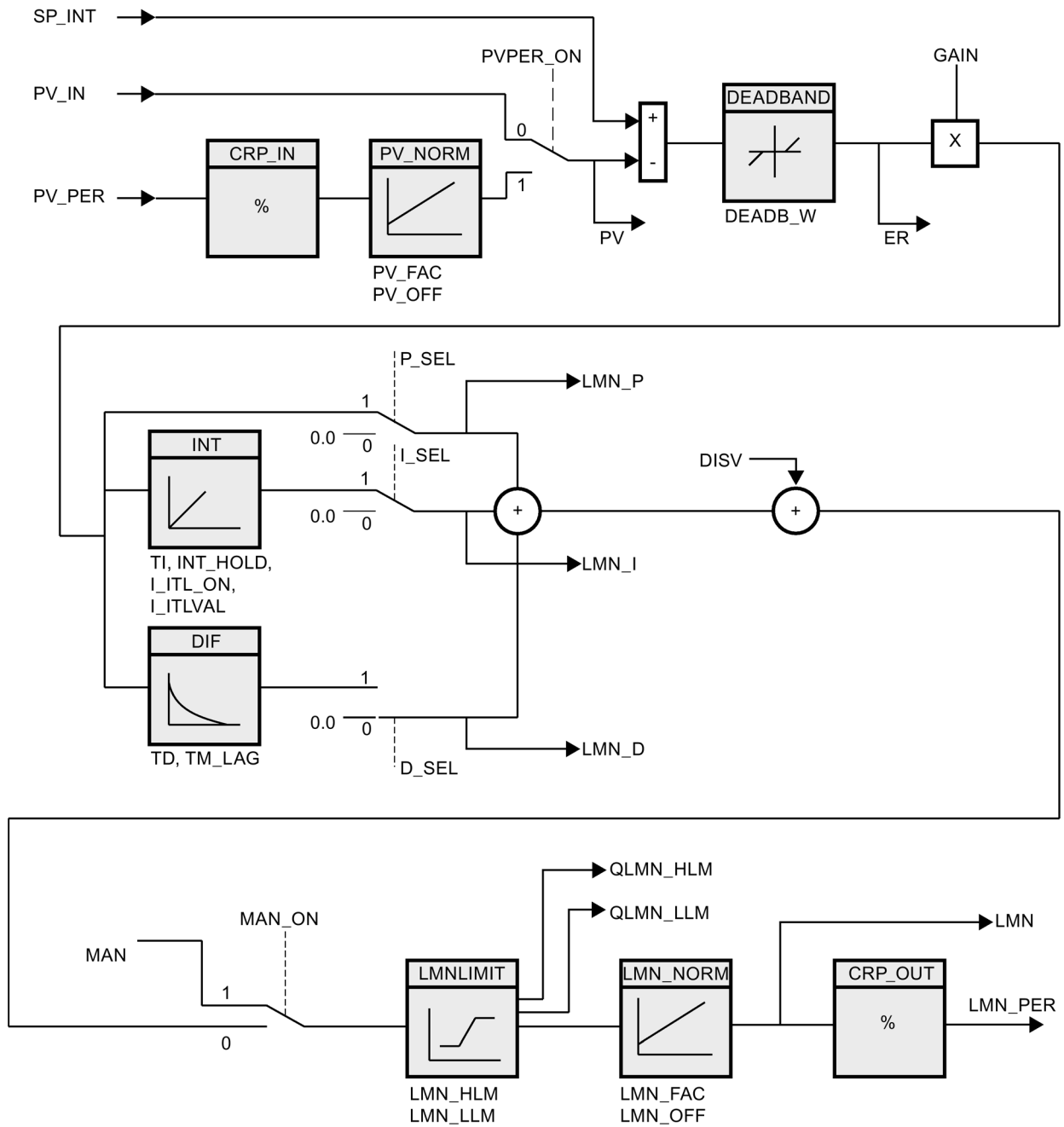
The manipulated value is also available in I/O format. The CRP\_OUT function converts the LMN floating-point value to a I/O value according to the following rule:

$$\text{LMN\_PER} = \text{LMN} * 27648 / 100$$

### Feedforward control

A disturbance variable can be added at the DISV input.

8.4.1.3 CONT\_C block diagram





## 8.4.1.4 Input parameter CONT\_C

Table 8- 13

Parameters	Data type	Default	Description
COM_RST	BOOL	FALSE	The instruction has an initialization routine that is processed when the "Restart" input is set.
MAN_ON	BOOL	TRUE	If the input "Enable manual mode" is set then the control loop is interrupted. A manual value is set as the manipulated value.
PVPER_ON	BOOL	FALSE	If the process value is to be read in from the I/Os, the PV_PER input must be interconnected with the I/Os and the "Enable process value I/Os" input must be set.
P_SEL	BOOL	TRUE	The PID actions can be switched on and off individually in the PID algorithm. P-action is on when the "Enable P-action" input is set.
I_SEL	BOOL	TRUE	The PID actions can be switched on and off individually in the PID algorithm. I action is on when the input "I-action on" is set.
INT_HOLD	BOOL	FALSE	The output of the integral action can be frozen. For this the input "I-action hold" must be set.
I_ITL_ON	BOOL	FALSE	The output of the integral action can be set at the I_ITLVAL input. For this the input "Set I-action" must be set.
D_SEL	BOOL	FALSE	The PID actions can be switched on and off individually in the PID algorithm. D-action is on when the input "Enable D-action" is set.
CYCLE	TIME	T#1s	The time between block calls must be constant. The "Sampling time" input specifies the time between block calls. CYCLE >= 1ms
SP_INT	REAL	0.0	The input "Internal setpoint" is used to specify a setpoint. Permissible are values from -100 to 100 % or a physical variable 1).
PV_IN	REAL	0.0	At the "Process value input" you can assign parameters to a commissioning value or you can interconnect an external process value in floating-point format. Permissible are values from -100 to 100 % or a physical variable 1).
PV_PER	WORD	W#16#0000	The process value in I/O format is interconnected with the controller at the "Process value I/O" input.
MAN	REAL	0.0	The "Manual value" input is used to set a manual value using the operator interface functions. Permissible are values from -100 to 100 % or a physical variable 2).
GAIN	REAL	2.0	The "Proportional gain" input specifies controller amplification.
TI	TIME	T#20s	The "Integration time" input determines the time response of the integral action. TI >= CYCLE
TD	TIME	T#10s	The "Derivative action time" input determines the time response of the derivative action. TD >= CYCLE
TM_LAG	TIME	T#2s	Time lag of the D-action The algorithm of the D-action contains a delay for which parameters can be assigned at the input "Time lag of the D-action". TM_LAG >= CYCLE/2
DEADB_W	REAL	0.0	A dead band is applied to the system deviation. The "Dead band width" input determines the size of the dead band. DEADB_W >= 0.0 (%) or a physical variable 1)

8.4 PID basic functions

Parameters	Data type	Default	Description
LMN_HLM	REAL	100.0	The manipulated value is always restricted to a high limit and low limit. The "High limit of manipulated value" input specifies the high limit. Permissible are real values starting at LMN_LLM or a physical variable 2).
LMN_LLM	REAL	0.0	The manipulated value is always restricted to a high limit and low limit. The "Low limit of manipulated value" input specifies the low limit. Permissible are real values up to LMN_HLM or a physical variable 2).
PV_FAC	REAL	1.0	The "Process value factor" input is multiplied by the process value. The input is used to scale the process value range.
PV_OFF	REAL	0.0	The input "Process value offset" is added to the process value. The input is used to scale the process value range.
LMN_FAC	REAL	1.0	The "Manipulated value factor" input is multiplied with the manipulated value. The input is used to scale the manipulated value range.
LMN_OFF	REAL	0.0	The input "Manipulated value offset" is added to the process value. The input is used to scale the manipulated value range.
I_ITLVAL	REAL	0.0	The output of the integral action can be set at the I_ITL_ON input. The initialization value is applied to the input "Initialization value of the I-action." Permissible are values of -100.0 to 100.0 (%) or a physical variable 2).
DISV	REAL	0.0	For feedforward control, the disturbance variable is interconnected to the "Disturbance variable" input. Permissible are values of -100.0 to 100.0 (%) or a physical variable 2).

- 1) Parameters in the setpoint and process value branches with the same unit
- 2) Parameters in the manipulated value branch with the same unit

### 8.4.1.5 Output parameters CONT\_C

Table 8- 14

Parameter	Data type	Default	Description
LMN	REAL	0.0	The effective "Manipulated value" is output in floating point format at the "Manipulated value" output.
LMN_PER	WORD	W#16#0000	The manipulated value in I/O format is interconnected on the input "Manipulated value I/O" with the controller.
QLMN_HLM	BOOL	FALSE	The manipulated value is always restricted to a high limit and low limit. The output "High limit of manipulated value reached" indicates that the high limit has been reached.
QLMN_LLM	BOOL	FALSE	The manipulated value is always restricted to a high limit and low limit. The output "Low limit of manipulated value reached" indicates that the low limit has been reached.
LMN_P	REAL	0.0	The "P-action" output contains the proportional action of the manipulated variable.
LMN_I	REAL	0.0	The "I-action" output contains the integral action of the manipulated variable.
LMN_D	REAL	0.0	The "D-action" output contains the derivative action of the manipulated variable.
PV	REAL	0.0	The effective process value is output at the "Process value" output.
ER	REAL	0.0	The effective system deviation is output at the "Error signal" output.

## 8.4.2 CONT\_S

### 8.4.2.1 Description CONT\_S

The CONT\_S instruction is used on SIMATIC S7 automation systems to control technical processes with binary output value output signals for actuators with integrating behavior. During parameter assignment, you can activate or deactivate sub-functions of the PI step controller to adapt the controller to the controlled system. In addition to the functions in the process value branch, the instruction implements a complete proportional-plus-integral-action controller with binary output value output and the option of manually influencing the value of the output value. The step controller operates without a position feedback signal.

### Application

You can use the controller as a PI fixed setpoint controller or in secondary control loops in cascade, blending or ratio controllers, however you cannot use it as the primary controller. The functions of the controller are based on the PI control algorithm of the sampling controller supplemented by the functions for generating the binary output signal from the analog actuating signal.

### Call

The CONT\_S instruction has an initialization routine that is run through when input parameter COM\_RST = TRUE is set. All the signal outputs are set to zero. COM\_RST = FALSE has to be set after the initialization routine has been completed.

The calculation of the values in the control blocks is only correct if the block is called at regular intervals. You should therefore call the control blocks in a cyclic interrupt OB (OB 30 to OB 38). Enter the sampling time in the CYCLE parameter.

If you call the instruction CONT\_S as a multiple instance DB, no technology object is created. No parameter assignment interface or commissioning interface is available. You must assign parameters for CONT\_S directly in the multiple instance DB and commission it via a watch table.

### Error information

The error message word RET\_VAL is not evaluated by the block.

### 8.4.2.2 Mode of operation CONT\_S

#### Setpoint branch

The setpoint is entered in floating-point format at the SP\_INT input.

#### Process value branch

The process value can be input in I/O or floating-point format. The function CRP\_IN converts the I/O value PV\_PER to a floating-point format -100 to +100 % in accordance with the following rule:

$$\text{Output of CRP\_IN} = \text{PV\_PER} * 100 / 27648$$

The PV\_NORM function normalizes the output of CRP\_IN according to the following rule:

$$\text{Output of PV\_NORM} = (\text{output of CRP\_IN}) * \text{PV\_FAC} + \text{PV\_OFF}$$

PV\_FAC has a default of 1 and PV\_OFF a default of 0.

#### Forming the error signal

The difference between the setpoint and process value is the error signal. To suppress a small constant oscillation due to the manipulated variable quantization (for example, due to a limited resolution of the manipulated value by the control valve), a dead band is applied to the error signal (DEADBAND). With DEADB\_W = 0, the dead band is switched off.

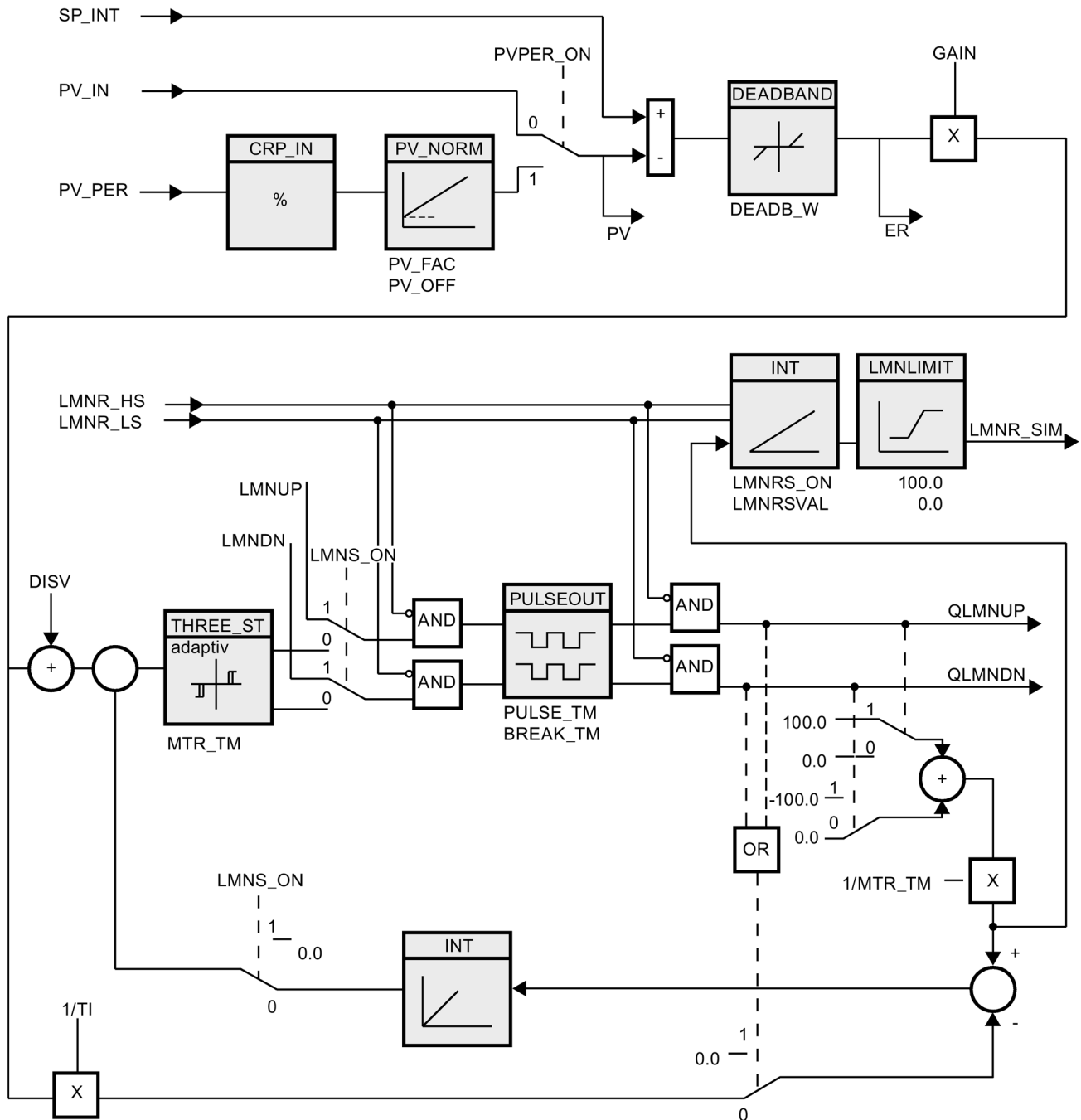
#### PI step algorithm

The instruction operates without position feedback. The I-action of the PI algorithm and the assumed position feedback signal are calculated in **one** integral action (INT) and compared with the remaining P-action as a feedback value. The difference is applied to a three-step element (THREE\_ST) and a pulse shaper (PULSEOUT) that generates the pulses for the control valve. The switching frequency of the controller can be reduced by adapting the response threshold of the three-step element.

#### Feedforward control

A disturbance variable can be added at the DISV input.

8.4.2.3 Block diagram CONT\_S



## 8.4.2.4 Input parameters CONT\_S

Table 8- 15

Parameters	Data type	Default	Description
COM_RST	BOOL	FALSE	The block has an initialization routine that is processed when the "Restart" input is set.
LMNR_HS	BOOL	FALSE	The signal "Control valve at high endstop" is interconnected at the input "High endstop signal of position feedback". LMNR_HS=TRUE means: The control valve is at high endstop.
LMNR_LS	BOOL	FALSE	The signal "Control valve at low endstop" is interconnected on the input "Low endstop signal of position feedback". LMNR_LS=TRUE means The control valve is at low endstop.
LMNS_ON	BOOL	FALSE	Manipulated value signal processing is switched to manual mode at the "Enable manual mode of manipulated signal".
LMNUP	BOOL	FALSE	The output signal QLMNUP is operated in manual mode of the manipulated value signals at the input "Manipulated value signal up".
LMNDN	BOOL	FALSE	The output signal QLMNDN is operated in manual mode of the manipulated value signals at the input "Manipulated value signal down"
PVPER_ON	BOOL	FALSE	If the process value is to be read from the I/O then the input PV_PER must be interconnected with the I/O and the input "Enable process value I/O" must be set.
CYCLE	TIME	T#1s	The time between block calls must be constant. The "Sampling time" input specifies the time between block calls. CYCLE >= 1ms
SP_INT	REAL	0.0	The input "Internal setpoint" is used to specify a setpoint. Permissible are values from -100 to 100 % or a physical variable <sup>1)</sup> .
PV_IN	REAL	0.0	At the "Process value input" you can assign parameters to a commissioning value or you can interconnect an external process value in floating-point format. Permissible are values from -100 to 100 % or a physical variable <sup>1)</sup> .
PV_PER	WORD	W#16#0000	The process value in I/O format is interconnected with the controller at the "Process value I/O" input.
GAIN	REAL	2.0	The "Proportional gain" input specifies controller amplification.
TI	TIME	T#20s	The "Integration time" input determines the time response of the integral action. TI >= CYCLE
DEADB_W	REAL	1.0	A dead band is applied to the system deviation. The "Dead band width" input determines the size of the dead band. Permissible are values from 0 to 100 % or a physical variable <sup>1)</sup> .
PV_FAC	REAL	1.0	The "Process value factor" input is multiplied by the process value. The input is used to scale the process value range.
PV_OFF	REAL	0.0	The input "Process value offset" is added to the process value. The input is used to scale the process value range.
PULSE_TM	TIME	T#3s	You can assign a minimum pulse time at the parameter "Minimum pulse time". PULSE_TM >= CYCLE
BREAK_TM	TIME	T#3s	You can assign a minimum break time at the parameter "Minimum break time". BREAK_TM >= CYCLE

Parameters	Data type	Default	Description
MTR_TM	TIME	T#30s	The time required by the actuator to move from limit stop to limit stop is entered at the "Motor actuating time" parameter. MTR_TM >= CYCLE
DISV	REAL	0.0	For feedforward control, the disturbance variable is interconnected to the "Disturbance variable" input. Permissible are values from -100 to 100 % or a physical variable <sup>2)</sup> .

1) Parameters in setpoint and process value branches with identical unit

2) Parameters in the manipulated value branch with same unit

### 8.4.2.5 Output parameters CONT\_S

Table 8- 16

Parameters	Data type	Default	Description
QLMNUP	BOOL	FALSE	If the output "Manipulated value signal up" is set then the control valve should be open.
QLMNDN	BOOL	FALSE	If the output "Manipulated value signal down" is set then the control valve should be closed.
PV	REAL	0.0	The effective process value is output at the "Process value" output.
ER	REAL	0.0	The effective system deviation is output at the "Error signal" output.



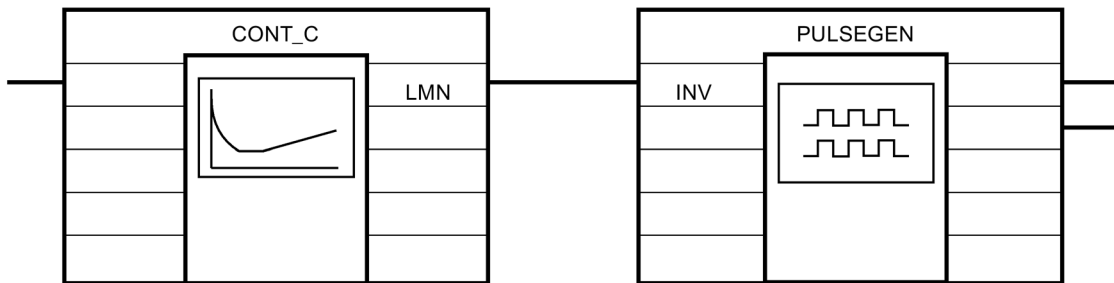
## 8.4.3 PULSEGEN

### 8.4.3.1 Description PULSEGEN

The instruction PULSEGEN serves as the structure of a PID controller with impulse output for proportional actuators. PULSEGEN transforms the input value INV (= LMN of the PID controller) through modulation of the impulse width in an impulse sequence with a constant period duration, which corresponds with the cycle time with which the input value is updated.

### Application

You can use the PULSEGEN instruction to configure two- or three-step PID controllers with pulse width modulation. The function is normally used in conjunction with the continuous controller CONT\_C.



### Call

The PULSEGEN instruction has an initialization routine that is run through when input parameter COM\_RST = TRUE is set. All the signal outputs are set to zero. COM\_RST = FALSE has to be set after the initialization routine has been completed.

The calculation of the values in the control blocks is only correct if the block is called at regular intervals. You should therefore call the control blocks in a cyclic interrupt OB (OB 30 to OB 38). Enter the sampling time in the CYCLE parameter.

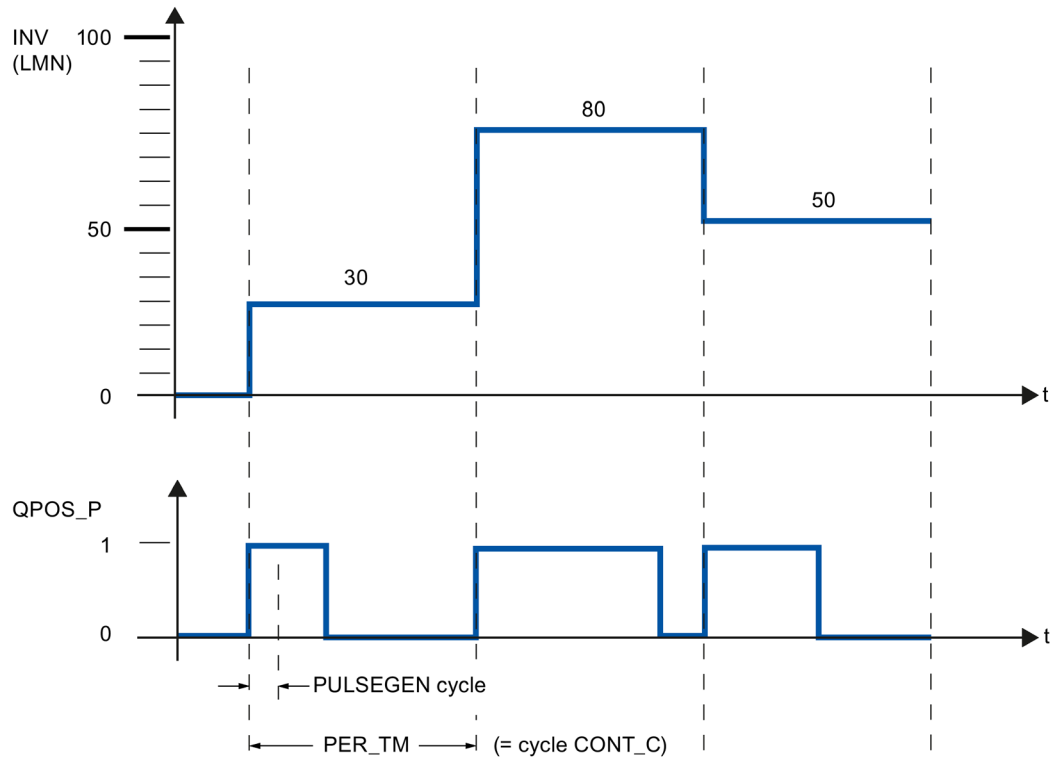
### Responses in the event of an error

The error message word RET\_VAL is not evaluated by the block.

### 8.4.3.2 Mode of operation PULSEGEN

#### Impulse width modulation

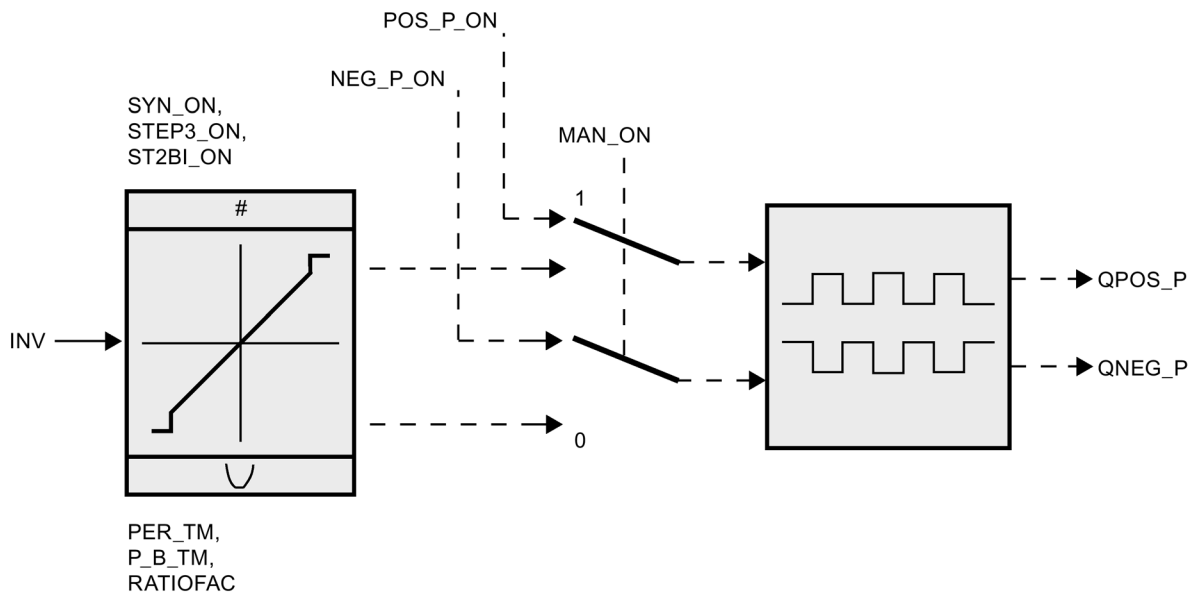
The duration of a pulse per period duration is proportional to the input variable. The cycle assigned via PER\_TM is not identical to the processing cycle of the PULSEGEN instruction. Rather, a PER\_TM cycle is made up of several processing cycles of the PULSEGEN instruction, whereby the number of PULSEGEN calls per PER\_TM cycle determines the accuracy of the pulse width.



An input variable of 30% and 10 PULSEGEN calls per PER\_TM mean the following:

- "One" at the QPOS\_P output for the first three calls of PULSEGEN (30% of 10 calls)
- "Zero" at the QPOS\_P output for seven further calls of PULSEGEN (70% of 10 calls)

## Block diagram



## Accuracy of the manipulated value

With a "Sampling ratio" of 1:10 (CONT\_C calls to PULSEGEN calls) the accuracy of the manipulated value in this example is restricted to 10%, in other words, set input values INV can only be simulated by a pulse duration at the QPOS\_P output in steps of 10 %.

The accuracy is increased as the number of PULSEGEN calls per CONT\_C call is increased.

If PULSEGEN is called, for example, 100 times more often than CONT\_C, a resolution of 1 % of the manipulated value range is achieved.

---

### Note

The reduction ratio of the call frequency must be programmed by the user.

---

## Automatic synchronization

It is possible to automatically synchronize the pulse output with the instruction that updates the input variable INV (e.g. CONT\_C). This ensures that a change in the input variable is output as quickly as possible as a pulse.

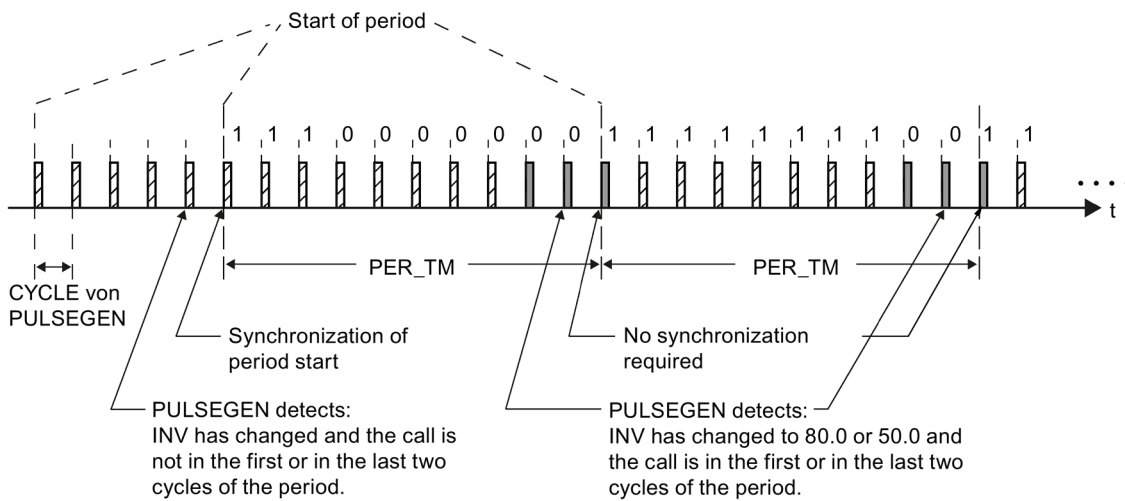
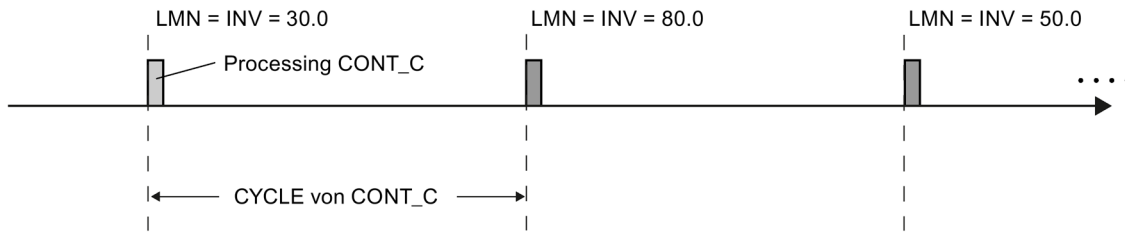
The pulse shaper evaluates the input value INV at intervals corresponding to the period duration PER\_TM and converts the value into a pulse signal of corresponding length.

Since, however, INV is usually calculated in a slower cyclic interrupt class, the pulse shaper should start the conversion of the discrete value into a pulse signal as soon as possible after the updating of INV.


8.4 PID basic functions

To allow this, the block can synchronize the start of the period using the following procedure:

If INV changes and if the block call is not in the first or last two call cycles of a period, a synchronization is performed. The pulse duration is recalculated and in the next cycle is output with a new period.



 Processing PULSEGEN

 PULSEGEN processing in the first or in the last two cycles of the period.

The automatic synchronization is switched off, if SYN\_ON = FALSE.

**Note**

The start of a new period and subsequent synchronization usually leads to a certain imprecision when the old value of INV (i.e. of LMN) is mapped to the pulse signal.

### 8.4.3.3 Mode of operation PULSEGEN

#### Modes

Depending on the parameters assigned to the pulse shaper, PID controllers with a three-step output or with a bipolar or unipolar two-step output can be configured. The following table illustrates the setting of the switch combinations for the possible modes.

Mode	MAN_ON	STEP3_ON	ST2BI_ON
Three-step control	FALSE	TRUE	Any
Two-step control with bi-polar Manipulating range (-100 % to 100 %)	FALSE	FALSE	TRUE
Two-step control with unipolar Manipulating range (0 % to 100 %)	FALSE	FALSE	FALSE
Manual mode	TRUE	Any	Any

#### Manual mode in two/three-step control

In the manual mode (MAN\_ON = TRUE), the binary outputs of the three-step or two-step controller can be set using the signals POS\_P\_ON and NEG\_P\_ON regardless of INV.

Control	POS_P_ON	NEG_P_ON	QPOS_P	QNEG_P
Three-step control	FALSE	FALSE	FALSE	FALSE
	TRUE	FALSE	TRUE	FALSE
	FALSE	TRUE	FALSE	TRUE
	TRUE	TRUE	FALSE	FALSE
Two-step control	FALSE	Any	FALSE	TRUE
	TRUE	Any	TRUE	FALSE

### 8.4.3.4 Three-step control

#### Three-step control

In "Three-step control" mode, it is possible to generate three statuses of the actuating signal. For this, the status values of the binary output signals QPOS\_P and QNEG\_P are assigned to the respective operating statuses of the actuator. The table shows the example of a temperature control:

Output signals	Heat	Off	Cool
QPOS_P	TRUE	FALSE	FALSE
QNEG_P	FALSE	FALSE	TRUE

The pulse duration is calculated from the input variable via a characteristic curve. The form of the characteristic curve is defined by the minimum pulse duration or minimum interval and the ratio factor. The normal value for the ratio factor is 1.

The "doglegs" in the curves are caused by the minimum pulse duration or minimum interval.

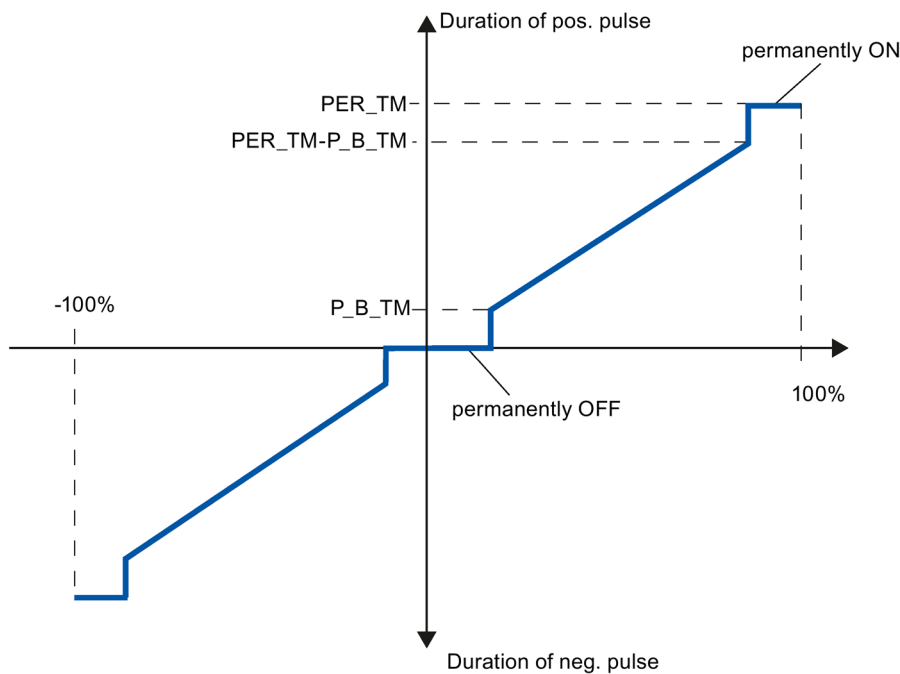
**Minimum pulse duration or minimum interval**

A correctly assigned minimum pulse duration or minimum interval P\_B\_TM can prevent short on/off times, which reduce the working life of switching elements and actuators. Small absolute values of input variable LMN that would otherwise generate a pulse duration shorter than P\_B\_TM are suppressed. Large input values that would generate a pulse duration longer than PER\_TM - P\_B\_TM are set to 100% or -100%.

The duration of positive or negative pulses is calculated by multiplying the input variable (in %) by the period duration:

$$\text{Pulse duration} = \text{INV} / 100 * \text{PER\_TM}$$

The following figure shows a symmetrical characteristic curve of the three-step controller (ratio factor = 1).



**Asymmetrical three-step control**

Using the ratio factor RATIOFAC, the ratio of the duration of positive to negative pulses can be changed. In a thermal process, for example, this would allow different system time constants for heating and cooling.

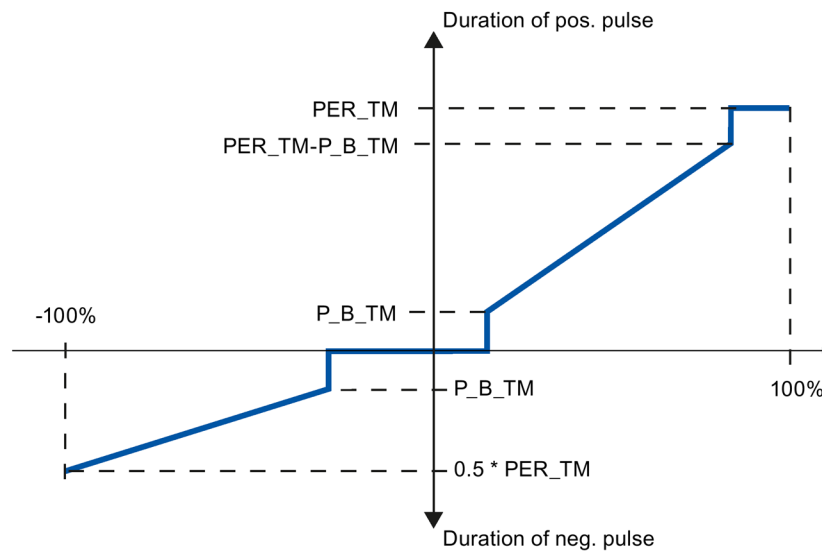
**Ratio factor < 1**

The pulse duration at the negative pulse output, calculated by multiplying the input variable by the period duration, is multiplied by the ratio factor.

$$\text{Positive pulse duration} = \text{INV} / 100 * \text{PER\_TM}$$

$$\text{Negative pulse duration} = \text{INV} / 100 * \text{PER\_TM} * \text{RATIOFAC}$$

The following figure shows the asymmetrical characteristic curve of the three-step controller (ratio factor = 0.5):



### Ratio factor > 1

The pulse duration at the positive pulse output, calculated by multiplying the input variable by the period duration, is divided by the ratio factor.

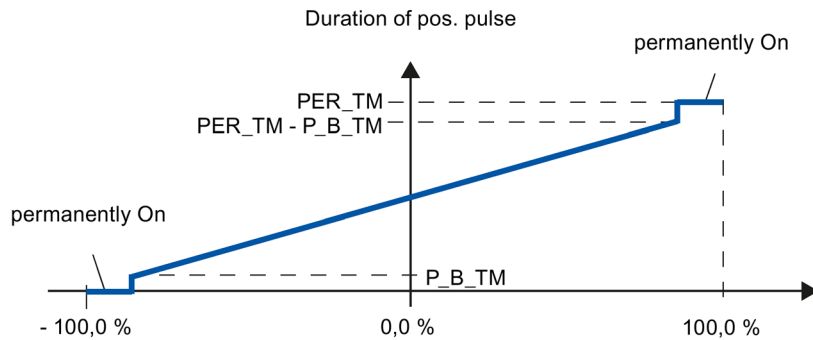
$$\text{Positive pulse duration} = \text{INV} / 100 * \text{PER\_TM} / \text{RATIOFAC}$$

$$\text{Negative pulse duration} = \text{INV} / 100 * \text{PER\_TM}$$

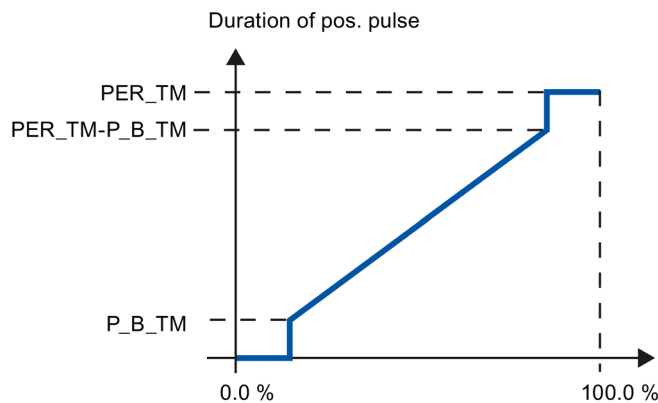
### 8.4.3.5 Two-step control

In two-step control, only the positive pulse output QPOS\_P of PULSEGEN is connected to the on/off actuator. Depending on the manipulated value range used, the two-step controller has a bipolar or a unipolar manipulated value range.

#### Two-step control with bipolar manipulated variable range (-100% to 100%)



#### Two-step control with unipolar manipulated variable range (0% to 100%)



The negated output signal is available at QNEG\_P if the connection of the two-step controller in the control loop requires a logically inverted binary signal for the actuating pulses.

Pulse	Actuator On	Actuator Off
QPOS_P	TRUE	FALSE
QNEG_P	FALSE	TRUE



### 8.4.3.6 Input parameters PULSEGEN

The values of the input parameters are not limited in the block. There is no parameter check.

Table 8- 17

Parameters	Data type	Default	Description
INV	REAL	0.0	At the input parameter "Input variable" an analog manipulated variable is connected. Values from -100 to 100 % are permitted.
PER_TM	TIME	T#1s	At the parameter "Period duration" the constant period duration of the pulse width modulation is entered. This corresponds to the sampling time of the controller. The ratio between the sampling time of the pulse shaper and the sampling time of the controller determines the accuracy of the pulse width modulation. PER_TM >=20*CYCLE
P_B_TM	TIME	T#50 ms	You can assign a minimum pulse/break time at the parameter "Minimum pulse/break time". P_B_TM >= CYCLE
RATIOFAC	REAL	1.0	Using the "Ratio factor" input parameter the ratio of the duration of positive to negative pulses can be changed. In a thermal process, this would, for example, allow different time constants for heating and cooling to be compensated (for example, in a process with electrical heating and water cooling). Values from 0.1 to 10.0 are permitted.
STEP3_ON	BOOL	TRUE	At the input parameter "Enable three-step control" the appropriate mode is activated. In three-step control both output signals are active.
ST2BI_ON	BOOL	FALSE	At the input parameter "Enable two-step control for bipolar manipulated value range" you can select from the modes "Two-step control for bipolar manipulated value range" and "Two-step control for unipolar manipulated value range". STEP3_ON = FALSE is required.
MAN_ON	BOOL	FALSE	Setting the input parameter "Enable manual mode" allows the output signals to be set manually.
POS_P_ON	BOOL	FALSE	For manual mode three-step control, the output signal QPOS_P can be operated on the input parameter "Positive pulse on". In manual mode with two-step control, QNEG_P is always set inversely to QPOS_P.
NEG_P_ON	BOOL	FALSE	For manual mode three-step control, the output signal QNEG_P can be operated on the input parameter "Negative pulse on". In manual mode with two-step control, QNEG_P is always set inversely to QPOS_P.
SYN_ON	BOOL	TRUE	By setting the input parameter "Enable synchronization", it is possible to synchronize the pulse output automatically with the block that updates the input variable INV. This ensures that a change in the input variable is output as quickly as possible as a pulse.
COM_RST	BOOL	FALSE	The block has an initialization routine that is processed when the input "Restart" is set.
CYCLE	TIME	T#10ms	The time between block calls must be constant. The "Sampling time" input specifies the time between block calls. CYCLE >= 1ms

### 8.4.3.7 Output parameter PULSEGEN

Table 8- 18

Parameters	Data type	Default	Description
QPOS_P	BOOL	FALSE	The output parameter "Output signal positive pulse" is set if a pulse will be output. In three-step control, this is always the positive pulse. In two-step control, the QNEG_P is always set inversely to QPOS_P.
QNEG_P	BOOL	FALSE	The output parameter "Output signal negative pulse" is set if a pulse will be output. In three-step control, this is always the negative pulse. In two-step control, QNEG_P is always set inversely to QPOS_P.

## 8.4.4 TCONT\_CP

### 8.4.4.1 Description TCONT\_CP

The instruction TCONT\_CP is used to control temperature processes with continuous or pulsed control signals. The controller functionality is based on the PID control algorithm with additional functions for temperature processes. To improve the control response with temperature processes, the block includes a control zone and reduction of the proportional component if there is a setpoint step change.

The instruction can set the PI/PID parameters itself using the controller optimization function.

### Application

The controller controls one actuator; in other words, with one controller you can either heat or cool but not both. If you use the block for cooling, GAIN must be assigned a negative value. This inversion of the controller means that if the temperature rises, for example, the manipulated variable LMN and with it the cooling action is increased.

### Call

The instruction TCONT\_CP must be called equidistant. To achieve this, use a cyclic interrupt priority class (for example, OB35 for an S7-300).

The TCONT\_CP instruction has an initialization routine that is run through when input parameter COM\_RST = TRUE is set. During initialization, the integral action is set to the initialization value I\_ITVAL. All the signal outputs are set to zero. Following execution of the initialization routine, the block sets COM\_RST back to FALSE. If you require initialization when the CPU restarts, call the block in OB100 with COM\_RST = TRUE.

If you call the instruction TCONT\_CP as a multiple instance DB, no technology object is created. No parameter assignment interface or commissioning interface is available. You must assign parameters for TCONT\_CP directly in the multiple instance DB and commission it via a watch table.

### See also

Operating principle of the pulse generator (Page 493)

Block diagram TCONT\_CP (Page 496)

### 8.4.4.2 Mode of operation TCONT\_CP

#### Setpoint branch

The setpoint is entered at input SP\_INT in floating-point format as a physical value or percentage. The setpoint and process value used to form the control deviation must have the same unit.

#### Process value options (PVPER\_ON)

Depending on PVPER\_ON, the process value can be read in, in the I/O or floating-point format.

PVPER_ON	Process Value Input
TRUE	The process value is read in via the analog I/Os (PIW xxx) at input PV_PER.
FALSE	The process value is acquired in floating-point format at input PV_IN.

#### Process value format conversion CRP\_IN (PER\_MODE)

The CRP\_IN function converts the I/O value PV\_PER to floating-point format depending on the PER\_MODE switch according to the following rules:

PER_MODE	Output of CRP_IN	Analog Input Type	Unit
0	PV_PER * 0.1	Thermoelements; PT100/NI100; standard	°C;°F
1	PV_PER * 0.01	PT100/NI100; climate;	°C;°F
2	PV_PER * 100/27648	Voltage/current	%

#### Process value scaling PV\_NORM (PF\_FAC, PV\_OFFS)

The PV\_NORM function calculates the output of CRP\_IN according to the following rule:

$$\text{"Output of PV\_NORM"} = \text{"Output of CRP\_IN"} * \text{PV\_FAC} + \text{PV\_OFFS}$$

It can be used for the following purposes:

- Process value adjustment with PV\_FAC as process value factor and PV\_OFFS as process value offset.
- Scaling of temperature to percentage

You want to enter the setpoint as a percentage and must now convert the measured temperature value to a percentage.

- Scaling of percentage to temperature

You want to enter the setpoint in the physical temperature unit and must now convert the measured voltage/current value to a temperature.

Calculation of the parameters:

- $PV\_FAC = \text{range of } PV\_NORM / \text{range of } CRP\_IN$ ;
- $PV\_OFFS = LL (PV\_NORM) - PV\_FAC * LL(CRP\_IN)$ ;

where LL: Low limit

The scaling is switched off with the default values ( $PV\_FAC = 1.0$  and  $PV\_OFFS = 0.0$ ). The effective process value is output at the PV output.

---

### Note

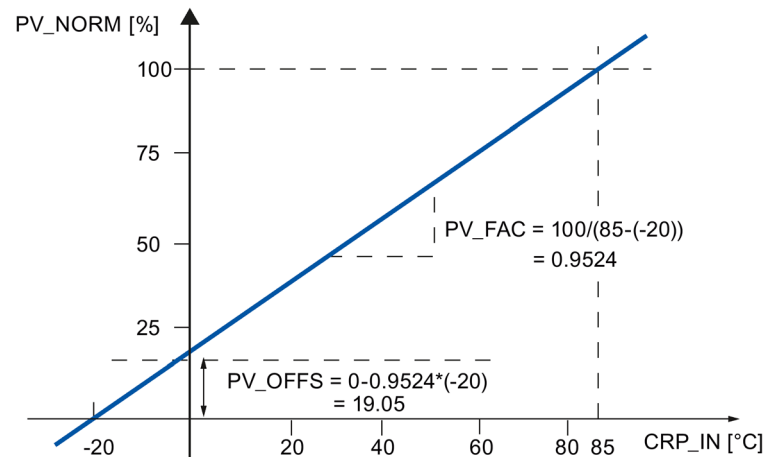
With pulse control, the process value must be transferred to the block in the fast pulse call (reason: mean value filtering). Otherwise, the control quality can deteriorate.

---

### Example of Process Value Scaling

If you want to enter the setpoint as a percentage, and you have a temperature range of -20 to 85 °C applied to , CRP\_IN you must normalize the temperature range as a percentage.

The diagram below shows an example of adapting the temperature range -20 to 85 °C to an internal scale of 0 to 100 %:



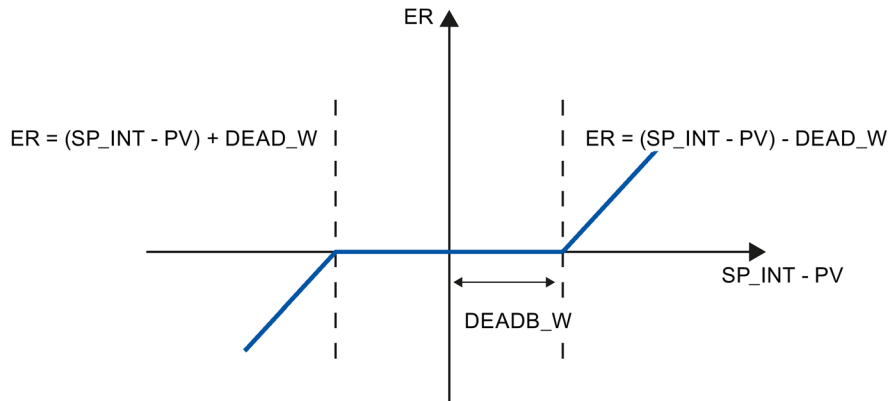
### Forming the control deviation

The difference between the setpoint and process value is the control deviation before the dead band.

The setpoint and process value must exist in the same unit.

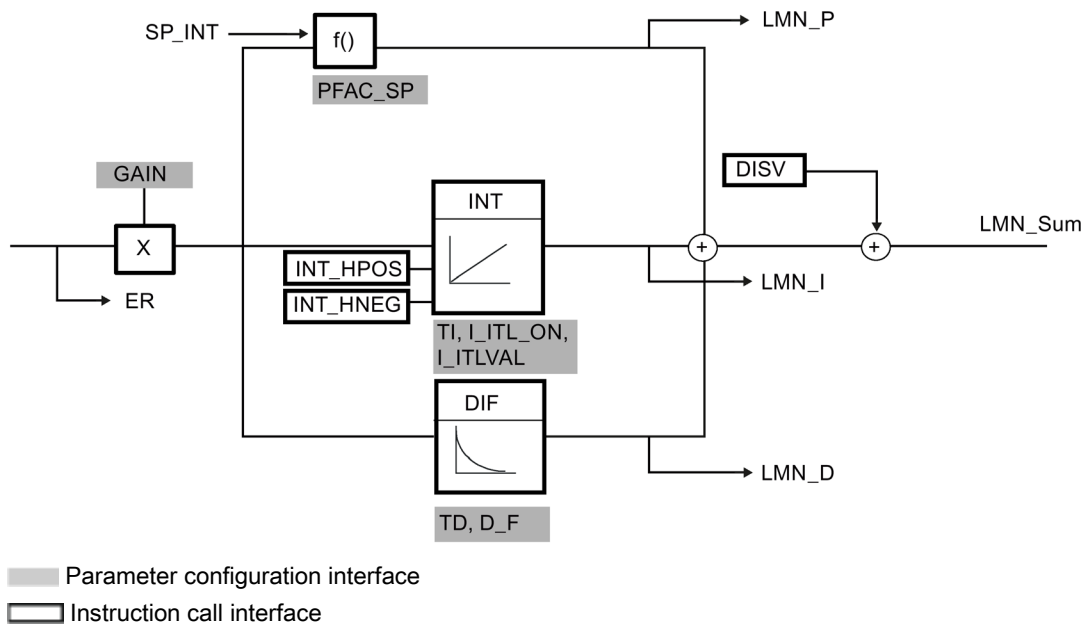
### Dead band (DEADB\_W)

To suppress a minor sustained oscillation due to the manipulated variable quantization (for example, in pulse width modulation with PULSEGEN) a dead band is applied to the (DEADBAND) control deviation. With DEADB\_W = 0.0, the dead band is disabled. The effective control deviation is indicated by the ER parameter.



### PID Algorithm

The following figure shows the block diagram of the PID algorithm.



### PID Algorithm (GAIN, TI, TD, D\_F)

The PID algorithm operates as a position algorithm. The proportional, integral (INT), and derivative (DIF) actions are connected in parallel and can be activated or deactivated individually. This allows P, PI, PD, and PID controllers to be configured.

Controller tuning supports PI and PID controllers. Controller inversion is implemented using a negative GAIN (cooling controller).

If you set TI and TD to 0.0, you obtain a pure P controller at the operating point.

The step response in the time range is:

$$LMN\_Sum(t) = GAIN * ER(0) \left( 1 + \frac{1}{TI} * t + D\_F * e^{-\frac{t}{TD/D\_F}} \right)$$

Where:

LMN\_Sum(t) the manipulated variable in the controller's automatic mode

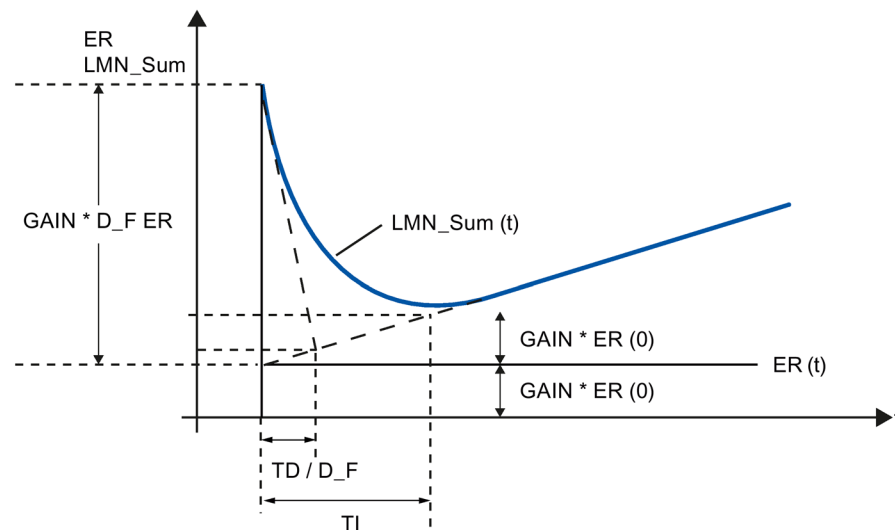
ER (0) is the step height of the normalized control deviation

GAIN is the controller gain

TI is the integration time

TD is the derivative action time

D\_F is the derivative factor



### Integral action (TI, I\_ITL\_ON, I\_ITLVAL)

In manual mode, it is corrected as follows:  $LMN_I = LMN - LMN_P - DISV$ .

If the output value is limited, the integral action is halted. If the control deviation moves the integral action back in the direction of the output range, the integral action is enabled again.

The integral action is also modified by the following measures:

- The integral action of the controller is deactivated by  $TI = 0.0$
- Weakening of the proportional action when setpoint changes occur
- Control zone
- The output value limits can be modified online

### Weakening of the proportional action when setpoint changes occur (PFAC\_SP)

To prevent overshoot, you can weaken the proportional action using the parameter "Proportional factor for setpoint changes" (PFAC\_SP). Using PFAC\_SP, you can select continuously between 0.0 and 1.0 to decide the effect of the proportional action when the setpoint changes:

- $PFAC\_SP = 1.0$ : Proportional action has full effect if the setpoint changes
- $PFAC\_SP = 0.0$ : Proportional action has no effect if the setpoint changes

The weakening of the proportional action is achieved by compensating the integral action.

### Derivative action (TD, D\_F)

- The derivative action of the controller is deactivated by  $TD = 0.0$
- If the derivative action is active, the following relationship should apply:

$$TD = 0.5 * CYCLE * D\_F$$

### Parameter Settings of a P or PD Controller with Operating Point

In the user interface, deactivate the integral action ( $TI = 0.0$ ) and possibly also the derivative action ( $TD = 0.0$ ). Then make the following parameter settings:

- $I\_ITL\_ON = TRUE$
- $I\_ITLVAL =$  operating point;

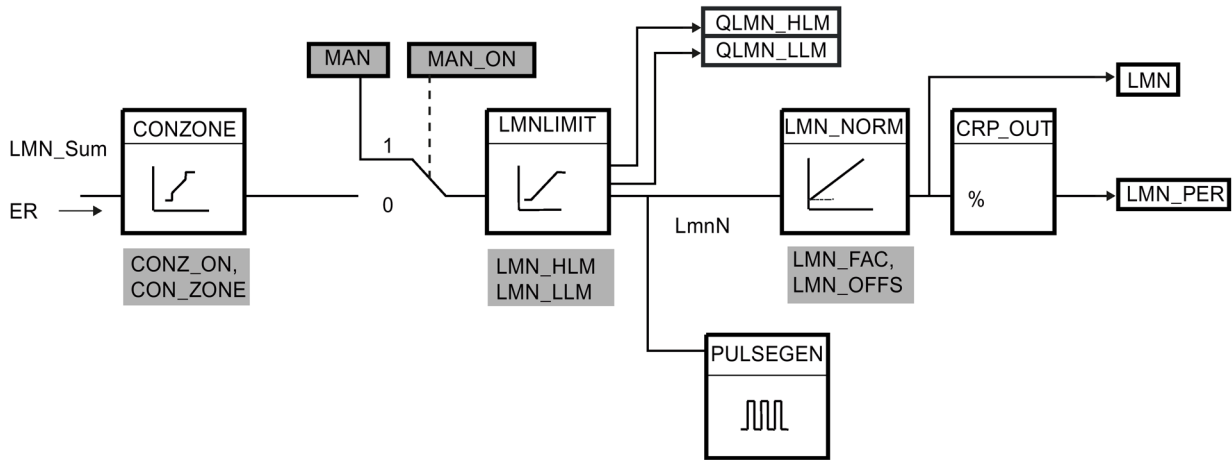
### Feedforward control (DISV)

A disturbance variable can be added at the DISV input.



### Calculating the output value

The diagram below is the block diagram of the output value calculation:



- Parameter configuration interface
- Instruction call interface
- Parameter configuration interface, call interface

### Control zone (CONZ\_ON, CON\_ZONE)

If CONZ\_ON = TRUE, the controller operates with a control zone. This means that the controller operates according to the following algorithm:

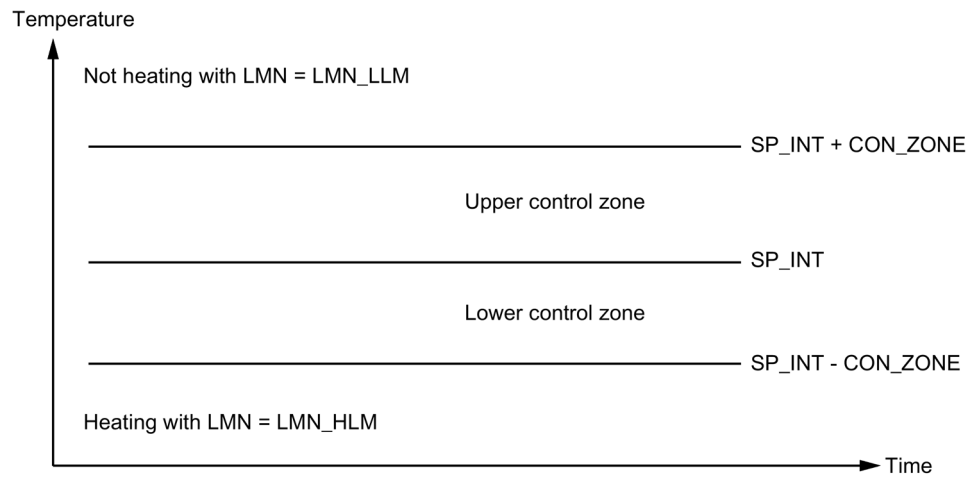
- If process value PV exceeds the setpoint SP\_INT by more than CON\_ZONE, the value LMN\_LLM is output as the manipulated variable.
- If the process value PV falls below setpoint SP\_INT by more than CON\_ZONE, LMN\_HLM is output.
- If the process value PV is within the control zone (CON\_ZONE), the output value takes its value from the PID algorithm LMN\_Sum.

---

#### Note

Changing the manipulated variable from LMN\_LLM or LMN\_HLM to LMN\_Sum occurs under compliance of a hysteresis of 20% of the control zone.

---



---

#### Note

Before enabling the control zone manually, make sure that the control zone band is not too narrow. If the control zone band is too small, oscillations will occur in the manipulated variable and process value.

---

### Advantage of the Control Zone

When the process value enters the control zone, the D-action causes an extremely fast reduction of the manipulated variable. This means that the control zone is only useful when the D-action is activated. Without a control zone, only the reducing P-action would essentially reduce the manipulated variable. The control zone leads to faster settling without overshoot or undershoot if the output minimum or maximum manipulated variable is a long way from the manipulated variable required for the new operating point.

### Manual value processing (MAN\_ON, MAN)

You can change over between manual and automatic mode. In manual mode, the manipulated variable is corrected to a manually selected value.

The integral action (INT) is set internally to  $LMN - LMN\_P - DISV$  and the derivative action (DIF) is set to 0 and synchronized internally. Changeover to automatic mode is therefore bumpless.

---

#### Note

The MAN\_ON parameter has no effect during tuning.

---

### Output value limit LMNLIMIT (LMN\_HLM, LMN\_LLM)

The LMNLIMIT function is used to limit the output value to the limits LMN\_HLM and LMN\_LLM. If these limits are reached, this is indicated by the message bits QLMN\_HLM and QLMN\_LLM.

If the output value is limited, the integral action is halted. If the control deviation moves the integral action back in the direction of the output range, the integral action is enabled again.

### Changing the Manipulated Value Limits Online

If the range of the output value is reduced and the new unlimited value of the output value is outside the limits, the integral action and therefore the output value shifts.

The output value is reduced by the same amount as the output value limit changed. If the output value was unlimited prior to the change, it is set exactly to the new limit (described here for the high output value limit).

### Scaling of output value LMN\_NORM (LMN\_FAC, LMN\_OFFS)

The LMN\_NORM function normalizes the output value according to the following rule:

$$LMN = LmnN * LMN\_FAC + LMN\_OFFS$$

It can be used for the following purposes:

- Output value scaling with LMN\_FAC as output value factor and LMN\_OFFS as output value offset.

The output value is also available in I/O format. The CRP\_OUT function converts the LMN floating-point value to an I/O value according to the following rule:

$$LMN\_PER = LMN * 27648/100$$

The scaling is switched off with the default values (LMN\_FAC = 1.0 and LMN\_OFFS = 0.0). The effective output value is sent to output LMN.

### Save controller parameters SAVE\_PAR

If you classify the current controller parameters as utilizable, you can save these before a manual change in structure parameters provided specifically for this in the instance DB of the instruction TCONT\_CP. If you optimize the controller, the saved parameters are overwritten by the values that were valid prior to tuning.

PFAC\_SP, GAIN, TI, TD, D\_F, CONZ\_ON and CONZONE are written to the structure PAR\_SAVE.

### Reloading Saved Controller Parameters UNDO\_PAR

The last controller parameter settings you saved can be activated for the controller again using this function (in manual mode only).

### Change between PI and PID parameters LOAD\_PID (PID\_ON)

Following tuning, the PI and PID parameters are stored in the PI\_CON and PID\_CON structures. Depending on PID\_ON, you can use LOAD\_PID in manual mode to write the PI or PID parameters to the effective controller parameters.

PID parameters PID_ON = TRUE	PI parameters PID_ON = FALSE
<ul style="list-style-type: none"><li>• GAIN = PID_CON.GAIN</li><li>• TI = PID_CON.TI</li><li>• TD = PID_CON.TD</li></ul>	<ul style="list-style-type: none"><li>• GAIN = PI_CON.GAIN</li><li>• TI = PI_CON.TI</li></ul>

---

#### Note

The controller parameters are only written back to the controller with UNDO\_PAR or LOAD\_PID, if the controller gain is not equal to 0:

With LOAD\_PID, the parameters are only copied if the corresponding GAIN <> 0 is (either the PI or PID parameters). This strategy takes into account the situation that no tuning has yet been made or that PID parameters are missing. If PID\_ON = TRUE and PID.GAIN = FALSE, PID\_ON is set to FALSE and the PI parameter is copied.

- D\_F, PFAC\_SP are preset by the the tuning. These can then be modified by the user. LOAD\_PID does not change these parameters.
  - With LOAD\_PID, the control zone is always recalculated (CON\_ZONE = 250/GAIN), even if CONZ\_ON = FALSE.
- 

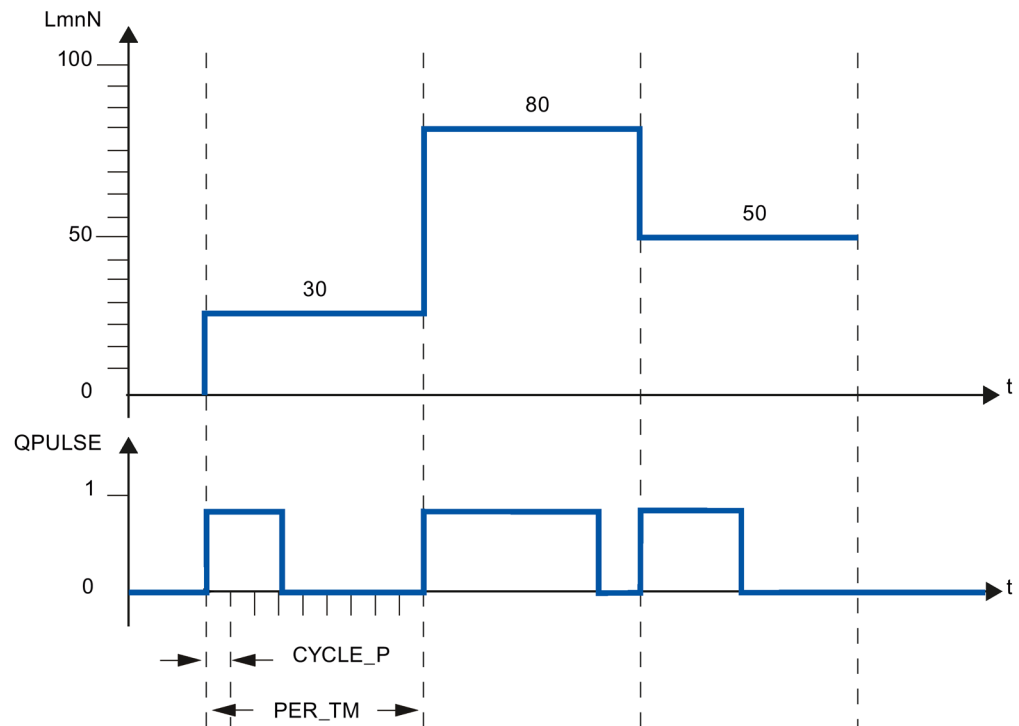
### See also

Operating principle of the pulse generator (Page 493)

Block diagram TCONT\_CP (Page 496)

### 8.4.4.3 Operating principle of the pulse generator

The function PULSEGEN transforms the analog manipulated value LmnN through pulse width module into an impulse sequence with the period duration PER\_TM. PULSEGEN is switched on with PULSE\_ON = TRUE and is processed in the cycle CYCLE\_P.



A manipulated value of LmnN = 30% and 10 PULSEGEN calls per PER\_TM therefore means:

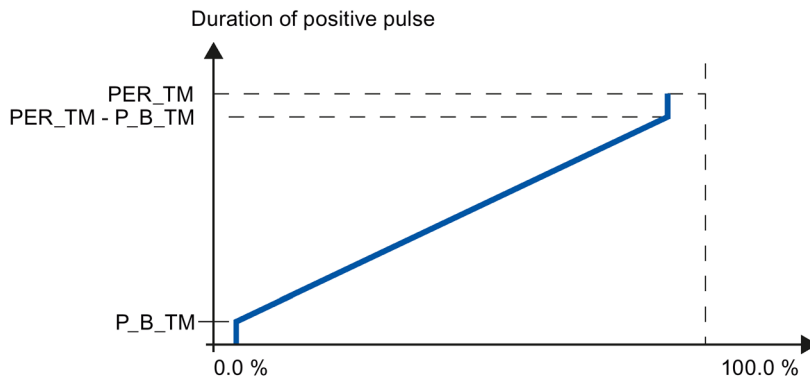
- TRUE at output QPULSE for the first three PULSEGEN calls (30% of 10 calls)
- FALSE at output QPULSE for seven further PULSEGEN calls (70% of 10 calls)

The duration of a pulse per pulse repetition period is proportional to the manipulated variable and is calculated as follows:

$$\text{Pulse duration} = \text{PER\_TM} * \text{LmnN} / 100$$

By suppressing the minimum pulse or break time, the characteristic curve of the conversion develops "knees" in the start and end regions.

The following diagram illustrates two-step control with a unipolar manipulated variable range (0% to 100%):



**Minimum pulse or minimum break time (P\_B\_TM)**

Short on or off times hinder the lifespan of actuators and fine controlling units. These can be avoided by setting a minimum pulse duration or minimum break time P\_B\_TM.

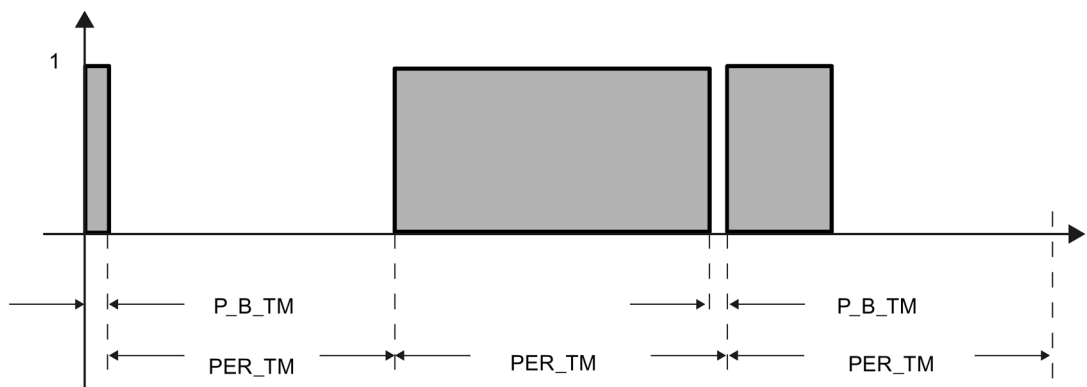
Small absolute values at the input variable LmnN that could otherwise generate a pulse duration shorter than P\_B\_TM are suppressed.

Large input values that would generate a pulse duration greater than PER\_TM - P\_B\_TM are set to 100%. This reduces the dynamics of pulse generation.

Set values of  $P\_B\_TM \leq 0,1 * PER\_TM$  are recommended for the minimum pulse duration and the minimum break duration.

The "knees" in the curves in the diagram above are caused by the minimum pulse or minimum break times.

The following schematic illustrates the switching response of the pulse output:



### Accuracy of pulse generation

The smaller the pulse generator CYCLE\_P is compared to the period duration PER\_TM, the more precise the pulse width modulation is. To achieve sufficiently accurate control, the following relationship should apply:

$$\text{CYCLE\_P} \leq \text{PER\_TM}/50$$

The manipulated value is transformed with a resolution of  $\leq 2\%$  into an impulse.

---

#### Note

When calling the controller in the pulse shaper cycle, you must note the following:

Calling the controller in the pulse shaper cycle will cause the process value to be averaged. As a result, at output PV, different values may be at input PV\_IN and PV\_PER. If you want to track the setpoint value, you must save the process value at input parameter PV\_IN at the call times for complete controller processing (QC\_ACT = TRUE). For pulse shaper calls occurring between these times, you must supply the input parameters PV\_IN and SP\_INT with the saved process value.

---

### See also

Description TCONT\_CP (Page 483)

Mode of operation TCONT\_CP (Page 484)

Block diagram TCONT\_CP (Page 496)

Input parameters TCONT\_CP (Page 498)

Output parameters TCONT\_CP (Page 499)

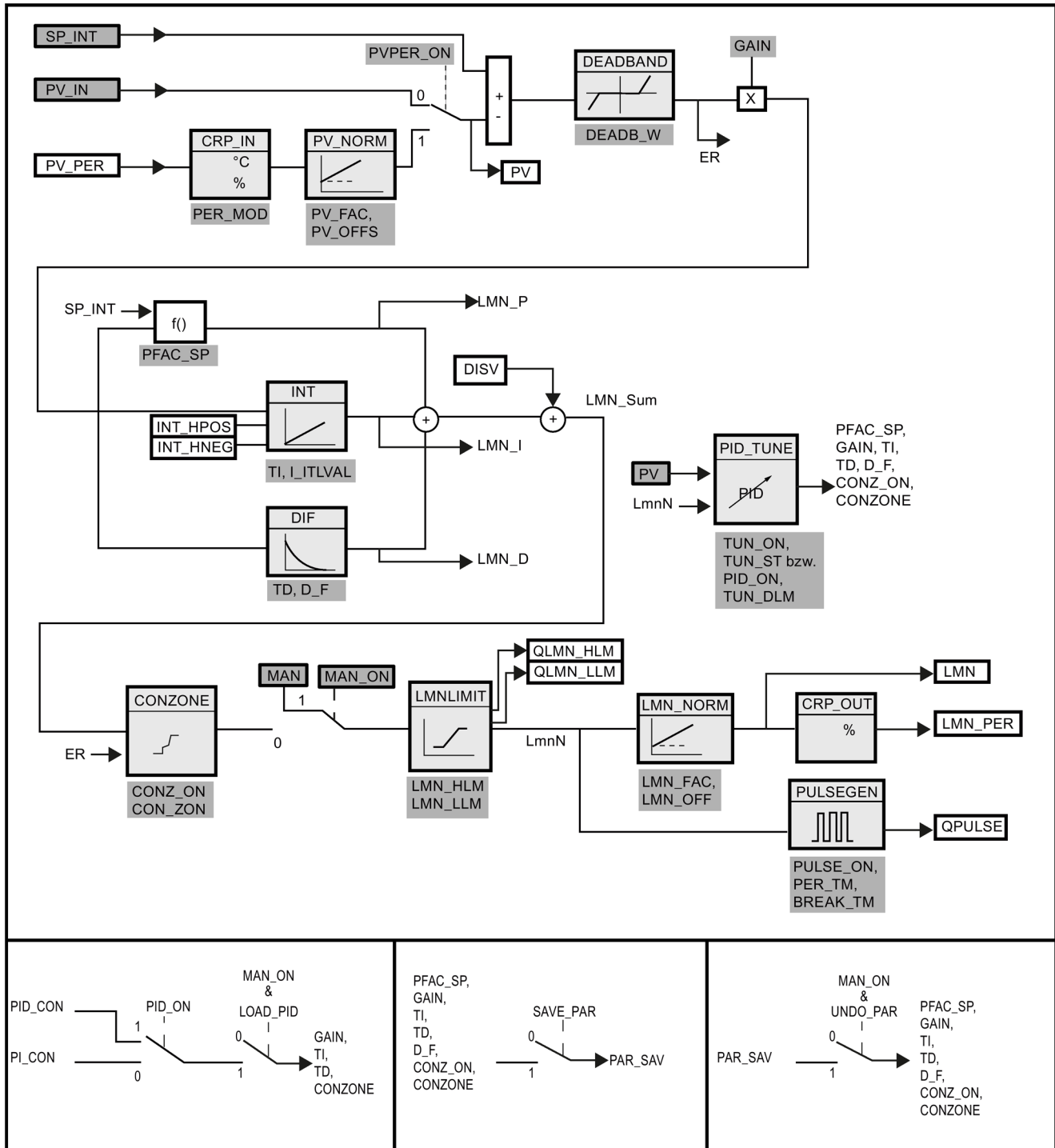
In/out parameters TCONT\_CP (Page 500)

Static variables TCONT\_CP (Page 501)

Parameter STATUS\_H (Page 506)

Parameters STATUS\_D (Page 507)

8.4.4.4 Block diagram TCONT\_CP





**See also**

Description TCONT\_CP (Page 483)  
Mode of operation TCONT\_CP (Page 484)  
Operating principle of the pulse generator (Page 493)  
Input parameters TCONT\_CP (Page 498)  
Output parameters TCONT\_CP (Page 499)  
In/out parameters TCONT\_CP (Page 500)  
Static variables TCONT\_CP (Page 501)  
Parameter STATUS\_H (Page 506)  
Parameters STATUS\_D (Page 507)

8.4.4.5 Input parameters TCONT\_CP

Table 8- 19

Parameters	Ad- dress	Data type	Default	Description
PV_IN	0.0	REAL	0.0	At the "Process value input" you can assign parameters to a commissioning value or you can interconnect an external process value in floating-point format. The valid values depend on the sensors used.
PV_PER	4.0	INT	0	The process value in I/O format is interconnected with the controller at the "Process value I/O" input.
DISV	6.0	REAL	0.0	For feedforward control, the disturbance variable is interconnected to the "Disturbance variable" input.
INT_HPOS	10.0	BOOL	FALSE	The output of the integral action can be held in the positive direction. For this, the input INT_HPOS must be set to TRUE. In a cascade control, INT_HPOS of the primary controller is connected to QLMN_HLM of the secondary controller.
INT_HNEG	10.1	BOOL	FALSE	The output of the integral action can be held in the negative direction. For this, the input INT_HNEG must be set to TRUE. In a cascade control, INT_HNEG of the primary controller is connected to QLMN_LLM of the secondary controller.
SELECT	12.0	INT	0	If the pulse shaper is on, there are several ways of calling the PID algorithm and pulse shaper: <ul style="list-style-type: none"> <li>• SELECT = 0: The controller is called in a fast cyclic interrupt priority class and the PID algorithm and pulse shaper are processed.</li> <li>• SELECT = 1: The controller is called in OB1 and only the PID algorithm is processed.</li> <li>• SELECT = 2: The controller is called in a fast cyclic interrupt priority class and only the pulse shaper is processed.</li> <li>• SELECT = 3: The controller is called a slow cyclic interrupt priority class and only the PID algorithm is processed.</li> </ul>

See also

Operating principle of the pulse generator (Page 493)

Block diagram TCONT\_CP (Page 496)

### 8.4.4.6 Output parameters TCONT\_CP

Table 8- 20

Parameter	Ad- dress	Data type	Default	Description
PV	14.0	REAL	0.0	The effective process value is output at the "Process value" output. The valid values depend on the sensors used.
LMN	18.0	REAL	0.0	The effective "Manipulated value" is output in floating point format at the "Ma- nipulated value" output.
LMN_PER	22.0	INT	0	The manipulated value in I/O format is interconnected with the controller on the output "Manipulated value I/O".
QPULSE	24.0	BOOL	FALSE	The manipulated value is pulse-width-modulated at the QPULSE output.
QLMN_HLM	24.1	BOOL	FALSE	The manipulated value is always restricted to a high limit and low limit. The output QLMN_HLM signals that the high limit has been reached.
QLMN_LLM	24.2	BOOL	FALSE	The manipulated value is always restricted to a high limit and low limit. The output QLMN_LLM signals that the low limit has been reached.
QC_ACT	24.3	BOOL	TRUE	This parameter indicates whether continuous control component will be pro- cessed the next time the block is called (relevant only when SELECT has the value 0 or 1).

#### See also

Operating principle of the pulse generator (Page 493)

Block diagram TCONT\_CP (Page 496)

Parameter STATUS\_H (Page 506)

Parameters STATUS\_D (Page 507)

8.4.4.7 In/out parameters TCONT\_CP

Table 8- 21

Parameters	Ad- dress	Data type	Default	Description
CYCLE	26.0	REAL	0.1 s	Sets the sampling time for the PID algorithm. In phase 1, the tuner calculates the sampling time and enters it in CYCLE. CYCLE > 0.001 s
CYCLE_P	30.0	REAL	0.02 s	At this input, you set the sampling time for the pulse shaper action. In phase 1, the TCONT_CP instruction calculates the sampling time and enters it in CYCLE_P. CYCLE_P > 0.001 s
SP_INT	34.0	REAL	0.0	The input "Internal setpoint" is used to specify a setpoint. The valid values depend on the sensors used.
MAN	38.0	REAL	0.0	The "Manual value" input is used to set a manual value. In automatic mode, it tracks the manipulated value.
COM_RST	42.0	BOOL	FALSE	The block has an initialization routine that is processed when the COM_RST input is set.
MAN_ON	42.1	BOOL	TRUE	If the input "Enable manual mode" is set then the control loop is interrupted. The manual value MAN is set as manipulated value.

See also

Operating principle of the pulse generator (Page 493)

Block diagram TCONT\_CP (Page 496)

## 8.4.4.8 Static variables TCONT\_CP

Table 8- 22

Parameters	Address	Data type	Default	Description
DEADB_W	44.0	REAL	0.0	A deadband is applied to the control deviation. The "Deadband width" input determines the size of the deadband. The valid values depend on the sensors used.
I_ITLVAL	48.0	REAL	0.0	The output of the integrator can be set at the I_ITL_ON input. The initialization value is applied to the "Initialization value of the I-action" input. During a restart COM_RST = TRUE, the I-action is set to the initialization value. Values from -100 to 100 % are permitted.
LMN_HLM	52.0	REAL	100.0	The output value is always restricted to a high limit and low limit. The "Manipulated value high limit" input specifies the high limit. LMN_HLM > LMN_LLM
LMN_LLM	56.0	REAL	0.0	The output value is always restricted to a high limit and low limit. The "Manipulated value low limit" input specifies the low limit. LMN_LLM < LMN_HLM
PV_FAC	60.0	REAL	1.0	The "Process value factor" input is multiplied by the "Process value I/O". The input is used to scale the process value range.
PV_OFFS	64.0	REAL	0.0	The "Process value offset" input is added to the "Process value I/O". The input is used to scale the process value range.
LMN_FAC	68.0	REAL	1.0	The "Output value factor" input is multiplied with the output value. The input is used to scale the output value range.
LMN_OFFS	72.0	REAL	0.0	The "Output value offset" input is added to the output value. The input is used to scale the output value range.
PER_TM	76.0	REAL	1.0 s	The period duration of the pulse width modulation is entered at the PER_TM parameter. The relationship of the period duration to the sampling time of the pulse shaper determines the accuracy of the pulse width modulation. PER_TM ≥ CYCLE
P_B_TM	80.0	REAL	0.02 s	You can assign a minimum pulse or break time at the parameter "Minimum pulse/break time". P_B_TM is internally limited to > CYCLE_P.
TUN_DLMN	84.0	REAL	20.0	Process excitation for controller tuning results from a output value step change at TUN_DLMN. Values from -100 to 100 % are permitted.

8.4 PID basic functions

Parameters	Address	Data type	Default	Description
PER_MODE	88.0	INT	0	<p>You can use this switch to enter the type of I/O module. The process value at input PV_PER is then scaled as follows at the PV output.</p> <ul style="list-style-type: none"> <li>PER_MODE = 0: Thermoelements; PT100/NI100; standard PV_PER * 0.1 Unit: °C, °F</li> <li>PER_MODE = 1: PT100/NI100; climate PV_PER * 0.01 Unit: °C, °F</li> <li>PER_MODE = 2: Current/voltage PV_PER * 100/27648 Unit: %</li> </ul>
PVPER_ON	90.0	BOOL	FALSE	If the process value is to be read in from the I/Os, the PV_PER input must be interconnected with the I/Os and the "Enable process value I/Os" input must be set.
I_ITL_ON	90.1	BOOL	FALSE	The output of the integrator can be set at the I_ITLVAL input. The "Set I-action" input must be set for this.
PULSE_ON	90.2	BOOL	FALSE	If PULSE_ON = TRUE is set, the pulse shaper is activated.
TUN_KEEP	90.3	BOOL	FALSE	The mode changes to automatic only when TUN_KEEP changes to FALSE.
ER	92.0	REAL	0.0	The effective control deviation is output at the "Control deviation" output. The valid values depend on the sensors used.
LMN_P	96.0	REAL	0.0	The "P-action" output contains the proportional action of the manipulated tag.
LMN_I	100.0	REAL	0.0	The "integral action" output contains the integral action of the manipulated tag.
LMN_D	104.0	REAL	0.0	The "D-action" output contains the derivative action of the manipulated tag.
PHASE	108.0	INT	0	<p>The current phase of controller tuning is indicated at the PHASE output.</p> <ul style="list-style-type: none"> <li>PHASE = 0: No tuning mode; automatic or manual mode</li> <li>PHASE = 1: Ready to start tuning; check parameters, wait for excitation, measure the sampling times</li> <li>PHASE = 2: Actual tuning: Searching for point of inflection with constant output value. Entering the sampling time in instance DB.</li> <li>PHASE = 3: Calculating process parameters. Saving valid controller parameters prior to tuning.</li> <li>PHASE = 4: Controller design</li> <li>PHASE = 5: Following up the controller to the new manipulated tag</li> <li>PHASE = 7: Validating the process type</li> </ul>
STATUS_H	110.0	INT	0	STATUS_H indicates the diagnostic value via the search for the point of inflection during the heating process.
STATUS_D	112.0	INT	0	STATUS_D indicates the diagnostic value via the controller design during the heating process.

Parameters	Address	Data type	Default	Description
QTUN_RUN	114.0	BOOL	0	The tuning manipulated tag has been applied, tuning has started and is still in phase 2 (searching for point of inflection).
PI_CON	116.0	STRUCT		PI controller parameters
GAIN	+0.0	REAL	0.0	PI controller gain %/phys. unit
TI	+4.0	REAL	0.0 s	PI integration time [s]
PID_CON	124.0	STRUCT		PID controller parameters
GAIN	+0.0	REAL	0.0	PID controller gain
TI	+4.0	REAL	0.0s	PID integration time [s]
TD	+8.0	REAL	0.0s	PID derivative action time [s]
PAR_SAVE	136.0	STRUCT		The PID parameters are saved in this structure.
PFAC_SP	+0.0	REAL	1.0	Proportional factor for setpoint changes Values from 0.0 to 1.0 are permitted.
GAIN	+4.0	REAL	0.0	Controller gain %/phys. unit
TI	+8.0	REAL	40.0 s	Integration time [s]
TD	+12.0	REAL	10.0 s	Derivative action time (s)
D_F	+16.0	REAL	5.0	Derivative factor Values from 5.0 to 10.0 are permitted.
CON_ZONE	+20.0	REAL	100.0	Control zone band If the control deviation is greater than the control zone band, the high output value limit is output as output value. If the control deviation is less than the negative control zone band, the low output value limit is output as the output value. CON_ZONE ≥ 0.0
CONZ_ON	+24.0	BOOL	FALSE	Enable control zone
PFAC_SP	162.0	REAL	1.0	PFAC_SP specifies the effective P-action when there is a setpoint change. This is set between 0 and 1. <ul style="list-style-type: none"> <li>1: P-action has full effect if the setpoint changes.</li> <li>0: P-action has no effect if the setpoint changes.</li> </ul> Values from 0.0 to 1.0 are permitted.
GAIN	166.0	REAL	2.0	The "Proportional gain" input specifies controller amplification. The direction of control can be reversed by giving GAIN a negative sign. %/phys. unit
TI	170.0	REAL	40.0 s	The "Integration time" (integral-action time) input defines the integrator's time response.
TD	174.0	REAL	10.0 s	The "Derivative-action time" (rate time) input decides the time response of the differentiator.
D_F	178.0	REAL	5.0	The derivative factor decides the lag of the D-action. D_F = derivative-action time/"Lag of the D-action" Values from 5.0 to 10.0 are permitted.

8.4 PID basic functions

Parameters	Address	Data type	Default	Description
CON_ZONE	182.0	REAL	100.0	If the control deviation is greater than the control zone band, the high output value limit is output as output value. If the control deviation is less than the negative control zone band, the low output value limit is output as the output value. The valid values depend on the sensors used.
CONZ_ON	186.0	BOOL	FALSE	You can use CONZ_ON =TRUE to enable the control zone.
TUN_ON	186.1	BOOL	FALSE	If TUN_ON=TRUE, the output value is averaged until the output value excitation TUN_DLMN is enabled either by a setpoint step-change or by TUN_ST=TRUE.
TUN_ST	186.2	BOOL	FALSE	If the setpoint is to remain constant during controller tuning at the operating point, a output value step-change by the amount of TUN_DLMN is activated by TUN_ST=1.
UNDO_PAR	186.3	BOOL	FALSE	Loads the controller parametersPFAC_SP, GAIN, TI, TD, D_FCONZ_ON and CON_ZONE from the data structure PAR_SAVE (only in manual mode).
SAVE_PAR	186.4	BOOL	FALSE	Saves the controller parameters PFAC_SP, GAIN, TI, TD, D_F, CONZ_ON and CON_ZONE in the data structure PAR_SAVE.
LOAD_PID	186.5	BOOL	FALSE	Loads the controller parametersGAIN, TI,TD depending on PID_ON from the data structure PI_CON or PID_CON (only in manual mode)
PID_ON	186.6	BOOL	TRUE	At the PID_ON input, you can specify whether or not the tuned controller will operate as a PI or PID controller. <ul style="list-style-type: none"> <li>• PID controller: PID_ON = TRUE</li> <li>• PI controller: PID_ON = FALSE</li> </ul> With certain process types it is nevertheless possible that only a PI controller will be designed despite PID_ON = TRUE.
GAIN_P	188.0	REAL	0.0	Identified process gain. In the case of process type I, GAIN_P tends to be estimated too low.
TU	192.0	REAL	0.0	Identified time lag of the process. $TU \geq 3 * CYCLE$
TA	196.0	REAL	0.0	Identified recovery time of the process. In the case of process type I, TA tends to be estimated too low.
KIG	200.0	REAL	0.0	Maximum process value rise at manipulated tag excitation from 0 to 100 % [1/s] $GAIN_P = 0.01 * KIG * TA$
N_PTN	204.0	REAL	0.0	The parameter specifies the order of the process. "Non-integer values" are also possible. Values from 1.01 to 10.0 are permitted.
TM_LAG_P	208.0	REAL	0.0	Time constants of a PTN model (practical values only for N_PTN >= 2).
T_P_INF	212.0	REAL	0.0	Time from process excitation until the point of inflection.
P_INF	216.0	REAL	0.0	Process value change from process excitation until the point of inflection. The valid values depend on the sensors used.
LMN0	220.0	REAL	0.0	Output value at the start of tuning Detected in phase 1 (mean value). Values from 0 to 100 % are permitted.
PV0	224.0	REAL	0.0	Process value at the start of tuning



Parameters	Address	Data type	Default	Description
PVDT0	228.0	REAL	0.0	Process value slew rate at start of tuning [1/s] Sign adapted.
PVDT	232.0	REAL	0.0	Current process value slew rate [1/s] Sign adapted.
PVDT_MAX	236.0	REAL	0.0	Max. change in the process value per second [1/s] Maximum derivative of the process value at the point of inflection (sign adapted, always > 0); is used to calculate TU and KIG.
NOI_PVDT	240.0	REAL	0.0	Noise action in PVDT_MAX in % The higher the noise action, the less accurate (less aggressive) the control parameters.
NOISE_PV	244.0	REAL	0.0	Absolute noise in process value Difference between maximum and minimum process value in phase 1.
FIL_CYC	248.0	INT	1	Number of cycles of the mean value filter The process value is determined through FIL_CYC cycles. FIL_CYC is increased from 1 to a max. of 1024 if needed.
POI_CMAX	250.0	INT	2	Maximum number of cycles after point of inflection This time is used to find another (i.e. better) inflection point for measuring noise. The tuning is completed only after this time.
POI_CYCL	252.0	INT	0	Number of cycles after inflection point

### See also

Operating principle of the pulse generator (Page 493)

Block diagram TCONT\_CP (Page 496)

8.4.4.9 Parameter STATUS\_H

STATUS_H	Description	Remedy
0	Default, or no/no new controller parameters	
10000	Tuning completed + suitable controller parameters found	
2xxxx	Tuning completed + controller parameters uncertain	
2xx2x	Point of inflection not reached (only if excited via setpoint step-change)	If the controller oscillates, weaken the controller parameters, or repeat the test with a smaller manipulated value difference TUN_DLMN.
2x1xx	Estimation error (TU < 3*CYCLE)	Reduce CYCLE and repeat attempt. Special case for PT1-only process: Do not repeat test, if necessary reduce controller parameters.
2x3xx	Estimation error TU too high	Repeat test under better conditions.
21xxx	Estimation error N_PTN < 1	Repeat test under better conditions.
22xxx	Estimation error N_PTN > 10	Repeat test under better conditions.
3xxxx	Tuning canceled in phase 1 owing to faulty parameter assignment:	
30002	Effective manipulated value differential < 5%	Correct manipulated value differential TUN_DLMN.
30005	The sampling times CYCLE and CYCLE_P differ by more than 5% of the measured values.	Compare CYCLE and CYCLE_P with the cycle time of the cyclic interrupt priority class and note any loop scheduler.  Check CPU load. An excessively loaded CPU can result in prolonged sampling times that are inconsistent with CYCLE or CYCLE_P.

**Note**

If you cancel tuning in phase 1 or 2, STATUS\_H = 0 is set. However, STATUS\_D still displays the status of the last controller calculation.

The higher the value of STATUS\_D, the higher the order of the control process, the greater the TU/TA ratio and the gentler the controller parameters will be.

**See also**

Operating principle of the pulse generator (Page 493)

Block diagram TCONT\_CP (Page 496)

#### 8.4.4.10 Parameters STATUS\_D

STATUS_D	Description
0	No controller parameters were calculated.
110	$N_{PTN} \leq 1.5$ Process type I fast
121	$N_{PTN} > 1.5$ Process type I
200	$N_{PTN} > 1.9$ Process type II (transition range)
310	$N_{PTN} \geq 2.1$ Process type III fast
320	$N_{PTN} > 2.6$ Process type III
111, 122, 201, 311, 321	Parameters have been corrected from phase 7.

---

**Note**

The higher the value of STATUS\_D, the higher the order of the control process, the greater the TU/TA ratio and the gentler the controller parameters will be.

---

**See also**

Operating principle of the pulse generator (Page 493)

Block diagram TCONT\_CP (Page 496)

## 8.4.5 TCONT\_S

### 8.4.5.1 Description TCONT\_S

The TCONT\_S instruction is used on SIMATIC S7 automation systems to control technical temperature processes with binary manipulated value output signals for actuators with integrating behavior. The functionality is based on the PI control algorithm of the sampling controller. The step controller operates without a position feedback signal.

#### Application

You can also use the controller in a cascade control as a secondary position controller. You specify the actuator position via the setpoint input SP\_INT. In this case, you must set the process value input and the parameter TI (integration time) to zero. An application might be, for example, temperature control with heating power control using pulse-break activation and cooling control using a butterfly valve. To close the valve completely, the manipulated variable (ER\*GAIN) should be negative.

#### Call

The instruction TCONT\_S must be called equidistant. To achieve this, use a cyclic interrupt priority class (for example, OB35 for an S7-300). The sampling time is specified at the CYCLE parameter.

If you call the instruction TCONT\_S as a multiple instance DB, no technology object is created. No parameter assignment interface or commissioning interface is available. You must assign parameters for TCONT\_S directly in the multiple instance DB and commission it via a watch table.

#### CYCLE sampling time

The CYCLE sampling time match the time difference between two calls (cycle time of the cyclic interrupt OB taking into account the reduction ratios).

The controller sampling time should not exceed 10% of the calculated integration time of the controller (TI). Generally, you must set the sampling time to a much lower value to achieve the required accuracy of the step controller.

Required accuracy G	MTR_TM	CYCLE = MTR_TM*G	Comment
0.5 %	10 s	0.05 s	The sampling time is determined by the required accuracy of the step controller.

#### Start-up

The TCONT\_S instruction has an initialization routine that is run through when input parameter COM\_RST = TRUE is set. Following execution of the initialization routine, the block sets COM\_RST back to FALSE. All outputs are set to their initial values. If you require initialization when the CPU restarts, call the block in OB100 with COM\_RST = TRUE.

#### See also

Block diagram TCONT\_S (Page 513)

### 8.4.5.2 Mode of operation TCONT\_S

#### Setpoint branch

The setpoint is entered at input SP\_INT in floating-point format as a physical value or percentage. The setpoint and process value used to form the control deviation must have the same unit.

#### Process value options (PVPER\_ON)

Depending on PVPER\_ON, the process value can be read in, in the I/O or floating-point format.

PVPER_ON	Process Value Input
TRUE	The process value is read in via the analog I/Os (PIW xxx) at input PV_PER.
FALSE	The process value is acquired in floating-point format at input PV_IN.

#### Process value format conversion CRP\_IN (PER\_MODE)

The CRP\_IN function converts the I/O value PV\_PER to floating-point format depending on the PER\_MODE switch according to the following rules:

PER_MODE	Output of CRP_IN	Analog Input Type	Unit
0	$PV\_PER * 0.1$	Thermoelements; PT100/Ni100; standard	°C;°F
1	$PV\_PER * 0.01$	PT100/Ni100; climate;	°C;°F
2	$PV\_PER * 100/27648$	Voltage/current	%

#### Process value scaling PV\_NORM (PF\_FAC, PV\_OFFS)

The PV\_NORM function calculates the output of CRP\_IN according to the following rule:

$$\text{"Output of PV\_NORM"} = \text{"Output of CRP\_IN"} * PV\_FAC + PV\_OFFS$$

It can be used for the following purposes:

- Process value adjustment with PV\_FAC as process value factor and PV\_OFFS as process value offset.
- Normalization of temperature to percentage

You want to enter the setpoint as a percentage and must now convert the measured temperature value to a percentage.

- Normalization of percentage to temperature

You want to enter the setpoint in the physical temperature unit and must now convert the measured voltage/current value to a temperature.

Calculation of the parameters:

- $PV\_FAC = \text{range of } PV\_NORM / \text{range of } CRP\_IN;$
- $PV\_OFFS = LL (PV\_NORM) - PV\_FAC * LL(CRP\_IN);$

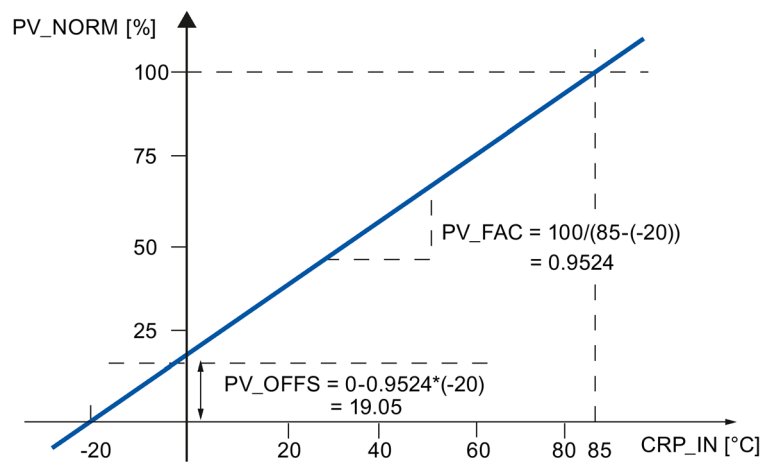
where LL: low limit

The normalization is switched off with the default values ( $PV\_FAC = 1.0$  and  $PV\_OFFS = 0.0$ ). The effective process value is output at the PV output.

### Example of Process Value Normalization

If you want to enter the setpoint as a percentage, and you have a temperature range of -20 to 85 °C applied to ,  $CRP\_IN$  you must normalize the temperature range as a percentage.

The following diagram shows an example of adapting the temperature range -20 to 85 °C to an internal scale of 0 to 100 %:



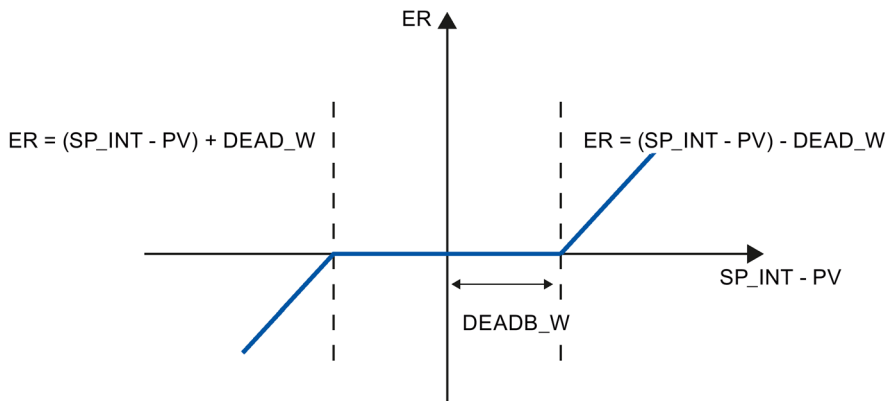
### Forming the control deviation

The difference between the setpoint and process value is the control deviation before the dead band.

The setpoint and process value must exist in the same unit.

## Dead band (DEADB\_W)

To suppress a minor sustained oscillation due to the manipulated variable quantization (for example, in pulse width modulation with PULSEGEN) a dead band is applied to the (DEADBAND) control deviation. With DEADB\_W = 0.0, the dead band is switched off.



## PI step controller algorithm

The instruction TCONT\_S operates without position feedback. The I-action of the PI algorithm and the assumed position feedback signal are calculated in an integrator (INT) and compared as a feedback value with the remaining P-action. The difference is applied to a three-step element (THREE\_ST) and a pulse shaper (PULSEOUT) that generates the pulses for the control valve. Adapting the response threshold of the three-step element reduces the switching frequency of the controller.

## Weakening of the P-action when setpoint changes occur

To prevent overshoot, you can weaken the P-action using the "Proportional factor for setpoint changes" parameter (PFAC\_SP). Using PFAC\_SP, you can now select continuously between 0.0 and 1.0 to decide the effect of the P-action when the setpoint changes:

- PFAC\_SP = 1.0: P-action has full effect if the setpoint changes
- PFAC\_SP = 0.0: P-action has no effect if the setpoint changes

As in the case of the continuous controller, a value of PFAC\_SP < 1.0 can reduce the overshoot if the motor run time MTR\_TM is small compared with the recovery time TA and the ratio is TU/TA < 0.2. If MTR\_TM reaches 20% of TA, only a slight improvement can still be achieved.

## Feedforward control

A disturbance variable can be added at the DISV input.

### **Manual value processing (LMNS\_ON, LMNUP, LMNDN)**

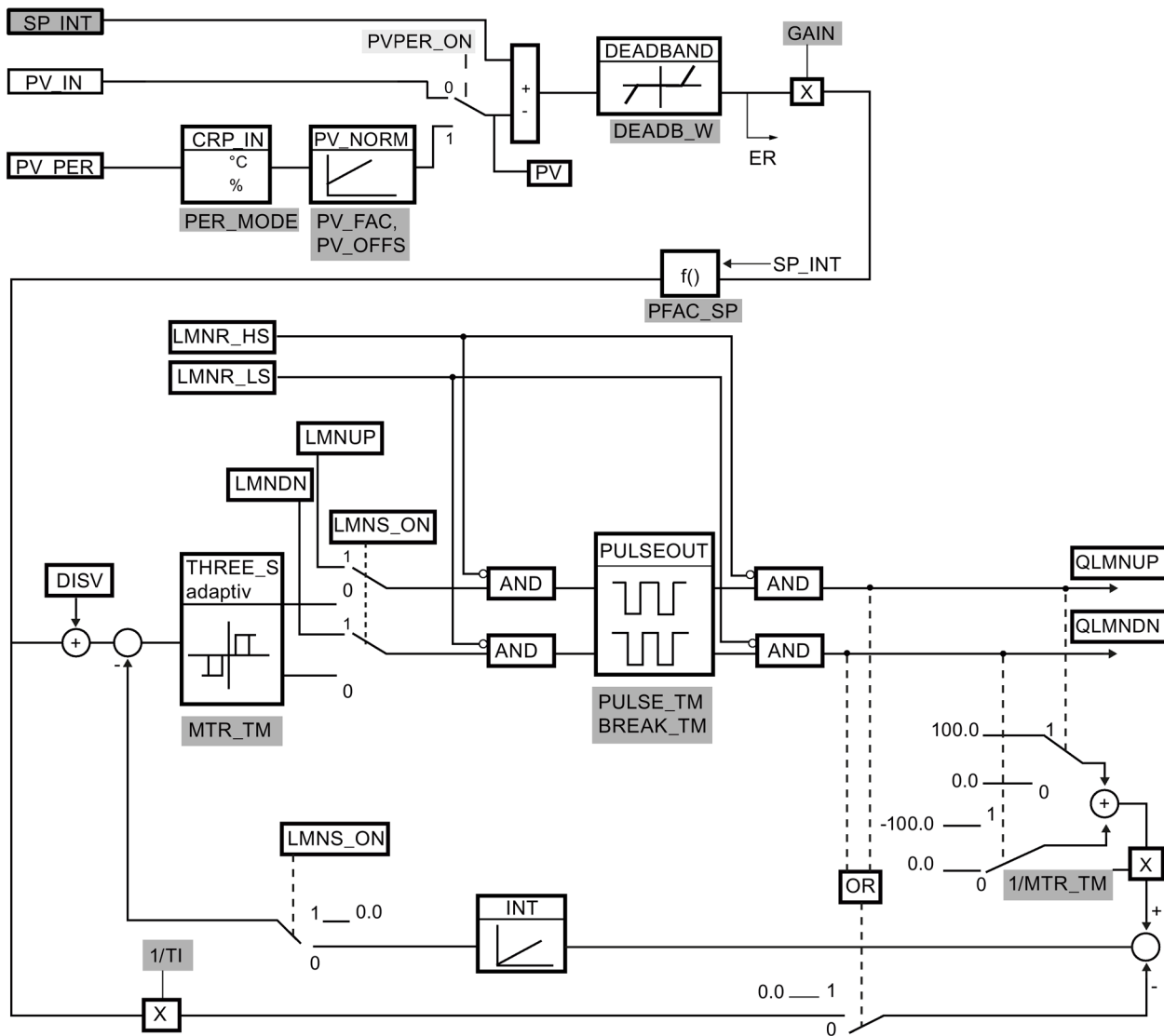
With LMNS\_ON, you can change between manual and automatic mode. In manual mode, the actuator is halted and the integral action (INT) is set to 0 internally. Using LMNUP and LMNDN, the actuator can be adjusted to OPEN and CLOSED. Switching over to automatic mode therefore involves a bump. As a result of the GAIN, the existing control deviation leads to a step change in the internal manipulated variable. The integral component of the actuator, however, results in a ramp-shaped excitation of the process.

### **See also**

Block diagram TCONT\_S (Page 513)



### 8.4.5.3 Block diagram TCONT\_S



- Parameter configuration interface
- Instruction call interface
- Parameter configuration interface, call interface

**See also**

Description TCONT\_S (Page 508)

Mode of operation TCONT\_S (Page 509)

Input parameters TCONT\_S (Page 515)

Output parameters TCONT\_S (Page 516)

In/out parameters TCONT\_S (Page 516)

Static variables TCONT\_S (Page 517)

### 8.4.5.4 Input parameters TCONT\_S

Table 8- 23

Parameters	Ad- dress	Data type	Default	Description
CYCLE	0.0	REAL	0.1 s	At this input, you enter the sampling time for the controller. CYCLE $\geq$ 0.001
SP_INT	4.0	REAL	0.0	The input "Internal setpoint" is used to specify a setpoint. The valid values depend on the sensors used.
PV_IN	8.0	REAL	0.0	At the "Process variable input" you can assign parameters to a commissioning value or you can interconnect an external process value in floating-point format. The valid values depend on the sensors used.
PV_PER	12.0	INT	0	The process value in I/O format is interconnected with the controller at the "Process value I/O" input.
DISV	14.0	REAL	0.0	For feedforward control, the disturbance variable is interconnected to the "Disturbance variable" input.
LMNR_HS	18.0	BOOL	FALSE	The signal "Control valve at high endstop" is interconnected on the input "High endstop signal of position feedback". <ul style="list-style-type: none"> <li>LMNR_HS=TRUE: The control valve is at high endstop.</li> </ul>
LMNR_LS	18.1	BOOL	FALSE	The signal "Control valve at low endstop" is interconnected on the input "Low endstop signal of position feedback". <ul style="list-style-type: none"> <li>LMNR_LS=TRUE: The control valve is at low endstop.</li> </ul>
LMNS_ON	18.2	BOOL	TRUE	Manipulated value signal processing is switched to manual mode at the "Enable manual mode of manipulated signal".
LMNUP	18.3	BOOL	FALSE	In manual mode of manipulated signals, the output parameter QLMNUP is operated at the input parameter "Manipulated signal up".
LMNDN	18.4	BOOL	FALSE	In manual mode of the manipulated signals, the output parameter QLMNDN is operated at the input parameter "Manipulated signal down".

#### See also

Block diagram TCONT\_S (Page 513)

**8.4.5.5 Output parameters TCONT\_S**

Table 8- 24

Parameters	Ad- dress	Data type	Default	Description
QLMNUP	20.0	BOOL	FALSE	If the output "Manipulated value signal up" is set then the control valve should be open.
QLMNDN	20.1	BOOL	FALSE	If the output "Manipulated value signal down" is set then the control valve should be closed.
PV	22.0	REAL	0.0	The effective process value is output at the "Process value" output.
ER	26.0	REAL	0.0	The effective system deviation is output at the "Error signal" output.

**See also**

Block diagram TCONT\_S (Page 513)

**8.4.5.6 In/out parameters TCONT\_S**

Table 8- 25

Parameters	Ad- dress	Data type	Default	Description
COM_RST	30.0	BOOL	FALSE	The block has an initialization routine that is processed when the COM_RST input is set.

**See also**

Block diagram TCONT\_S (Page 513)

### 8.4.5.7 Static variables TCONT\_S

Table 8- 26

Parameters	Ad- dress	Data type	Default	Description
PV_FAC	32.0	REAL	1.0	The "Process value factor" input is multiplied by the process value. The input is used to scale the process value range.
PV_OFFS	36.0	REAL	0.0	The input "Process value offset" is added to the process value. The input is used to scale the process value range. The valid values depend on the sensors used.
DEADB_W	40.0	REAL	0.0	A deadband is applied to the control deviation. The "Deadband width" input determines the size of the deadband. $DEADB\_W \geq 0.0$
PFAC_SP	44.4	REAL	1.0	PFAC_SP specifies the effective P-action when there is a setpoint change. <ul style="list-style-type: none"> <li>1: P-action has full effect if the setpoint changes.</li> <li>0: P-action has no effect if the setpoint changes.</li> </ul> Values from 0.0 to 1.0 are permitted.
GAIN	48.0	REAL	2.0	The "Proportional gain" input specifies controller amplification. The direction of control can be reversed by giving GAIN a negative sign. %/phys. unit
TI	52.0	REAL	40.0 s	The "Integration time" (integral-action time) input defines the integrator's time response.
MTR_TM	56.0	REAL	30 s	The runtime from endstop to endstop of the control valve is entered at the "Motor actuating time" parameter. $MTR\_TM \geq CYCLE$
PULSE_TM	60.0	REAL	0.0 s	A minimum pulse time can be configured at the "Minimum pulse time" parameter.
BREAK_TM	64.0	REAL	0.0 s	You can assign a minimum break time at the parameter "Minimum break time".
PER_MODE	68.0	INT	0	You can use this switch to enter the type of I/O module. The process value at input PV_PER is then scaled as follows at the PV output. <ul style="list-style-type: none"> <li>PER_MODE = 0: Thermoelements; PT100/NI100; standard PV_PER * 0.1 Unit: °C, °F</li> <li>PER_MODE = 1: PT100/NI100; climate PV_PER * 0.01 Unit: °C, °F</li> <li>PER_MODE = 2: Current/voltage PV_PER * 100/27648 Unit: %</li> </ul>
PVPER_ON	70.0	BOOL	FALSE	If the process value is to be read in from the I/Os, the PV_PER input must be interconnected with the I/Os and the "Enable process value I/Os" input must be set.

#### See also

Block diagram TCONT\_S (Page 513)

## 8.4.6 Integrated system functions

### 8.4.6.1 CONT\_C\_SF

#### CONT\_C\_SF

The instruction CONT\_C\_SF is integrated in the S7-300 compact CPUs. The instruction must not be transmitted to the S7-300 CPU during loading. The scope of function corresponds with the instruction CONT\_C.

#### See also

Description CONT\_C (Page 461)

How CONT\_C works (Page 462)

CONT\_C block diagram (Page 464)

Input parameter CONT\_C (Page 465)

Output parameters CONT\_C (Page 467)

### 8.4.6.2 CONT\_S\_SF

#### CONT\_S\_SF

The instruction CONT\_S\_SF is integrated in the S7-300 compact CPUs. The instruction must not be transmitted to the S7-300 CPU during loading. The scope of function corresponds with the instruction CONT\_S.

#### See also

Description CONT\_S (Page 468)

Mode of operation CONT\_S (Page 469)

Block diagram CONT\_S (Page 470)

Input parameters CONT\_S (Page 471)

Output parameters CONT\_S (Page 472)

### 8.4.6.3 PULSEGEN\_SF

#### PULSEGEN\_SF

The instruction PULSEGEN\_SF is integrated in the S7-300 compact CPUs. The instruction must not be transmitted to the S7-300 CPU during loading. The scope of function corresponds with the instruction PULSEGEN.

#### See also

Description PULSEGEN (Page 473)

Mode of operation PULSEGEN (Page 474)

Mode of operation PULSEGEN (Page 477)

Three-step control (Page 477)

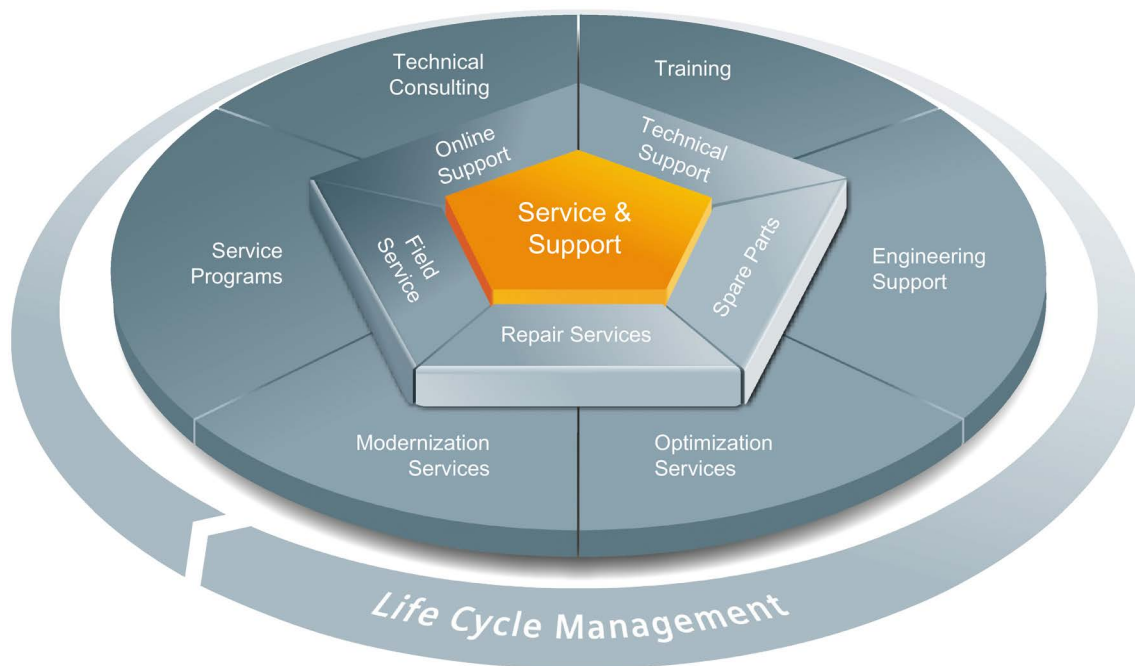
Two-step control (Page 480)

Input parameters PULSEGEN (Page 481)

Output parameter PULSEGEN (Page 482)

# Service & Support

# A



## Unmatched complete service for the entire life cycle

For machine manufacturers, solution providers and plant operators: The service offering from Siemens Industry Automation and Drive Technologies includes comprehensive services for a wide range of different users in all sectors of the manufacturing and process industry.

To accompany our products and systems, we offer integrated and structured services that provide valuable support in every phase of the life cycle of your machine or plant – from planning and implementation through commissioning as far as maintenance and modernization.

Our Service & Support accompanies you worldwide in all matters concerning automation and drive technology from Siemens. We provide direct on-site support in more than 100 countries through all phases of the life cycle of your machines and plants.

You have an experienced team of specialists at your side to provide active support and bundled know-how. Regular training courses and intensive contact among our employees – even across continents – ensure reliable service in the most diverse areas.

## Online Support

The comprehensive online information platform supports you in all aspects of our Service & Support at any time and from any location in the world.

You can find Online Support at the following address on the Internet.



## Technical Consulting

Support in planning and designing your project: From detailed actual-state analysis, definition of the goal and consultation on product and system questions right through to the creation of the automation solution.

## Technical Support

Expert advice on technical questions with a wide range of demand-optimized services for all our products and systems.

You can find Technical Support at the following address on the Internet.

## Training

Extend your competitive edge – through practical know-how directly from the manufacturer.

You can find the training courses at the following address on the Internet.

## Engineering Support

Support during project engineering and development with services fine-tuned to your requirements, from configuration through to implementation of an automation project.

## Field Service

Our Field Service offers you services for commissioning and maintenance – to ensure that your machines and plants are always available.

## Spare parts

In every sector worldwide, plants and systems are required to operate with constantly increasing reliability. We will provide you with the support you need to prevent a standstill from occurring in the first place: with a worldwide network and optimum logistics chains.

## Repairs

Downtimes cause problems in the plant as well as unnecessary costs. We can help you to reduce both to a minimum – with our worldwide repair facilities.

## Optimization

During the service life of machines and plants, there is often a great potential for increasing productivity or reducing costs.

To help you achieve this potential, we are offering a complete range of optimization services.

## Modernization

You can also rely on our support when it comes to modernization – with comprehensive services from the planning phase all the way to commissioning.

## Service programs

Our service programs are select service packages for an automation and drives system or product group. The individual services are coordinated with each other to ensure smooth coverage of the entire life cycle and support optimum use of your products and systems.

The services of a service program can be flexibly adapted at any time and used separately.

Examples of service programs:

- Service contracts
- Plant IT Security Services
- Life Cycle Services for Drive Engineering
- SIMATIC PCS 7 Life Cycle Services
- SINUMERIK Manufacturing Excellence
- SIMATIC Remote Support Services

Benefits at a glance:

- Reduced downtimes for increased productivity
- Optimized maintenance costs due to a tailored scope of services
- Costs that can be calculated and therefore planned
- Service reliability due to guaranteed response times and spare part delivery times
- Customer service personnel will be supported and relieved of additional tasks
- Comprehensive service from a single source, fewer interfaces and greater expertise

## Contact

At your service locally, around the globe: your partner for consultation, sales, training, service, support, spare parts... for the entire range of products from Industry Automation and Drive Technologies.

You can find your personal contact in our contacts database on the Internet.

# Index

## C

### CONT\_C

- Block diagram, 464
- Input parameters, 465
- Mode of operation, 462
- Output parameters, 467

### CONT\_S

- Block diagram, 470
- Input parameters, 471
- Instruction, 468
- Mode of operation, 469
- Output parameters, 472

## P

### PID\_3Step

- In/out parameters, 330
- Input parameters, 327, 362
- Instruction, 316, 353
- Output parameters, 329, 364
- Static tags, 366

### PID\_Compact

- In/out parameters, 263
- Input parameters, 260, 296
- Instruction, 292
- Output parameters, 262, 297
- Static tags, 264, 298

### PID\_Temp

- ActivateRecoverMode tag, 453
- Cascade, 405
- Cascading, 198
- ErrorBits parameter, 450
- In/out parameters, 405
- Input parameters, 401
- Mode, 405
- Multi-zone applications, 206
- Operating principle, 395
- Output parameters, 403
- PID\_Temp state and mode parameters, 440
- PwmPeriode, 456
- Static tags, 407
- Tag Warning, 455

### PULSEGEN

- Input parameters, 481
- Output parameters, 482

### PULSEGEN

- Instruction, 473
- Mode of operation, 474

## S

### Software controller

- Configuring, 38

### Symbol

- For value comparison, 42

## T

### TCONT\_CP

- In/out parameters, 500
- Input parameters, 498
- Instruction, 483
- Mode of operation, 484
- Output parameters, 499
- Static tags, 501

### TCONT\_S

- In/out parameters, 516
- Input parameters, 515
- Instruction, 508
- Operating principle, 509
- Output parameters, 516
- Static tags, 517

### Technology objects

- CONT\_C, 214
- CONT\_S, 220
- PID\_3Step, 120
- PID\_Compact, 119
- PID\_Temp, 164
- TCONT\_CP, 223
- TCONT\_S, 246

## V

### Values

- Comparing, 42