

SIEMENS

SIMATIC

PID Temperature Control

Manual

Preface, Contents

Introduction

Continuous Temperature
Controller FB 58 "TCONT_CP"

Controller Tuning in FB 58
"TCONT_CP"

Temperature Step Controller
FB59 "TCONT_S"

Getting Started

Examples for the Temperature
Controllers

Appendix

Abbreviations and Acronyms

Index

1

2

3

4

5

6

A

B

Safety Guidelines

This manual contains notices intended to ensure personal safety, as well as to protect the products and connected equipment against damage. These notices are highlighted by the symbols shown below and graded according to severity by the following texts:



Danger

indicates that death, severe personal injury or substantial property damage will result if proper precautions are not taken.



Warning

indicates that death, severe personal injury or substantial property damage can result if proper precautions are not taken.



Caution

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

Caution

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

Notice

draws your attention to particularly important information on the product, handling the product, or to a particular part of the documentation.

Qualified Personnel

Only **qualified personnel** should be allowed to install and work on this equipment. Qualified persons are defined as persons who are authorized to commission, to ground and to tag circuits, equipment, and systems in accordance with established safety practices and standards.

Correct Usage

Note the following:



Warning

This device and its components may only be used for the applications described in the catalog or the technical description, and only in connection with devices or components from other manufacturers which have been approved or recommended by Siemens.

This product can only function correctly and safely if it is transported, stored, set up, and installed correctly, and operated and maintained as recommended.

Trademarks

SIMATIC®, SIMATIC HMI® and SIMATIC NET® are registered trademarks of SIEMENS AG.

Third parties using for their own purposes any other names in this document which refer to trademarks might infringe upon the rights of the trademark owners.

Copyright © Siemens AG 2001-2003 All rights reserved

The reproduction, transmission or use of this document or its contents is not permitted without express written authority. Offenders will be liable for damages. All rights, including rights created by patent grant or registration of a utility model or design, are reserved.

Siemens AG
Bereich Automation and Drives
Geschaeftsgebiet Industrial Automation Systems
Postfach 4848, D- 90327 Nuernberg

Disclaimer of Liability

We have checked the contents of this manual for agreement with the hardware and software described. Since deviations cannot be precluded entirely, we cannot guarantee full agreement. However, the data in this manual are reviewed regularly and any necessary corrections included in subsequent editions. Suggestions for improvement are welcomed.

©Siemens AG 2001-2003
Technical data subject to change.

Preface

Purpose of the Manual

This manual supports you when you work with the temperature controller block from the **Standard Library > PID Control**. It will familiarize you with the functions of the controller blocks and, in particular, with tuning the controller and working with the user interface in which you set the parameters for the blocks. There is an online help system for both the blocks and the user interface that supports you when setting the parameters of the blocks.

This manual is intended for qualified personnel involved in programming, configuration, commissioning, and servicing of programmable controllers.

We recommend that you spend some time studying the "Examples of Temperature Controllers" in Chapter 6. These examples will help you to understand the application of temperature controllers quickly and simply.

Basic Knowledge Required

To understand this manual, you should be familiar with automation engineering and know the basics of closed-loop control.

You should also be familiar with using computers or similar tools (for example programming devices) with the Windows 95/98/NT/2000 or Me operating system. Since PID Temperature Control is used in conjunction with the STEP 7 basic software, you should be familiar with working with the basic software as described in the "Programming with STEP 7 V5.1" manual.

Scope of the Manual

This manual applies to the temperature controllers of the **Standard Library > PID Control** of the STEP 7 programming software, Version V5.1 Service Pack 3 and higher.

STEP 7 Documentation Packages

This manual is part of the STEP 7 Basic Information documentation package.

Manuals	Purpose	Order Number
STEP 7 Basic Information with <ul style="list-style-type: none"> • Working with STEP 7 V5.1 Getting Started • Programming with STEP 7 V5.1 • Configuring Hardware and Communication Connections, STEP 7 V5.1 • From S5 to S7, Convertor Manual 	Basic information for technical personnel describing the methods of implementing control tasks with STEP 7 and S7-300/400.	6ES7810-4CA05-8BA0
STEP 7 Reference with <ul style="list-style-type: none"> • Ladder Logic (LAD) / Function Block Diagram (FBD) / Statement List (STL) for S7-300/400 manuals • Standard and System Functions for S7-300/400 	Provides reference information and describes the programming languages LAD, FBD, and STL and standard and system functions extending the scope of STEP 7 basic information.	6ES7810-4CA05-8BR0
Elect. manual <ul style="list-style-type: none"> • PID Temperature Control 	This manual describes the temperature controllers of the Standard Library > PID Control.	Part of the STEP 7 software package

Online Help Systems	Purpose	Order Number
Help on STEP 7	Basic information on programming and configuring hardware with STEP 7 in the form of an online help.	Part of the STEP 7 software package
Reference help systems for <ul style="list-style-type: none"> • LAD/FBD/STL • SFBs/SFCs • Organization blocks • PID Temperature Control 	Context-sensitive reference information	Part of the STEP 7 software package

Further Closed-Loop Control Products in SIMATIC S7

- *SIMATIC S7* User Manuals: Standard PID Control, Modular PID Control, PID Self-Tuner, FM355/455 PID Control
- Jürgen Müller, "Regeln mit SIMATIC - Praxisbuch für Regelungen mit SIMATIC S7 und PCS7" published by MCI Publicis Verlag ISBN 3-89578-147-9 (German only)

Further Support

If you have any technical questions, please get in touch with your Siemens representative or agent responsible.

You will find your contact person at:

<http://www.ad.siemens.de/partner>

Training Centers

Siemens offers a number of training courses to familiarize you with the SIMATIC S7 automation system. Please contact your regional training center or our central training center in D 90327 Nuremberg, Germany for details:

Telephone: +49 (911) 895-3200.

Internet: <http://www.sitrain.com>

SIMATIC Documentation on the Internet

Documentation is available free of charge on the Internet at:

<http://www.ad.siemens.de/support>

Here, you can use the Knowledge Manager to locate the documentation you require quickly. If you have questions or suggestions regarding the documentation, a "Documentation" conference is available in the Internet Forum.

A&D Technical Support

Worldwide, available 24 hours a day:



<p>Worldwide (Nuernberg) Technical Support</p> <p>24 hours a day, 365 days a year Phone: +49 (180) 5050-222 Fax: +49 (180) 5050-223 E-Mail: adsupport@siemens.com GMT: +1:00</p>		
<p>Europe / Africa (Nuernberg) Authorization</p> <p>Local time: Mon.-Fri. 8:00 to 5:00 PM Phone: +49 (180) 5050-222 Fax: +49 (180) 5050-223 E-Mail: adsupport@siemens.com GMT: +1:00</p>	<p>United States (Johnson City) Technical Support and Authorization</p> <p>Local time: Mon.-Fri. 8:00 to 5:00 PM Phone: +1 (423) 262 2522 Fax: +1 (423) 262 2289 E-Mail: simatic.hotline@sea.siemens.com GMT: -5:00</p>	<p>Asia / Australia (Beijing) Technical Support and Authorization</p> <p>Local time: Mon.-Fri. 8:00 to 5:00 PM Phone: +86 10 64 75 75 75 Fax: +86 10 64 74 74 74 E-Mail: adsupport.asia@siemens.com GMT: +8:00</p>
<p>The languages of the SIMATIC Hotlines and the authorization hotline are generally German and English.</p>		

Service & Support on the Internet

In addition to our documentation, we offer our Know-how online on the internet at:

<http://www.siemens.com/automation/service&support>

where you will find the following:

- The newsletter, which constantly provides you with up-to-date information on your products.
- The right documents via our Search function in Service & Support.
- A forum, where users and experts from all over the world exchange their experiences.
- Your local representative for Automation & Drives.
- Information on field service, repairs, spare parts and more under "Services".

Contents

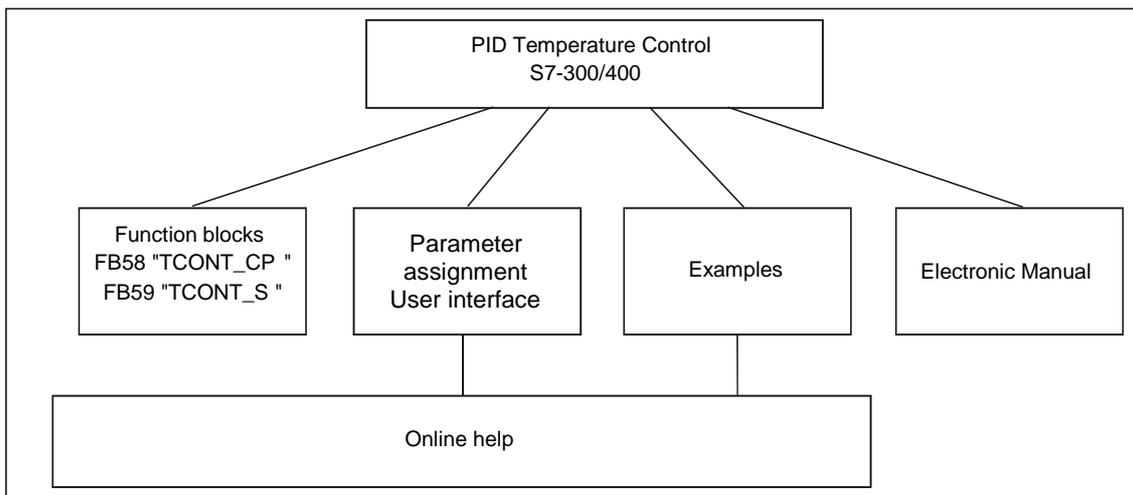
1	Introduction	1-1
1.1	FB 58 "TCONT_CP"	1-3
1.2	FB59 "TCONT_S"	1-4
2	Continuous Temperature Controller FB 58 "TCONT_CP"	2-1
2.1	Controller Section.....	2-1
2.1.1	Forming the Error	2-1
2.1.2	PID Algorithm	2-4
2.1.3	Calculating the Manipulated Variable	2-6
2.1.4	Saving and Reloading Controller Parameters.....	2-9
2.2	Pulse Generator PULSEGEN (PULSE_ON).....	2-11
2.3	Block Diagram	2-13
2.4	Including the Function Block in the User Program.....	2-14
2.4.1	Calling the Controller Block.....	2-14
2.4.2	Call without Pulse Generator (continuous controller)	2-15
2.4.3	Call with Pulse Generator (pulse controller)	2-15
2.4.4	Initialization.....	2-18
3	Controller Tuning in FB 58 "TCONT_CP"	3-1
3.1	Introduction.....	3-1
3.2	Process Types	3-2
3.3	Area of Application	3-3
3.4	The Phases of Controller Tuning	3-4
3.5	Preparations	3-6
3.6	Starting Tuning (Phase 1 -> 2).....	3-8
3.7	Searching for the Point of Inflection (Phase 2) and Calculating the Control Parameters (Phase 3, 4, 5)	3-10
3.8	Checking the Process Type (Phase 7)	3-10
3.9	Result of the Tuning	3-11
3.10	Tuning Stopped by the Operator.....	3-11
3.11	Error Situations and Remedies	3-12
3.12	Manual Fine Tuning in Control Mode.....	3-16
3.13	Parallel Tuning of Control Channels	3-19
4	Temperature Step Controller FB59 "TCONT_S"	4-1
4.1	Controller Section.....	4-1
4.1.1	Forming the Error	4-1
4.1.2	PI Step Controller Algorithm.....	4-4
4.2	Block Diagram	4-5
4.3	Including the Function Block in the User Program.....	4-6
4.3.1	Calling the Controller Block.....	4-6
4.3.2	Sampling Time	4-7
4.3.3	Initialization.....	4-7
5	Getting Started	5-1

6	Examples for the Temperature Controllers	6-1
6.1	Introduction.....	6-1
6.2	Example with FB 58 "TCONT_CP" (pulse control)	6-2
6.3	Samples for FB 58 "TCONT_CP" with Short Pulse Generator Sampling Time.....	6-6
6.4	Sample for FB 58 "TCONT_CP" (Continuous)	6-7
6.5	Sample for FB 59 "TCONT_S" (Step Controller)	6-11
A	Appendix	A-1
A.1	Technical Specifications.....	A-1
A.2	Execution Times.....	A-1
A.3	DB Assignment	A-2
A.3.1	Instance DB for FB 58 "TCONT_CP"	A-2
A.3.2	Instance DB for FB 59 "TCONT_S"	A-13
A.4	List of Possible Messages during Tuning	A-17
B	Abbreviations and Acronyms	B-1

Index

1 Introduction

Product Structure of "PID Temperature Control"



After you have installed STEP 7, the various parts of STEP 7 PID Temperature Control are located in the following folders:

- SIEMENS\STEP7\S7LIBS\ : FBs
- SIEMENS\STEP7\S7WRT\ : parameter assignment user interface, readme, online help
- SIEMENS\STEP7\EXAMPLES\ : sample programs
- SIEMENS\STEP7\MANUAL\ : manual

Function Blocks

The "Standard Library PID Control" contains two temperature controllers:

1. FB 58 "TCONT_CP":
Temperature controller for actuators with a continuous or pulsed input signal. This controller block also includes a self-tuning function for the PI/PID parameters.
2. FB 59 "TCONT_S":
Temperature step controller for actuators with an integral component such as a positioning motor.

The control blocks are purely software controllers in which a block includes the entire functionality of the controller. The data required for cyclic calculation are stored in the corresponding instance data blocks.

Parameter Assignment User Interface

You set the parameters for the controller and tune it using the parameter assignment user interface. The parameter settings are stored in the relevant instance DB. You can start the parameter assignment user interface by double-clicking on the relevant instance data block.

Online Help

You will find a description of the parameter assignment user interface and the function blocks in the online help systems.

Opening the Readme File

The readme file contains the latest information on the software you have received. You will find this file in the Windows Start menu.

1.1 FB 58 "TCONT_CP"

FB 58 "TCONT_CP" is used to control temperature processes with continuous or pulsed control signals. You can set parameters to enable or disable subfunctions of the PID controller and adapt it to the process. These settings can be made simply with the parameter assignment tool. You start this within a project by double-clicking on the instance DB in the SIMATIC Manager. You can open the electronic manual as follows:

Start > Simatic > Documentation > English > PID Temperature Control.

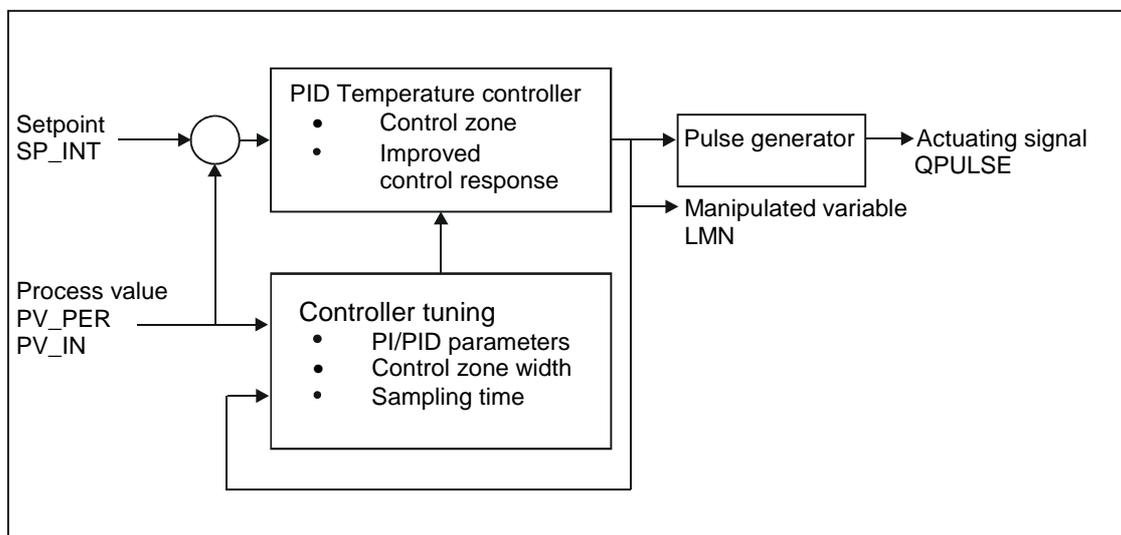
Application

The functionality is based on the PID control algorithm with additional functions for temperature processes. The controller supplies analog manipulated values and pulse-duration modulated actuating signals. The controller outputs signals to one actuator; in other words, with one controller, you can either heat or cool but not both.

Using the Controller in a Heating or Cooling Process

FB TCONT_CP can be used either purely for heating or purely for cooling. If you use the block for cooling, GAIN must be assigned a negative value. This inversion of the controller means that, for example if the temperature rises, the manipulated variable LMN and with it the cooling effort is increased.

Outline of the Structure



Description

Apart from the functions in the setpoint and process value branches, the FB implements a complete PID temperature controller with a continuous and binary manipulated variable output. To improve the control response with temperature processes, the block includes a control zone and reduction of the P-action if there is a setpoint step change. The block can set the PI/PID parameters itself using the controller tuning function.

1.2 FB59 "TCONT_S"

FB59 "TCONT_S" is used to control technical temperature processes with binary controller output signals for integrating actuators on the SIMATIC S7 programmable controllers. By setting parameters, subfunctions of the PI step controller can be activated or deactivated and the controller adapted to the process. These settings can be made simply in the parameter assignment user interface. You start this within a project by double-clicking on the instance DB in the SIMATIC Manager. You can open the electronic manual as follows:

Start > Simatic > Documentation > English > PID Temperature Control.

Application

The functionality is based on the PI control algorithm of the sampling controller. This is supplemented by the functions for generating the binary output signal from the analog actuating signal.

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 (integral 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.

Description

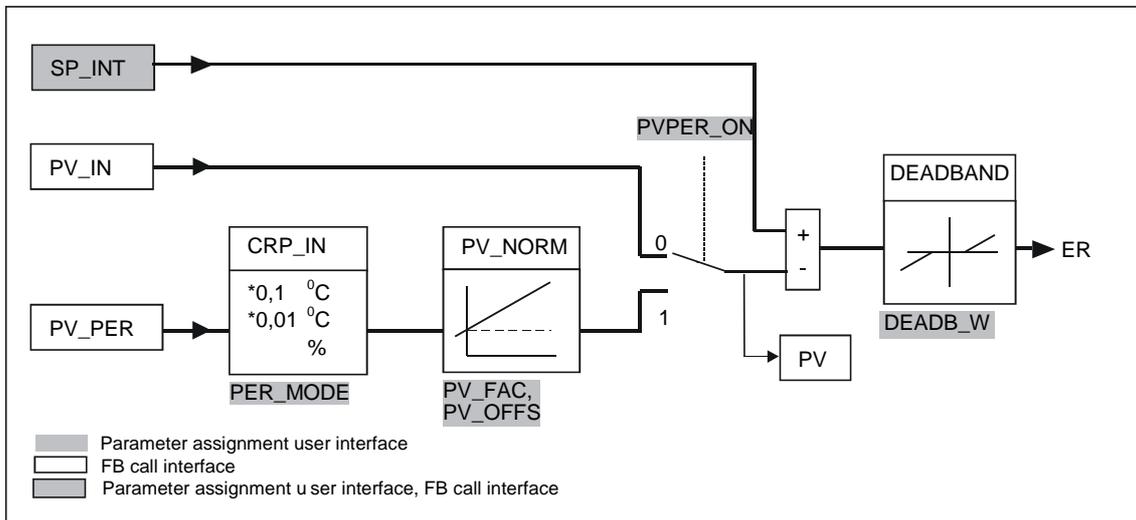
Apart from the functions in the process variable branch, FB 59 "TCONT_S" implements a complete PI controller with binary manipulated value output and the option of influencing the controller output signals manually. The step controller operates without a position feedback signal.

2 Continuous Temperature Controller FB 58 "TCONT_CP"

2.1 Controller Section

2.1.1 Forming the Error

The schematic below is a block diagram illustrating how the error is formed:



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 error must have the same unit.

Process Value Options (PVPER_ON)

Depending on PVPER_ON, the process value can be acquired in the peripheral (I/O) or floating-point format.

PVPER_ON	Process Value Input
TRUE	The process value is read in via the analog peripheral 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 peripheral value PV_PER to a floating-point format depending on the switch PER_MODE 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 Normalization PV_NORM (PF_FAC, PV_OFFS)

The PV_NORM function calculates the output of CRP_IN according to the following rule:

$$\text{"Output of PV_NORM"} = \text{"Output of CPR_IN"} * PV_FAC + PV_OFFS$$

It can be used for the following purposes:

- Process value correction with PV_FAC as the process value factor and PV_OFFS as the 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 is the lower limit

With the default values ($PV_FAC = 1.0$ and $PV_OFFS = 0.0$), normalization is disabled. The effective process value is output at the PV output.

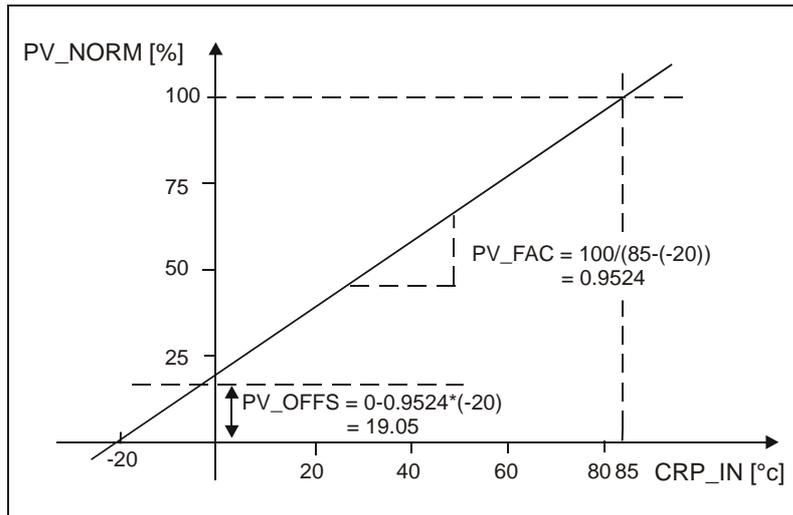
Note

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 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 schematic below shows an example of adapting the temperature range -20 to 85 °C to an internal scale of 0 to 100 %:



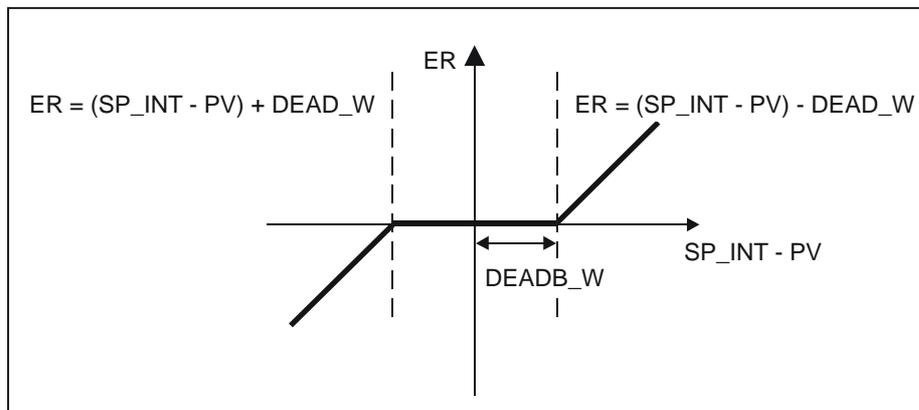
Forming the Error

The difference between the setpoint and process value is the error before the deadband.

The setpoint and process value must exist in the same unit.

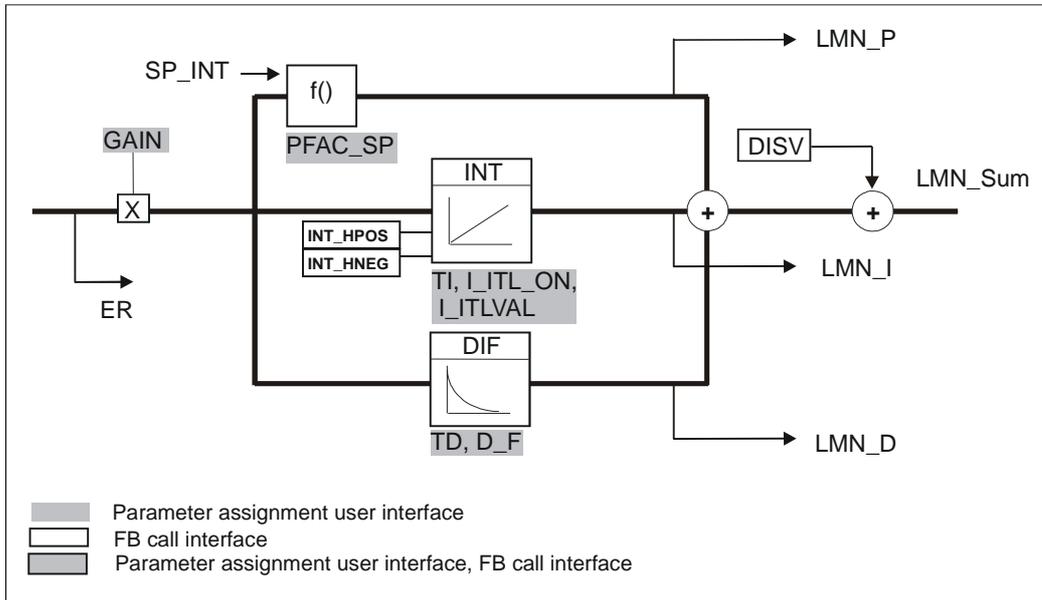
Deadband (DEADB_W)

To suppress a small constant oscillation due to the manipulated variable quantization (for example in pulse duration modulation with PULSEGEN) a deadband (DEADBAND) is applied to the error. If DEADB_W = 0.0, the deadband is deactivated. The effective error is indicated by the ER parameter.



2.1.2 PID Algorithm

The schematic below is 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.

The controller tuning supports PI and PID controllers. Controller inversion is implemented using a negative GAIN (cooling controller).

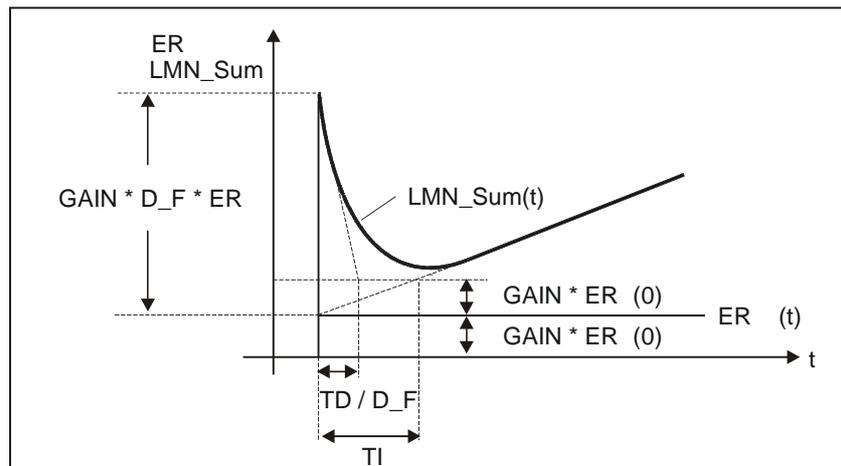
If you set TI and TD to 0.0, you obtain a pure P controller at the operating point.

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) is the manipulated variable in automatic mode of the controller
- ER (0) is the step change of the normalized error
- GAIN is the controller gain
- TI is the integral time
- TD is the derivative time
- D_F is the derivative factor



Integrator (TI, I_ITL_ON, I_ITLVAL)

In the manual mode, it is corrected as follows: $LMN_I = LMN - LMN_P - DISV$.

If the manipulated variable is limited, the I-action is stopped. If the error moves the I-action back in the direction of the manipulated variable range, the I-action is enabled again.

The I-action is also modified by the following measures:

- The I-action of the controller is deactivated by $TI = 0.0$
- Weakening the P-action when setpoint changes occur
- Control zone
- Online modification of the manipulated value limits

Weakening the P-Action when Setpoint Changes Occur (PFAC_SP)

To prevent overshoot, you can weaken the P-action using the "proportional factor for setpoint changes" parameter (PFAC_SP). Using PFAC_SP, you can 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

The weakening of the P-action is achieved by compensating the I-action.

Derivative action element (TD, D_F)

- The D-action of the controller is deactivated with $TD = 0.0$.
- If the D-action is active, the following relationship should apply:
 $TD \geq 0.5 * CYCLE * D_F$

Parameter Settings of a P or PD Controller with Operating Point

In the user interface, deactivate the I-action (TI = 0.0) and possible also the D-action (TD = 0.0). Then make the following parameter settings:

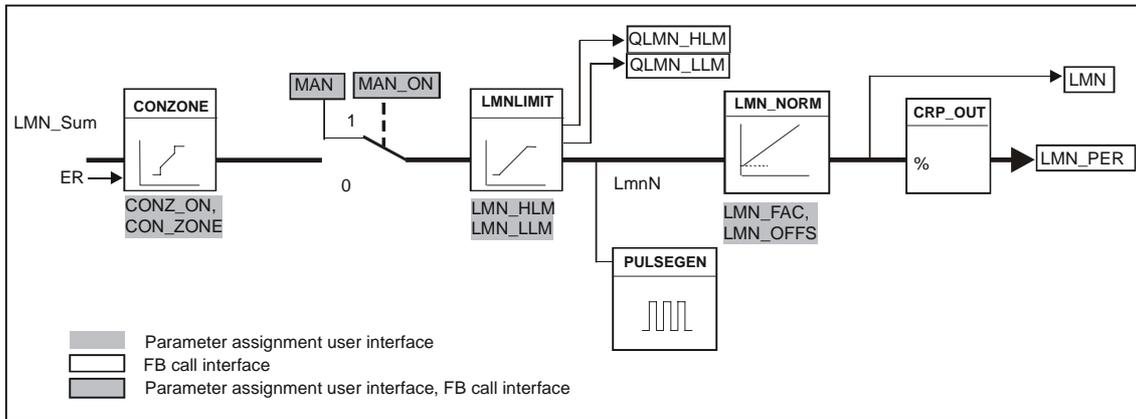
- I_ITL_ON = TRUE
- I_ITLVAL = operating point;

Feedforward Control (DISV)

A feedforward variable can be added at the DISV input.

2.1.3 Calculating the Manipulated Variable

The schematic below is the block diagram of the manipulated variable calculation:



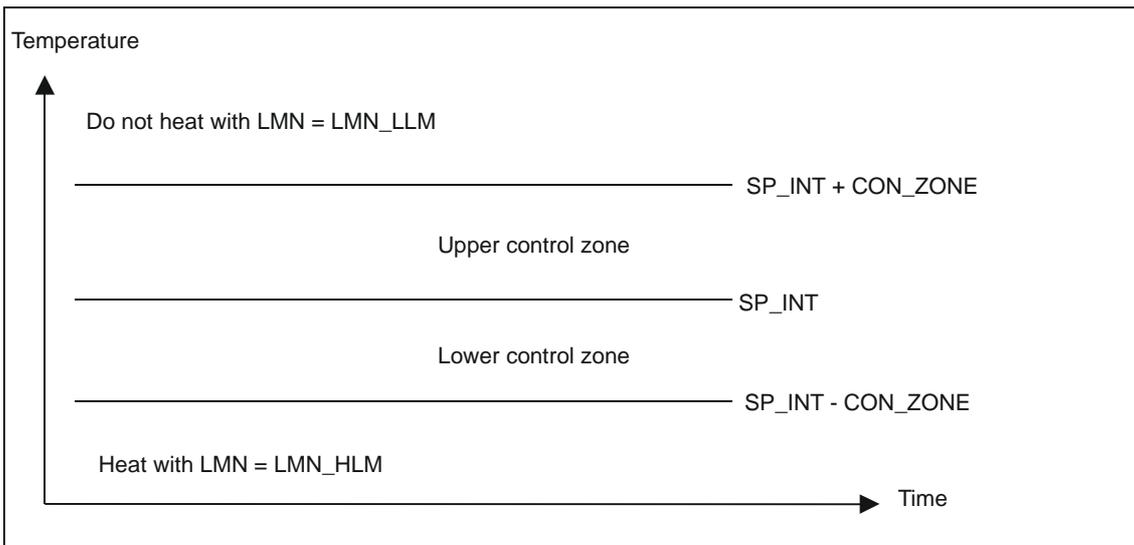
Control Zone (CONZ_ON, CON_ZONE)

If CONZ_ON = TRUE, the controller operates with a control zone. This means that the controller operates according to the following algorithm:

- If PV exceeds SP_INT by more than CON_ZONE, the value LMN_LLM is output as the manipulated variable (controlled closed-loop).
- If PV falls below SP_INT by more than CON_ZONE, the value LMN_HLM is output as the manipulated variable (controlled closed-loop).
- If PV is within the control zone (CON_ZONE), the manipulated variable takes its value from the PID algorithm LMN_Sum (automatic closed-loop control).

Note

The changeover from controlled closed-loop to automatic closed-loop control takes into account a hysteresis of 20% of the control zone.



Note

Before activating 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 variable.

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, basically only the reducing P-action would 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 switch over between manual and automatic operation. In the manual mode, the manipulated variable is corrected to a manual 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. Switching over to automatic mode is therefore bumpless.

Note

During tuning, the MAN_ON parameter is not effective.

Manipulated Variable Limitation LMNLIMIT (LMN_HLM, LMN_LLM)

The value of the manipulated variable is limited to the LMN_HLM and LMN_LLM limits by the LMNLIMIT function. If these limits are reached, this is indicated by the message bits QLMN_HLM and QLMN_LLM.

If the manipulated variable is limited, the I-action is stopped. If the error moves the I-action back in the direction of the manipulated variable range, the I-action is enabled again.

Changing the Manipulated Variable Limits Online

If the range of the manipulated variable is reduced and the new unlimited value of the manipulated variable is outside the limits, the I-action and therefore the value of the manipulated variable shifts.

The manipulated variable is reduced by the same amount as the manipulated variable limit changed. If the manipulated variable was unlimited prior to the change, it is set exactly to the new limit (described here for the upper manipulated variable limit).

Manipulated Variable Normalization LMN_NORM (LMN_FAC, LMN_OFFS)

The LMN_NORM function normalizes the manipulated variable according to the following formula:

$$LMN = LmnN * LMN_FAC + LMN_OFFS$$

It can be used for the following purposes:

- Manipulated variable adaptation with LMN_FAC as manipulated variable factor and LMN_OFFS manipulated variable offset

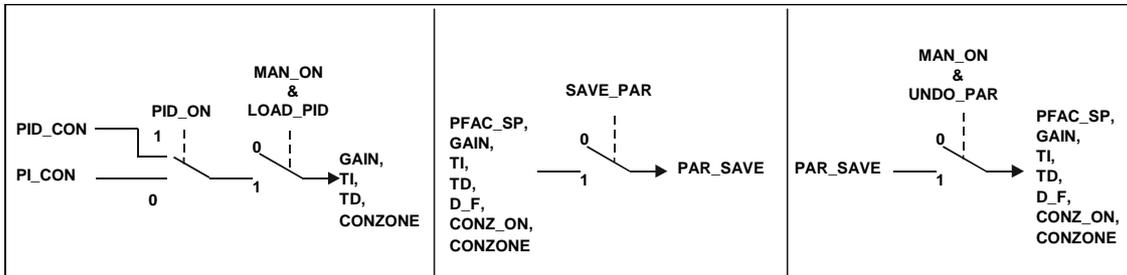
The value of the manipulated variable is also available in the peripheral format. The CRP_OUT function converts the LMN floating-point value to a peripheral value according to the following formula:

$$LMN_PER = LMN * 27648/100$$

With the default values (LMN_FAC = 1.0 and LMN_OFFS = 0.0), normalization is disabled. The effective manipulated variable is output at output LMN.

2.1.4 Saving and Reloading Controller Parameters

The schematic below shows the block diagram:



Saving Controller Parameters SAVE_PAR

If the current parameter settings are usable, you can save them in a special structure in the instance DB of FB58 "TCONT_CP" prior to making a manual change. If you tune 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 PAR_SAVE structure.

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).

Changing 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 the manual mode to write the PI or PID parameters to the effective controller parameters.

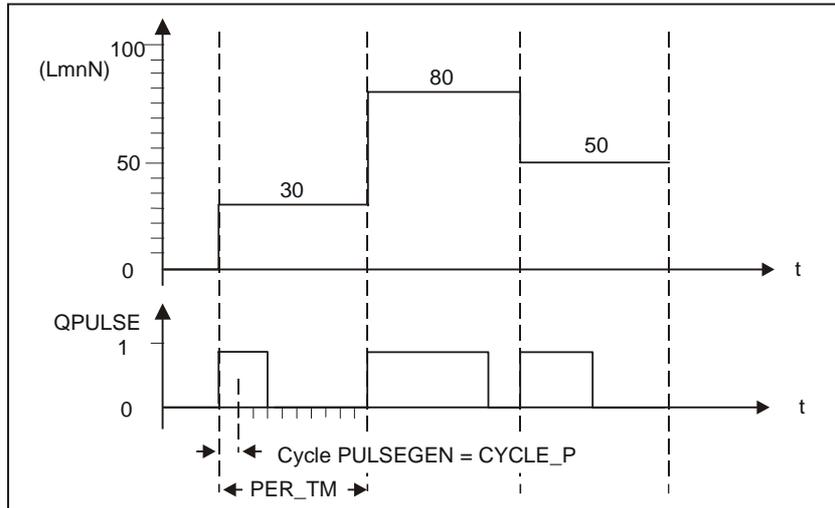
PID parameter PID_ON = TRUE	PI parameter PID_ON = FALSE
<ul style="list-style-type: none"> GAIN = PID_CON.GAIN TI = PID_CON.TI TD = PID_CON.TD 	<ul style="list-style-type: none"> GAIN = PI_CON.GAIN TI = PI_CON.TI

Note

- The controller parameters are only written back to the controller with UNDO_PAR or LOAD_PID when the controller gain is not 0:
LOAD_PID copies the parameters only if the relevant GAIN is $\neq 0$ (either of 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 were set, PID_ON will be set to FALSE and the PI parameters copied.
 - D_F, PFAC_SP are set to default values by 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 when CONZ_ON = FALSE is set.
-

2.2 Pulse Generator PULSEGEN (PULSE_ON)

The PULSEGEN function converts the analog manipulated variable value LmnN to a train of pulses with the period PER_TM using pulse duration modulation. PULSEGEN is activated with PULSE_ON=TRUE and is processed in the CYCLE_P cycle.



A manipulated variable value LmnN = 30 % and 10 PULSEGEN calls per PER_TM therefore means the following:

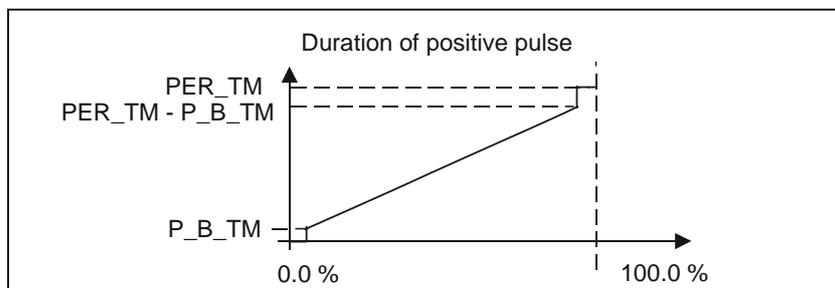
- TRUE at output QPULSE for the first three PULSEGEN calls (30 % of 10 calls)
- FALSE at output QPULSE for seven further PULSEGEN calls (70 % of 10 calls)

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 doglegs 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 and off times reduce the working life of switching elements and actuators. These can be avoided by setting a minimum pulse or minimum break time P_B_TM.

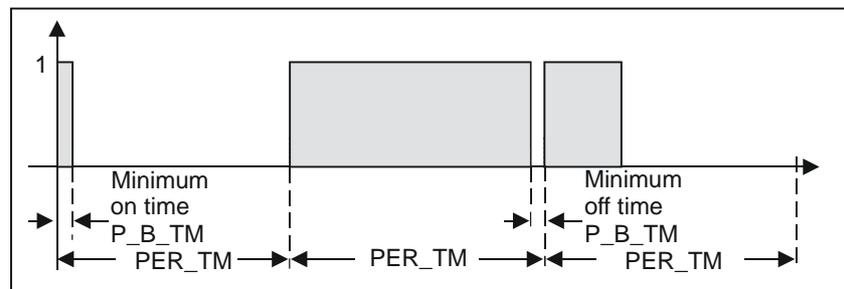
Small absolute values at the input variable LmnN that could otherwise generate a pulse time shorter than P_B_TM are suppressed.

High input values that would generate a pulse time longer than PER_TM - P_B_TM are set to 100%. This reduces the dynamics of pulse generation.

Values of $P_B_TM \leq 0.1 * PER_TM$ are recommended for the minimum pulse and the minimum break time.

The doglegs in the curves in the diagram above are caused by the minimum pulse time or minimum break time.

The schematic below illustrates the switching response of the pulse output:



Accuracy of Pulse Generation

The smaller the sampling time of the pulse generator CYCLE_P is compared with the pulse repetition period PER_TM, the more accurate the pulse duration modulation. For adequately accurate control, the following should apply:

$$CYCLE_P \leq PER_TM/50$$

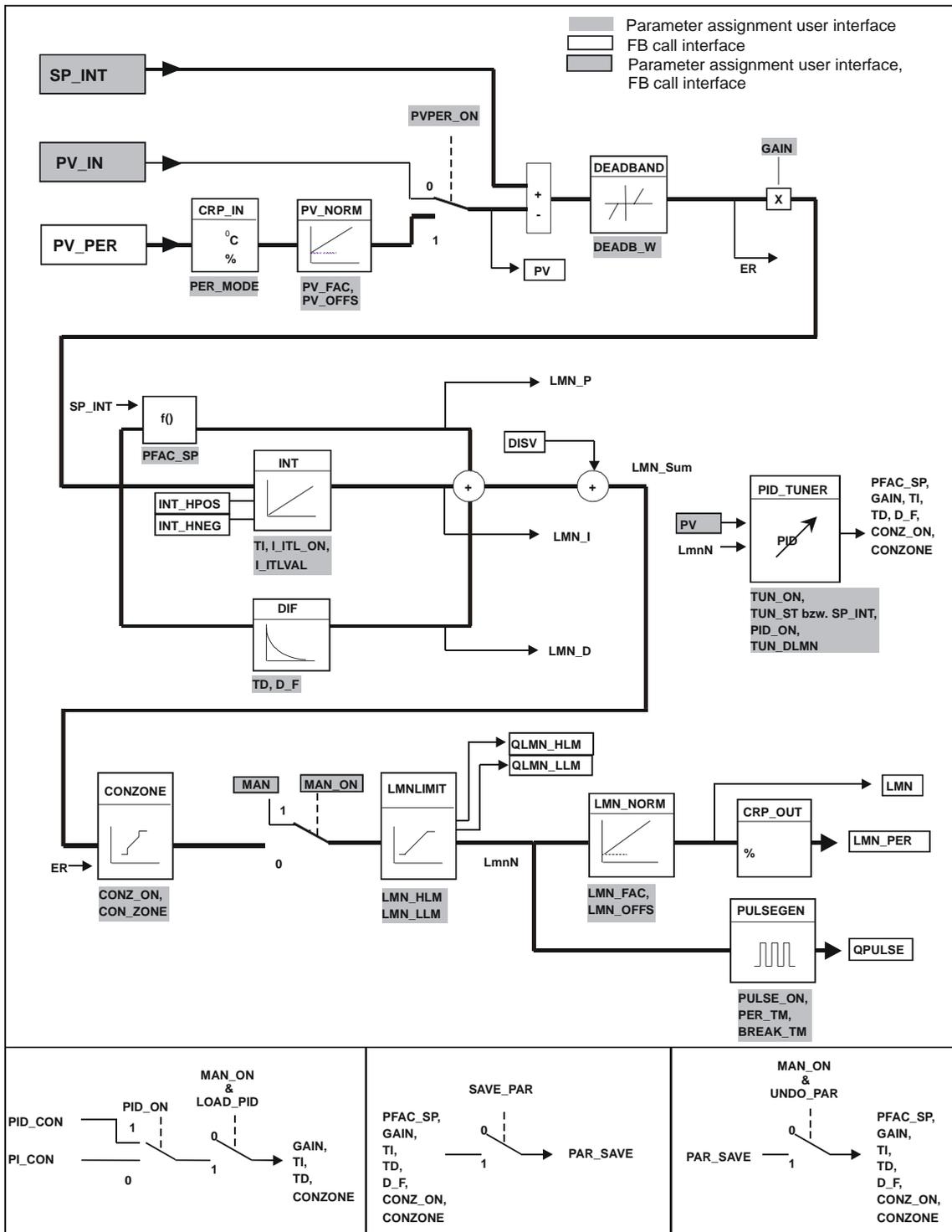
This means that the value of the manipulated variable is converted into pulses with a resolution of $\leq 2\%$ (see also of the example in Section 2.4.3, Page 2-15).

Notes

When you call the pulse generator cycle, you have to note the following:

If you call the controller in the pulse generator cycle, the process value will be averaged. As a result, different values may be output on the output PV and the input PV_IN or PV_PER. If you want to correct the setpoint value, you have to save the process value on the input parameter PV_IN at the call-up points of the entire controller processing (QC_ACT =TRUE). Other calls between the pulse generator cycle call and the call-up points of the the entire generator cycle are forwarded to the input parameters PV_IN and SP_INT with the saved process value.

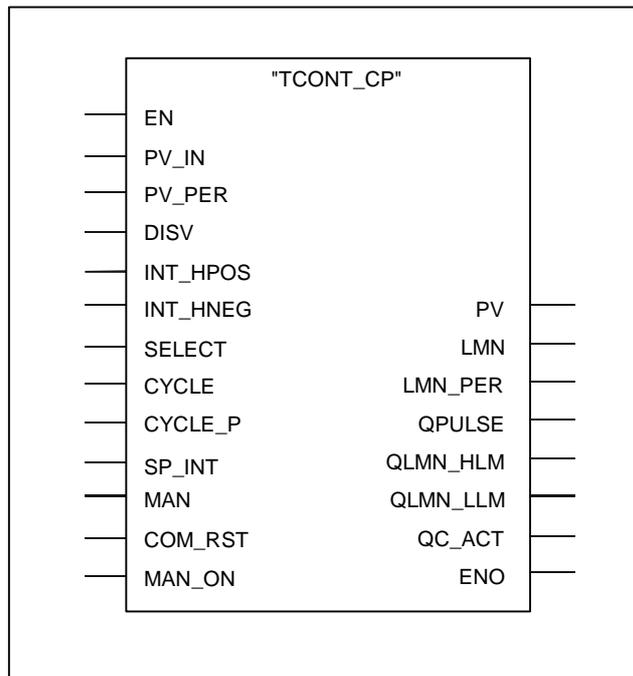
2.3 Block Diagram



2.4 Including the Function Block in the User Program

2.4.1 Calling the Controller Block

The following diagram shows the control call in FBD:



FB TCONT_CP must be called at constant intervals. To achieve this, use a cyclic interrupt OB (for example OB35 for an S7-300). The block interface provides the most important parameters that allow you to interconnect the block with process variables such as the setpoint, process value and manipulated variable (see also Appendix A.3 DB Assignment). You can also connect a manual value or disturbance variable directly to the block interface.

2.4.2 Call without Pulse Generator (continuous controller)

Controller Sampling Time CYCLE

You specify the sampling time at the CYCLE parameter. You can also enter the sampling time using the parameter assignment tool. The sampling time CYCLE must match the time difference between two calls (cycle time of the cyclic OB including scan rates).

During controller tuning, the block measures the time between the calls and compares it with the configured value CYCLE. If there is a difference of >5%, the optimization will be aborted and STATUS_H = 30005 is set.

Rule of Thumb for the CYCLE Controller Sampling Time

The controller sampling time should not exceed 10 % of the calculated integral time constant of the controller (TI):

$$\text{CYCLE} \leq \text{TI}/10$$

2.4.3 Call with Pulse Generator (pulse controller)

Controller Sampling Time CYCLE and Sampling Time of the Pulse Generator CYCLE_P

If you have activated the pulse generator stage (PULSE_ON = TRUE), you must enter two sampling times.

- Enter the sampling time of the pulse generator at the CYCLE_P input. This must match the clock rate of the calling cyclic interrupt OB. The duration of the generated pulse is always a whole multiple of this value.
- At the CYCLE input, you specify the sampling time for the other control functions of FB 58 "TCONT_CP".

During controller tuning, the block measures the times between the calls and compares it with the configured value CYCLE. If there is a difference of >5%, the optimization will be aborted and STATUS_H = 30005 is set.

FB 58 "TCONT_CP" calculates the scan rate and processes the control functions at the CYCLE sampling rate. Make sure that CYCLE is a whole multiple of CYCLE_P.

You can select a value for CYCLE that is lower than the pulse repetition period PER_TM. This can be useful when you require as high a pulse repetition period as possible to reduce wear on the actuators but when the sampling time needs to be low due to a fast process.

Rule of Thumb for the CYCLE and CYCLE_P Sampling Times

The controller sampling time should not exceed 10 % of the calculated integral time constant of the controller (TI): $CYCLE \leq TI/10$

For an adequately accurate manipulated variable resolution, make sure that the following relationship applies: $CYCLE_P \leq PER_TM/50$.

Rule of Thumb for the Pulse Repetition Period PER_TM

The pulse repetition period should not exceed 20 % of the calculated reset time of the controller (TI):

$$PER_TM \leq TI/5$$

Example of the Effects of the Parameters CYCLE_P, CYCLE and PER_TM:

$PER_TM = 10$ s, $CYCLE = 1$ s, $CYCLE_P = 100$ ms.

Every second, a new value is calculated for the manipulated variable, every 100 ms, the value is compared with the pulse length or break length output up to now.

- When a pulse is output, there are two possibilities:
 - The calculated value of the manipulated variable is higher than the pulse length/ PER_TM up to now. The pulse is then extended.
 - The calculated value of the manipulated variable is less than or equal to the pulse length/ PER_TM up to now. In this case, a pulse is no longer output.
- If no pulse is output, there are then also two possibilities:
 - The value (100 % - calculated value of the manipulated variable) is higher than the break length/ PER_TM up to now. The break is then extended.
 - The value (100 % - calculated value of the manipulated variable) is less than or equal to the break length/ PER_TM up to now. A pulse is then output.

Various Call Options for Pulse Control (SELECT)

In a fast process, extremely short pulse generator sampling times (for example 10 ms) are necessary. Due to the program run time (CPU utilization) it is not practical to process the control sections in the same cyclic interrupt OB as the calculation of the pulse output. You then either move the control functions to OB1 or to a slower cyclic interrupt OB (S7-400).

The following table provides an overview of the parameter settings for the SELECT input parameter:

Application	Block Call	Functionality
Default situation: The pulse generator sampling times are not particularly short on the S7-300 and S7-400 (for example CYCLE_P = 100 ms)	Call in cyclic interrupt OB with SELECT = 0	Control section and pulse output in the same cyclic interrupt OB
Short pulse generator sampling times on the S7-300 (for example CYCLE_P = 10 ms)	Conditional call (QC_ACT = TRUE) in OB1 with SELECT = 1	Control section in OB1
	Call in cyclic interrupt OB with SELECT = 2	Pulse output in cyclic interrupt OB
Short pulse generator sampling times on an S7-400 (for example CYCLE_P = 10 ms)	Call in slow cyclic interrupt OB with SELECT = 3	Control section in slow cyclic interrupt OB
	Call in fast cyclic interrupt OB with SELECT = 2	Pulse output in fast cyclic interrupt OB

Note

If you implement the processing of controller functions and pulse generator with two block calls, note the following:

- The process value (PV_IN or PV_PER) must be supplied with a value when the pulse generator is called. All other formal operands can be supplied with values when the controller functions are called.
- The SELECT parameter must be supplied with a value at every call.
- If you locate the call in OB1 with SELECT = 1, you implement the conditional call in the example "pulse controller, OB 35, OB 1".

Numeric Examples

Required Accuracy G	TI	CYCLE = TI/10	PER_TM = TI/5	CYCLE_P = PER_TM*G	Comment
1 %	100 s	10 s	20 s	0.2 s	Call with SELECT = 0 at a cycle time of 200 ms
1 %	5 s	0.5 s	1 s	0.01 s	Separate call of the pulse section in a separate cyclic interrupt level.

2.4.4 Initialization

FB "TCONT_CP" has an initialization routine that is processed when the input parameter COM_RST = TRUE is set. After processing the initialization routine, the block sets COM_RST back to FALSE.

During the initialization, the integral action is set to the value I_ITLVAL. When called in a cyclic interrupt level, it continues to operate starting at this value.

All other outputs are set to their initial values.

If you require initialization when the CPU restarts, call the block in OB100 with COM_RST = TRUE.

3 Controller Tuning in FB 58 "TCONT_CP"

3.1 Introduction

The controller optimization is to be used exclusively for heating or cooling processes.

With controller tuning in FB 58 "TCONT_CP", the PI/PID controller parameters are set automatically. There are two ways of tuning:

- Tuning by approaching the operating point with a setpoint step change
- Tuning at the operating point by setting a start bit

In both cases, the process is excited by a selectable manipulated variable step change. After detecting a point of inflection, the PI/PID controller parameters are available and the controller switches to automatic mode and continues to control with these parameters.

You can also tune your controller using the wizard in the parameter assignment user interface.

Optimizing the Controller Response

The controller design is intended for an optimum response to disturbances. The resulting "sharp" parameters would lead to overshoot of 10% to 40% of the step change in setpoint step changes. To avoid this, the P-action is weakened by the PFAC_SP parameter when a setpoint step change occurs. In typical temperature processes, overshoot as a result of large setpoint step changes can also be reduced by temporary use of a minimum or maximum manipulated variable (controlled closed-loop).

Measuring the Cycle Times CYCLE and CYCLE_P

At the beginning of the tuning process, the controller sampling time CYCLE and (if pulse control is active) the pulse generator sampling time CYCLE_P are measured. If the measured values differ by more than 5% of the configured value, the controller optimization will be aborted and STATUS_H = 30005 is set.

Saving the Controller Parameters (SAVE_PAR or UNDO_PAR)

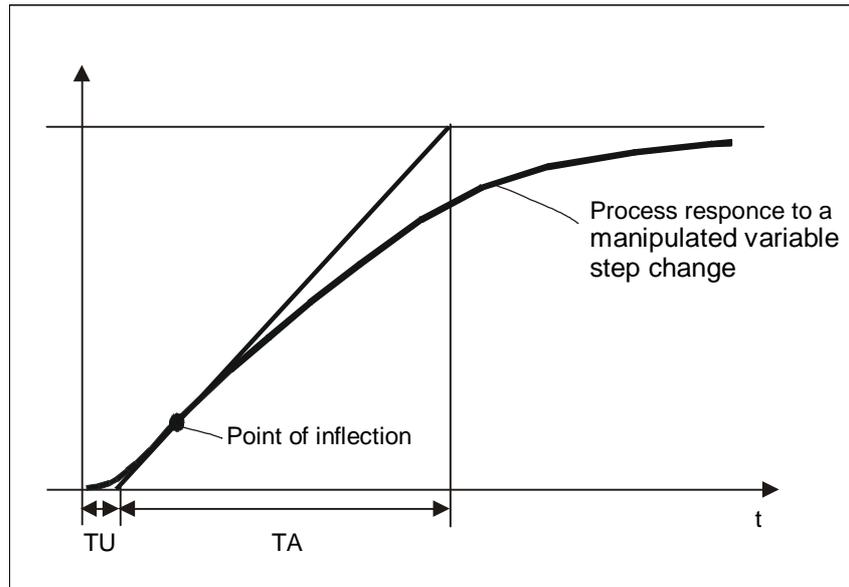
When you tune the controller, the parameters are saved before tuning is started. When tuning is completed, you can reactivate the parameter settings as they were prior to tuning using UNDO_PAR.

3.2 Process Types

Process Types

Apart from the process gain GAIN_P, the parameters shown in the schematic below, lag time TU and process time constant TA are characteristic parameters of a process.

The schematic below illustrates the step response:



The table below lists the various processes with which you can use FB 58 "TCONT_CP":

Process Type I	Process Type II	Process Type III
Typical temperature process (ideal situation)	Intermediate range	Higher order temperature process (high lag)
$TU/TA < 0.1$	TU/TA approx. 0.1	$TU/TA > 0.1$
One dominating time constant	2 approximately equivalent time constants	Several time constants

FB 58 "TCONT_CP" is designed for typical temperature control processes of type I. You can, however, also use the block for higher order processes of type II or III.

3.3 Area of Application

Transient Response

The process must have a stable, asymptotic transient response with time lag.

After a step change in the manipulated variable, the process variable must change to a steady state. This therefore excludes processes that have an oscillating response without control and processes that are not self-regulating (integrator in the process).

Linearity and Operating Range

The process must have a linear response over the operating range. A non-linear response occurs, for example, when the state of a unit is changed. The tuning must take place in a linear part of the operating range.

This means that both during tuning and during normal controlled operation, non-linear effects within the operating range must be insignificant. It is, however possible to retune the process when the operating point changes if the tuning is repeated in the close vicinity of the new operating point and providing that the non-linearity does not occur in the range covered during tuning.

If certain static non-linearities (for example valve characteristics) are known, it is always advisable to compensate these with a polyline curve to linearize the process behavior.

Disturbances 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 tuning the zones of an extruder, all zones must be heated at the same time.

For information on measurement noise and low-frequency interference, refer to Section 3.11, Page 3-12.

3.4 The Phases of Controller Tuning

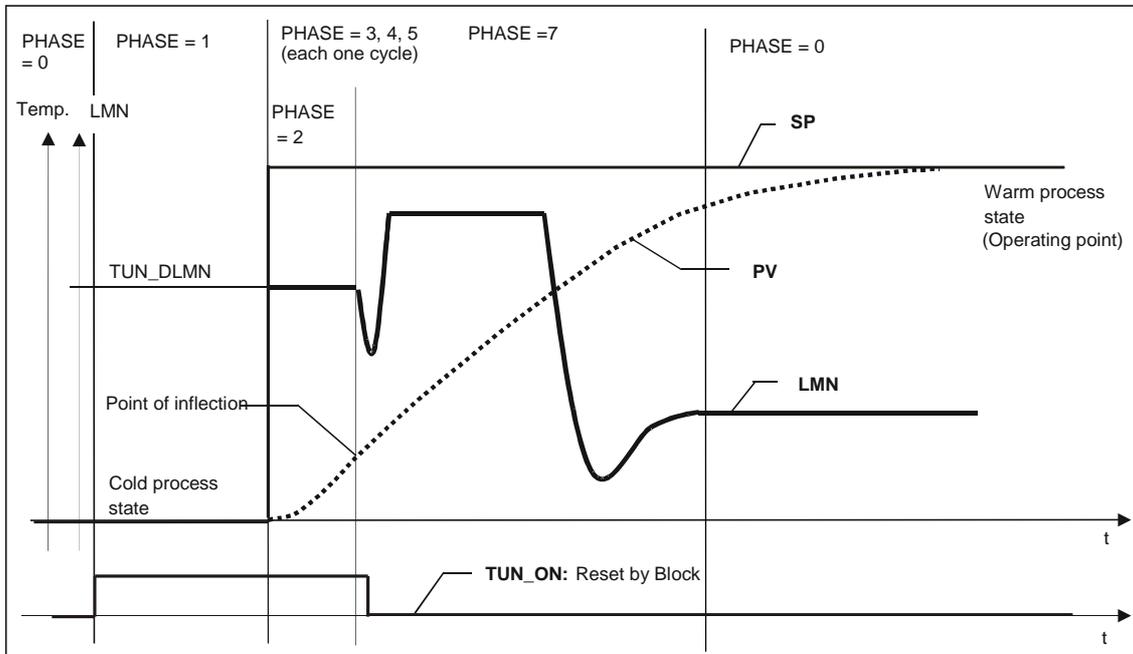
During tuning, several phases are run through in the block algorithm. The PHASE parameter indicates which phase the block is currently in.

You start the tuning as follows (see Section 3.6, Page 3-8):

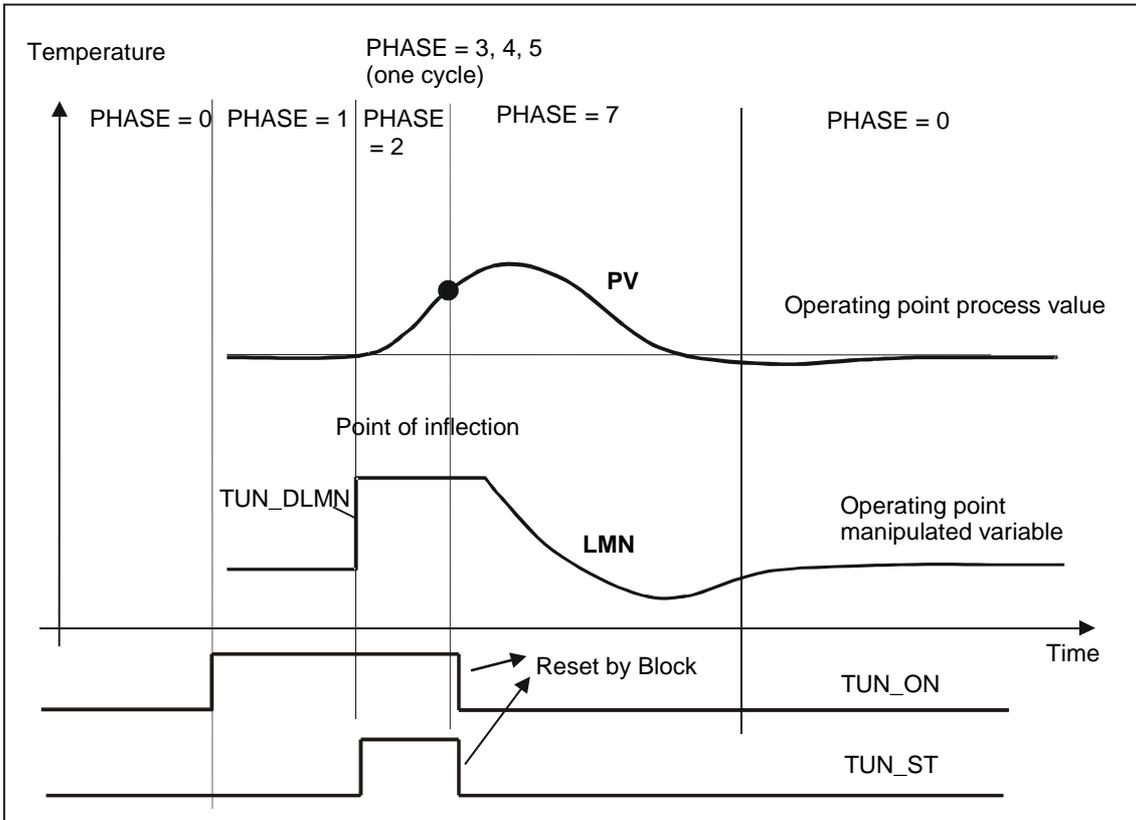
- Setting TUN_ON = TRUE prepares the controller for tuning. The controller changes from Phase 0 to 1.
- After waiting some time in phase 1, either set a setpoint step change at the SP_INT parameter or set TUN_ST = TRUE. The controller outputs a manipulated variable changed by the value TUN_DLMN and begins to search for the point of inflection.

PHASE	Description
0	No tuning; automatic or manual mode
1	Ready to start tuning; check parameters, wait for excitation, measure the sampling times
2	Actual tuning: Wait to detect point of inflection at a constant controller output value. Entry of the sampling time in the instance DB.
3 (1 cycle)	Calculation of the process parameters. The controller parameters valid prior to tuning are saved.
4 (1 cycle)	Controller design
5 (1 cycle)	Bring the controller to the new manipulated variable
7	Check the process type

The schematic below illustrates the phases of tuning as a result of a setpoint step change from the ambient temperature to the operating point:



The schematic below illustrates the phases of tuning at the operating point started with TUN_ST = TRUE:



At the end of the tuning (see Section 3.9, Page 3-11), when the block returns to Phase 0 and TUN_ON = FALSE is set, you can recognize whether or not the tuning was error free by the STATUS_H parameter.

3.5 Preparations

SIMATIC and Controller

Tuning is started by the in/out parameters TUN_ON, TUN_ST or SP_INT. You can set the parameters in the following ways:

- With the parameter assignment user interface
- With an operator control and monitoring device
- From the user program

You write to the in/out parameters only for one cycle since FB 58 "TCONT_CP" resets the parameters.



Warning

Death, serious injury, or considerable damage to property may occur.

During tuning, the MAN_ON parameter is not effective. As a result, the manipulated variable or process value can achieve unacceptable, extreme values.

The manipulated variable is set by the tuning functions. To stop the tuning, you must first set TUN_ON = FALSE. MAN_ON is then effective again.

Ensuring a Quasi Steady State Initial Situation (Phase 0)

If there is low-frequency oscillation of the controlled variable, for example due to bad controller parameters, the controller should be changed to manual prior to starting the tuning and you should wait until the oscillation dies down. As an alternative, you can change to a PI controller with less aggressive settings (low loop gain, high integral time).

You must now wait until a steady state is achieved; in other words, until the process value and value of the manipulated variable have settled. Asymptotic settling or slow drift of the process value is also permitted (quasi steady-state, see diagram below). The manipulated variable must be constant or fluctuate around a constant mean value.

Note

You should avoid changing the manipulated variable shortly before starting the tuning. A change in the manipulated variable can also be brought about accidentally when setting up the conditions required for tuning (for example closing a furnace door)! If this does happen, wait at least until the process variable begins to settle asymptotically towards the steady state. You will, however achieve better controller parameters if you wait until the transients have decayed completely.

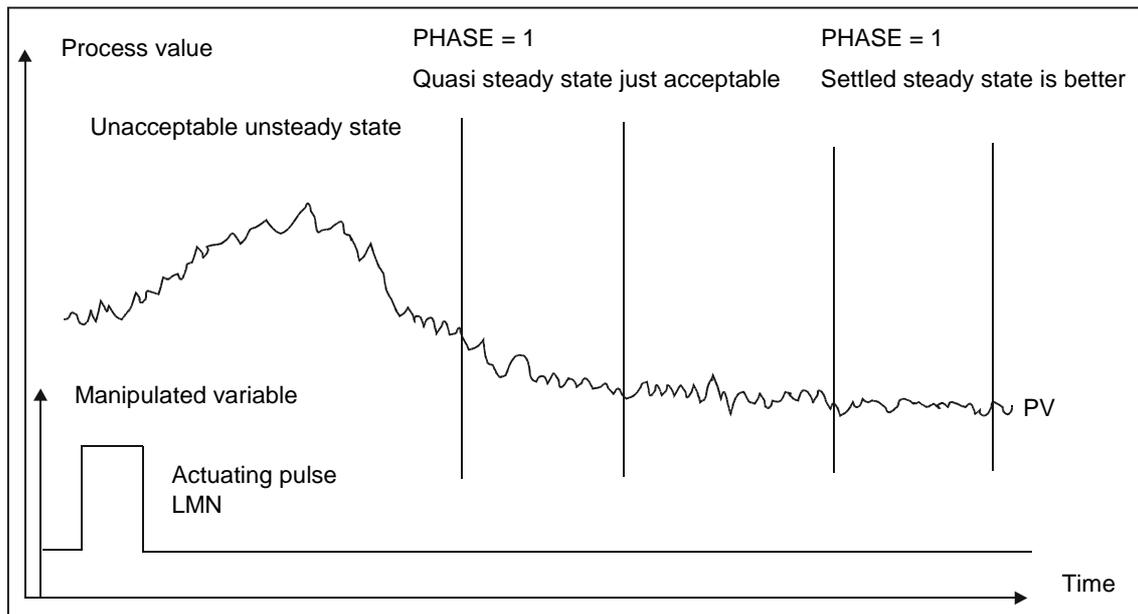
Preparing for Tuning (Phase 0 -> 1)

You can start the tuning both in manual or in automatic mode.

Set the parameter TUN_ON = TRUE. This makes FB 58 "TCONT_CP" ready for tuning (Phase 1). The TUN_ON bit must only be set in the steady state or during aperiodic settling to the steady state.

If the quasi steady state changes after setting the TUN_ON bit, this must be reset and the new quasi steady state must be signaled to FB 58 "TCONT_CP" by setting the TUN_ON bit again.

The schematic below illustrates how the process variable settles to the steady state:



In Phase 1, the time prior to making the manipulated variable step change is used by FB 58 "TCONT_CP" to calculate the process variable noise NOISE_PV, the initial rise PVDT0 and the mean value of the manipulated variable (initial value of the manipulated variable LMN0).

Note

You should only wait to excite the process in Phase 1 until the block has determined the mean value of the manipulated variable and the initial rise of the process variable (typically: 1 minute).

In Phase 1, both the controller sampling time CYCLE and the pulse generator sampling time CYCLE_P are measured and written to the relevant in/out parameters at the beginning of Phase 2. In the control mode without the pulse generator, CYCLE_P = CYCLE.

Note

If you call the pulse controller using SELECT = 0 or 1, you must specify the required ratio CYCLE/CYCLE_P with the parameters CYCLE and CYCLE_P before setting TUN_ON.

3.6 Starting Tuning (Phase 1 -> 2)

Tuning by Approaching the Operating Point with a Setpoint Step Change

The tuning manipulated variable (LMN0 + TUN_DLMN) is applied by changing the setpoint (transition Phase 1 -> 2). The setpoint, however, becomes effective only when the point of inflection is reached (the controller switches to automatic only when this point is reached).

The user is responsible for selecting the magnitude of the manipulated variable change (TUN_DLMN) according to the permitted process variable change. The sign of TUN_DLMN must be set depending on the intended process variable change (take into account the direction in which the control is operating).

The setpoint step change and TUN_DLMN must be suitably matched. If TUN_DLMN is too high, there is a danger that the point of inflection will not be found within 75% of the setpoint step change.

TUN_DLMN must, however, be high enough so that the process variable reaches at least 22% of the setpoint step change. Otherwise, you would remain in the tuning mode (Phase 2).

Remedy: Reduce the setpoint while the tuning function is attempting to detect the point of inflection.

Note

With extremely sluggish processes, it is advisable to set a somewhat lower target setpoint than the desired operating point during tuning and to monitor the status bits and PV (risk of overshoot).

Tuning only in the linear range:

Certain controlled processes (for example zinc and magnesium smelters) have a non-linear range shortly before the operating point (change in the state of the material).

By selecting a suitable setpoint step change, the tuning can be limited to the linear range. When the process variable has passed 75% of the setpoint step change (SP_INT-PV0), tuning is terminated.

At the same time, TUN_DLMN should be reduced so that the point of inflection is guaranteed to be found before reaching 75% of the setpoint step change.

Tuning at the Operating Point without Setpoint Step Change

The tuning manipulated variable ($LMN0 + TUN_DLMN$) is applied by setting the start bit TUN_ST (transition Phase 1 \rightarrow 2). When you change the setpoint, the new setpoint takes effect only when the point of inflection is reached (this is when the controller switches to automatic).

The user is responsible for selecting the magnitude of the manipulated variable change (TUN_DLMN) according to the permitted process variable change. The sign of TUN_DLMN must be set depending on the intended process variable change (take into account the direction in which the control is operating).

Caution!

If you excite the process with TUN_ST , there is no safety turn off at 75%. Tuning is terminated when the point of inflection is reached. In noisy processes, the point of inflection can, however, be significantly exceeded.

Protection Against Operator Input Errors

Operator Error	STATUS and Result	Comment
Setting TUN_ON and setpoint step change or TUN_ST at the same time	Transition to Phase 1, however tuning is not started. <ul style="list-style-type: none"> $SP_INT = SPold$ or $TUN_ST = FALSE$ 	The setpoint change is canceled. This prevents the controller controlling to the new setpoint and leaving the steady state operating point unnecessarily.
Effective $TUN_DLMN < 5\%$ (end of Phase 1)	$STATUS_H = 30002$ <ul style="list-style-type: none"> Transition to Phase 0 $TUN_ON = FALSE$ $SP = SPold$ 	Tuning is aborted. The setpoint change is canceled. This prevents the controller controlling to the new setpoint and leaving the steady state operating point unnecessarily.

3.7 Searching for the Point of Inflection (Phase 2) and Calculating the Control Parameters (Phase 3, 4, 5)

In Phase 2, the tuning function attempts to detect the point of inflection with the manipulated variable remaining constant. This method prevents the point of inflection 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 is always 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 is, however, only terminated when the following two conditions are met:

1. The process variable is more than $2 \cdot \text{NOISE_PV}$ away from the point of inflection.
2. The process variable has exceeded the point of inflection by 20%.

Note

When exciting the process using a setpoint step change, the tuning is terminated at the latest when the process variable exceeds 75% of the setpoint step change ($\text{SP_INT} - \text{PV0}$) (see below).

Phases 3, 4 and 5 are then run through once. The process type is then checked in Phase 7. The tuning mode is then terminated and FB 58 "TCONT_CP" is once again in Phase 0. The controller now always starts in the automatic mode with $\text{LMN} = \text{LMN0} + 0.75 \cdot \text{TUN_DLMN}$ (even if you were controlling in manual mode prior to the tuning).

3.8 Checking the Process Type (Phase 7)

Phase 7 therefore checks whether or not the process type is correct. This check is made **in automatic mode** with the controller parameters that have just been calculated and is completed at least $0.35 \cdot \text{TA}$ (recovery time) after the point of inflection. If the process order highly differs from the estimated value, the controller parameters will be recalculated and STATUS_D will be increased by 1 otherwise, the controller parameters remain unchanged.

Note

If Phase 7 is aborted by $\text{TUN_ON} = \text{FALSE}$, the controller parameters that have already been obtained are retained!

3.9 Result of the Tuning

The left digit of STATUS_H indicates the tuning status (for a detailed table, see Appendix A.4, Page A-22):

STATUS_H	Result:
0	Default or no new controller parameters have been found (yet).
10000	Suitable control parameters found
2xxxx	Control parameters found using estimated values; check the control response or check the STATUS_H diagnostic message and repeat the controller tuning.
3xxxx	Operator input error occurred; check the STATUS_H diagnostic message and repeat the controller tuning.

The following control parameters are updated in FB 58 "TCONT_CP":

- Factor for attenuating the P-action PFAC_SP = 0.8
- Controller gain GAIN
- Integral time TI
- Derivative time TD
- Derivative factor D_F = 5.0
- Control zone on/off CONZ_ON
- Control zone width CON_ZONE

The control zone is activated only if the process type is suitable (process type I and II) and a PID controller is being 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 reactivated with UNDO_PAR. A PI and PID parameter set is also saved 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.

The CYCLE and CYCLE_P sampling times were already checked in Phase 1.

3.10 Tuning Stopped by the Operator

Stopping Tuning Before Completion

In Phase 1, 2 or 3, you can stop the tuning by resetting TUN_ON to FALSE without new parameters being calculated. The controller starts in automatic mode with $LMN = LMN0 + TUN_DLMN$. If the controller was in manual mode prior to the tuning, the old manual value will be output.

If the tuning is stopped during Phase 4, 5 or 7 with TUN_ON = FALSE, the control parameters calculated to this point are retained.

3.11 Error Situations and Remedies

Point of inflection not reached (only with excitation by setpoint step change)

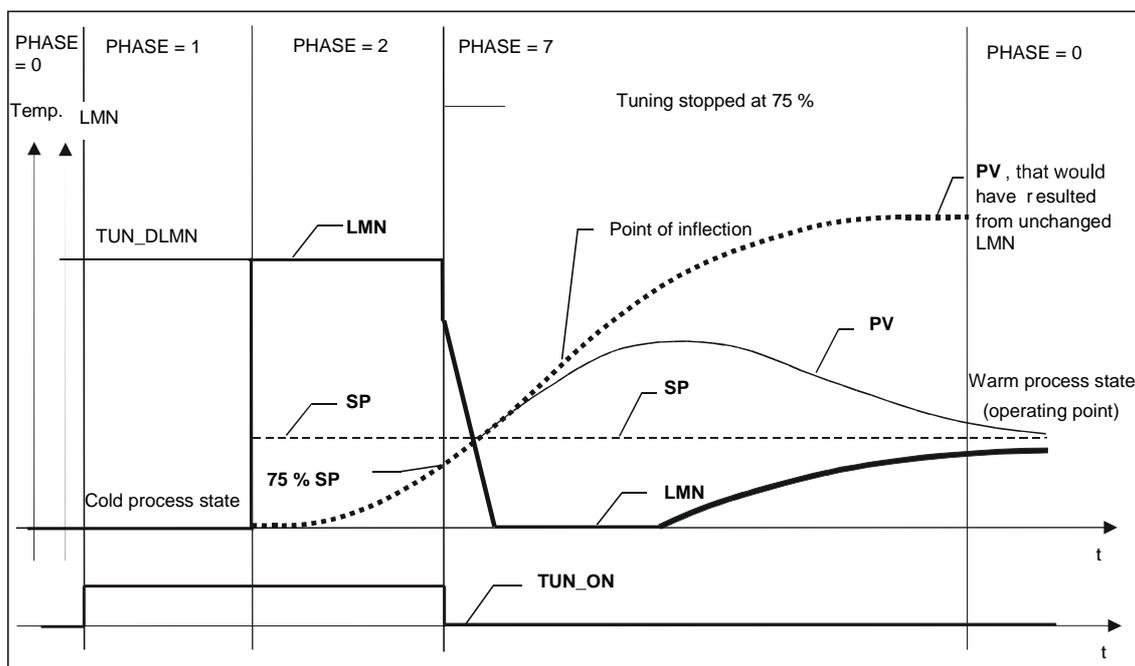
Tuning is terminated at the latest when the process value has exceeded 75% of the setpoint step change (SP-INT-PV0). This is signaled by "inflection point not reached" in STATUS_H (2xx2x).

The currently valid setpoint always applies. By reducing the setpoint, it is possible to achieve an earlier termination of the tuning function.

In typical temperature processes, terminating the tuning at 75% of the setpoint step change is normally adequate to prevent overshoot. In processes with a greater lag ($TU/TA > 0.1$, process type III) **caution** is advised. If the excitation of the manipulated variable is too high compared with the setpoint step change, the process variable can overshoot considerably (up to factor 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 to this, the control parameters are too aggressive. You should then weaken the controller parameters and repeat the attempt.

The schematic below illustrates the overshoot of the process variable when the excitation is too strong (process type III):



In typical temperature processes, aborting 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.

Errors estimating the lag or order

The lag (STATUS_H = 2x1xx or 2x3xx) or the order (STATUS_H = 21xxx or 22xxx) could not be obtained correctly. The tuning then continues with an estimated value that cannot lead to optimum controller parameters.

Repeat the tuning and make sure that there is no disturbance of the process variable.

Note

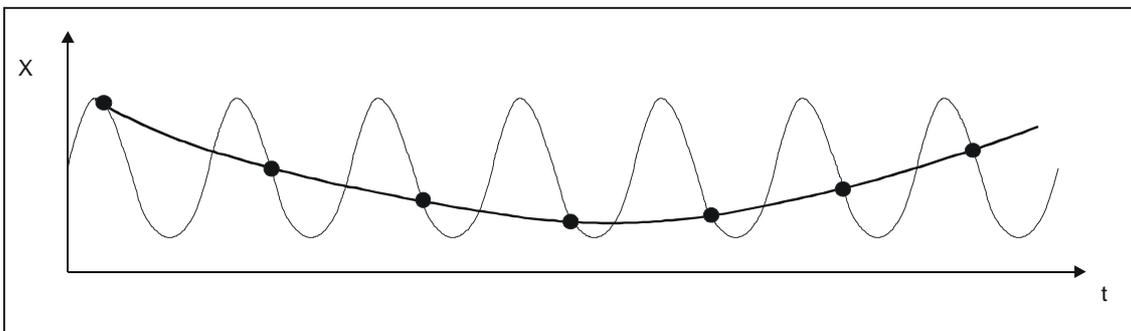
The special situation of a purely PT1 process is also indicated by STATUS_H = 2x1xx (TU <= 3*CYCLE). It is then not necessary to repeat the experiment. Weaken the controller parameters if the control oscillates.

Quality of the measuring signals (measurement noise, low-frequency interference)

The results of the 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 the noise period, the process variable 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 block 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 a software block. This should be filtered earlier in the measuring sensor to prevent the aliasing effect.

The schematic below illustrates the aliasing effect when the sampling time is too high:



- With low-frequency interference, it is relatively easy to ensure an adequately high sampling rate. On the other hand, 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 so that the accuracy of the tuning is adversely affected. Adequate accuracy is guaranteed with at least 40 noise periods 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"> Excitation by a manipulated variable change too high compared with the setpoint step change (see above). PI controller activated by PID_ON = FALSE. 	<ul style="list-style-type: none"> Increase the setpoint step change or reduce the manipulated variable step change. If the process permits a PID controller, start the tuning with PID_ON = TRUE.
Tuning in Phase 7	Initially, less aggressive controller parameters are obtained (process type III) that can lead to overshoot in Phase 7.	-
Control mode	PI controller and with PFAC_SP = 1.0 for process type I.	If the process permits a PID controller, start the tuning with PID_ON = TRUE.

3.12 Manual Fine Tuning in Control Mode

To achieve a setpoint response free of overshoot, you can take the measures described below:

Adapting the Control Zone

During the tuning, a control zone CON_ZONE is calculated by FB58 "TCONT_CP" that is activated (CONZ_ON = TRUE) if the process type is suitable (process type I and II) and a PID controller is being used. During control, you can modify the control zone or turn it off completely (with CONZ_ON = FALSE).

Note

Activating the control zone with higher order processes (process type III) does not normally bring 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 activate the control zone manually, make sure that the control zone width is not too small. If the control zone band is too small, oscillations will occur in the manipulated variable and process variable.

Continuously Weakening the Control Response with PFAC_SP

You can weaken the control response with the PFAC_SP parameter. This parameter specifies the amount of P-action that is effective in setpoint step changes.

Regardless of the process type, PFAC_SP is set to the default 0.8 by the tuning function, you can then modify the 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 (high lag)
PI	0.8	0.82	0.8
PID	0.6	0.75	0.96

Adapt the default factor (0.8) particularly in the following situations:

- Process type I with PID (0.8 -> 0.6): Setpoint step changes within the control zone 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 damped too strongly. This leads to a significantly slower response.

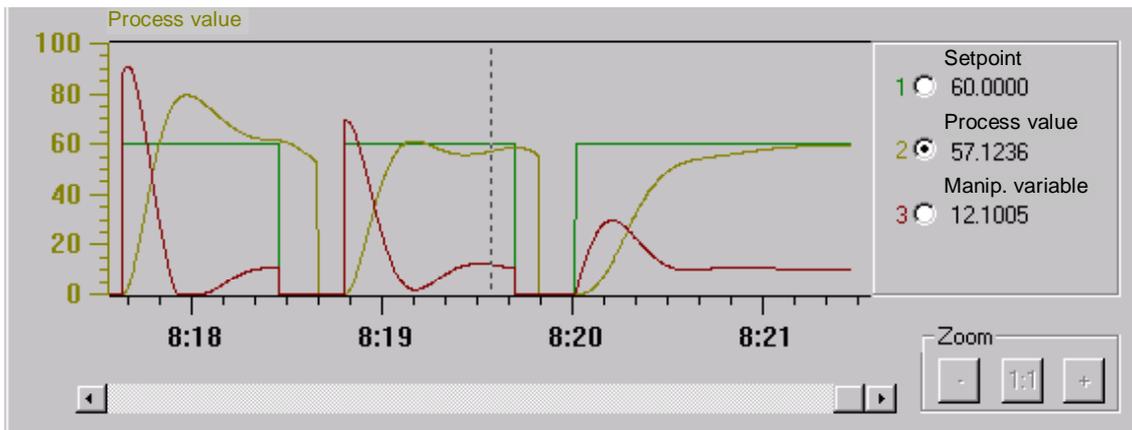
Example of Weakening the Control Response with PFAC_SP

Process parameters:

- GAIN = 6
- T1 = 50 s
- T2 = 5 s;

Controller parameters:

- GAIN = 1.45
- TI = 19.6 s



The figure below shows three attempts each with a setpoint step change from 0 to 60:

Attempt	PFAC_SP	Comment	Overshoot
Left 8:18	1.0	No P-action in the feedback; undamped control response	32 %
Middle 8:19	0.8	20% P-action in the feedback; optimum control response	2 %
Right 8:20	0.0	Complete P-action in the feedback; too strongly damped, long transient response	-

Damping Control Parameters

If oscillations occur in the closed control loop or if there is overshoot following setpoint step changes, you can reduce the controller GAIN (for example to 80 % of the original value) and increase the reset time TI (for example to 150 % of the original value). If the analog manipulated variable (LMN) of the continuous controller is converted to binary actuating signals with a pulse generator, quantization effects can cause small permanent oscillations. You can eliminate these by extending the controller deadband DEADB_W.

Changing the Control Parameters

To change the control parameters, follow the steps outlined below:

1. Save the current parameters with SAVE_PAR.
2. Change the parameter settings.
3. Test the control response.

If the new parameter settings are worse than the old settings, reload the old parameters with UNDO_PAR.

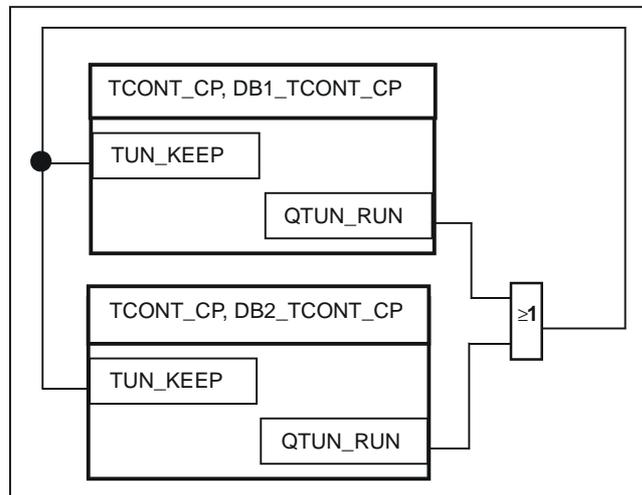
3.13 Parallel Tuning of Control Channels

Neighboring Zones (Strong Heat Coupling)

If two or more controllers control the temperature, for example, on a plate (in other words, there are two heaters and two measured process values with strong heat coupling), follow the steps outlined below:

1. OR the two outputs QTUN_RUN.
2. Interconnect the two TUN_KEEP inputs with the output of the OR element.
3. Start both controllers by setting a setpoint step change at the same time or by setting TUN_ST at the same time.

The schematic below illustrates the parallel tuning of controller channels.



Advantage:

Both controllers output LMNO + TUN_DLMN until both controllers have left Phase 2. This avoids that the controller that completes tuning first falsifies the tuning result of the other controller due to the change in its manipulated variable.

Caution!

Reaching 75% of the setpoint step change means that the tuning leaves Phase 2 and that the output QTUN_RUN is reset. Automatic mode starts, however only when TUN_KEEP also changes to 0.

Neighboring Zones (Weak Heat Coupling)

As a general rule, tuning should be carried out to reflect the way in which the control will work later. If zones are operated together during production so that the temperature differences between the zones remain the same, the temperature of the neighboring zones ought to be increased as well during tuning.

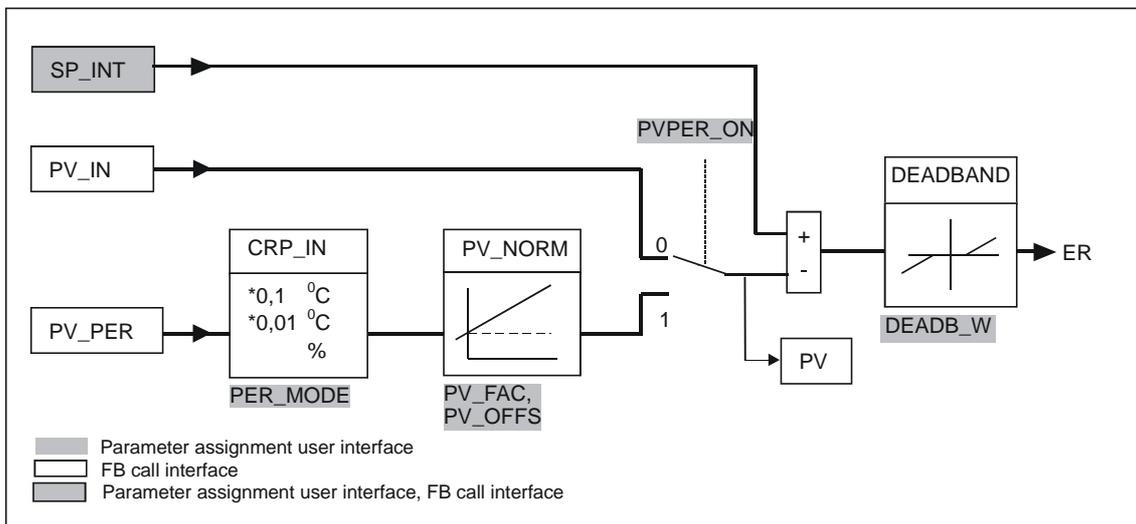
Differences in temperature at the beginning of the tuning are irrelevant since they will be compensated by the initial heating (-> initial rise = 0).

4 Temperature Step Controller FB59 "TCONT_S"

4.1 Controller Section

4.1.1 Forming the Error

Block Diagram



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 error must have the same unit.

Process Value Options (PVPER_ON)

Depending on PVPER_ON, the process value can be acquired in the peripheral (I/O) or floating-point format.

PVPER_ON	Process Value Input
TRUE	The process value is read in via the analog peripheral 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 peripheral value PV_PER to a floating-point format depending on the switch PER_MODE 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 Normalization PV_NORM (PF_FAC, PV_OFFS)

The PV_NORM function calculates the output of CRP_IN according to the following rule:

$$\text{"Output of PV_NORM"} = \text{"Output of CPR_IN"} * PV_FAC + PV_OFFS$$

This can be used for the following purposes:

- Process value correction with PV_FAC as the process value factor and PV_OFFS as the 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:

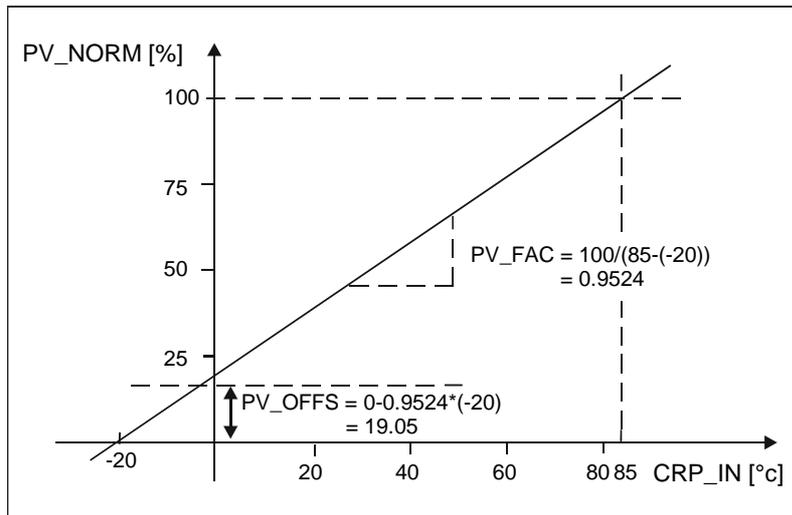
- $PV_FAC = \text{range of } PV_NORM / \text{range of } CRP_IN;$
- $PV_OFFS = LL(PV_NORM) - PV_FAC * LL(CRP_IN);$
where LL is the lower limit

With the default values ($PV_FAC = 1.0$ and $PV_OFFS = 0.0$), normalization is disabled. The effective process value is output at the PV output.

Example of Process Variable 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 schematic below shows the adaptation of the temperature range from -20 to 85°C to an internal scale of 0 to 100 %:



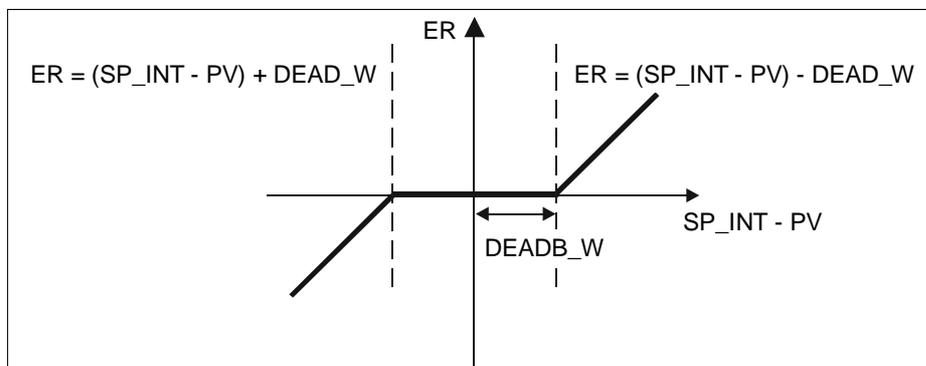
Forming the Error

The difference between the setpoint and process value is the error before the deadband.

The setpoint and process value must exist in the same unit.

Deadband (DEADB_W)

To suppress a small constant oscillation due to the manipulated variable quantization (for example in pulse duration modulation with PULSEGEN) a deadband (DEADBAND) is applied to the error. If DEADB_W = 0.0, the deadband is deactivated.



4.1.2 PI Step Controller Algorithm

FB 59 "TCONT_S" operates without a position feedback signal (see block diagram in Section 4.2, Page 4-5). 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 generator (PULSEOUT) that forms the pulses for the valve. Adapting the response threshold of the three-step element reduces the switching frequency of the controller.

Weakening the P-action when setpoint changes occur (PFAC_SP)

To prevent overshoot, you can attenuate 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

A value for PFAC_SP < 1.0 can reduce the overshoot as with the continuous controller if the motor run time MTR_TM is small compared with the recovery time TA and the ratio TU/TA is < 0.2. If MTR_TM reaches 20 % of TA, only a slight improvement can be achieved.

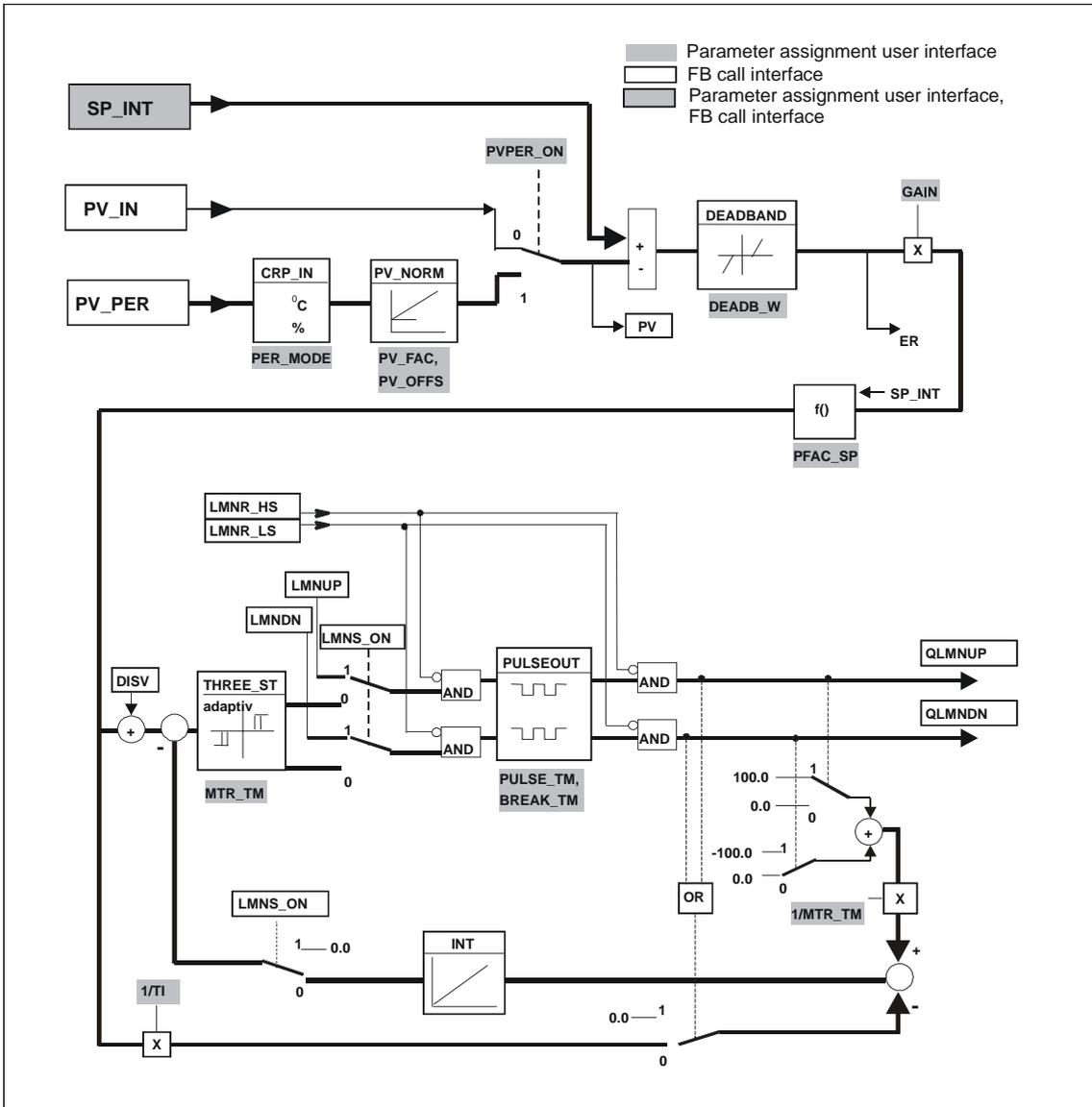
Feedforward Control

A feedforward 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 and the integrator (INT) are 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 error 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.

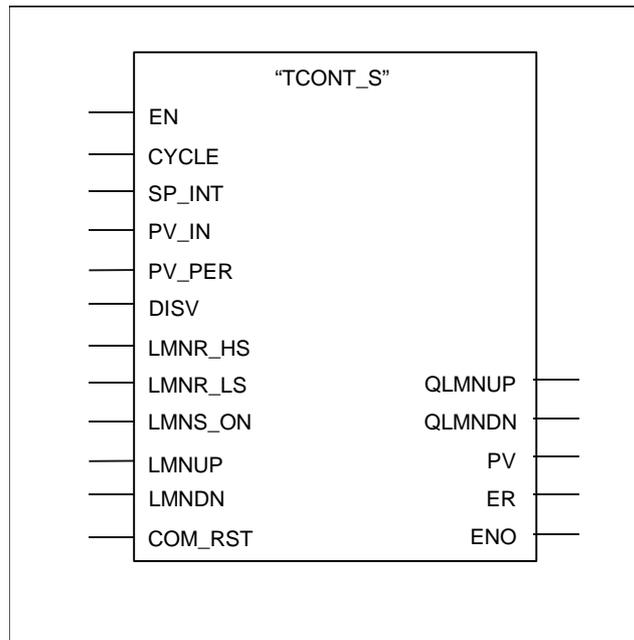
4.2 Block Diagram



4.3 Including the Function Block in the User Program

4.3.1 Calling the Controller Block

The following diagram shows the controller call in FBD:



FB TCONT_S must be called at constant intervals. To achieve this, use a cyclic interrupt OB (for example OB35 for an S7-300). The block interface provides the most important parameters that allow you to interconnect the block with process variables such as the setpoint, process value and manipulated variable. You can also connect the manual value signals or disturbance variable directly to the block interface.

4.3.2 Sampling Time

You specify the sampling time at the CYCLE parameter. You can also enter the sampling time using the parameter assignment tool. The sampling time CYCLE must match the time difference between two calls (cycle time of the cyclic OB including scan rates).

Rule of Thumb for the Controller Sampling Time CYCLE

The controller sampling time should not exceed 10 % of the calculated integral 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 (see numeric example below).

Numeric Example

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.

4.3.3 Initialization

FB "TCONT_S" has an initialization routine that is processed when the input parameter COM_RST = TRUE is set. After processing 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.

5 Getting Started

Aims

Based on the following simple example, "zEn01_13_STEP7__PID-Temp -> Pulse Controller", you will learn to control the simulated temperature process with the FB58 "TCONT_CP" temperature controller and to obtain the PID controller parameters online.

Requirements

The following requirements must be met:

- You are using an S7-300/400 station consisting of a power supply and a CPU.
- STEP 7 (≥ V5.1 SP3) is installed on your programming device.
- The programming device is connected to the CPU.

Creating a new Project and Copying the Example

Step	Activity	Result:
1	Create a project in the SIMATIC Manager with File->New...	The project window appears in the SIMATIC Manager.
2	Insert a SIMATIC 300 or 400 station to match your hardware configuration.	
3	Configure your station in HW Config and set the cycle time of the cyclic interrupt priority class of OB35 to 20 ms.	
4	Copy the pulse controller program from the sample project zEn01_13_STEP7__PID-Temp to your station.	The program is ready for downloading to the CPU.
5	Select your program and copy it to the CPU with PLC -> Download.	

Controller Tuning with the Parameter Assignment User Interface

Step	Activity	Result:
1	Open the parameter assignment tool by double-clicking on the instance DB DB_TCONT_CP in the SIMATIC Manager.	The parameter assignment tool opens.
2	Under Options, select the menu command Controller Tuning.	The curve recorder and the first dialog of the wizard open.
3	On the curve recorder, check that the manipulated variable and process value have practically settled and click on Next.	The "Selecting the Controller Type" dialog opens.
4	Set "PID parameters" and click on Next.	The "Selecting the Type of Process Excitation" dialog opens.
5	Set "Tune by approaching the operating point with a setpoint step change" and click on Next.	The "Process Excitation" dialog opens.
6	Set the operating point to 70 and the manipulated variable difference to 80 and click on Next.	The "Status and Result of the Tuning" dialog opens.
7	When the completion of the controller tuning is displayed, click on Close.	The wizard and curve recorder are closed.

You can now test these controller parameters by applying a setpoint step change or a disturbance load to the process.

Applying a Setpoint Step Change

Step	Activity	Result:
1	Open the curve recorder in the Options menu.	The curve recorder window opens.
2	Open the Commissioning dialog in the Options menu.	The Commissioning dialog opens.
3	Enter a setpoint step change to 90 for the setpoint parameter and click the Send button.	The setpoint changes abruptly in the curve recorder.
4	Observe the settling response of the process value and manipulated variable.	

Applying a Disturbance Load to the Process

Step	Activity	Result:
1	Open the VAT_LoopControl variable table in the SIMATIC Manager.	Variable table is opened.
2	Enter a process disturbance of 30 at the "DB_PROC_P".DISV parameter.	The curve of the process value changes in the curve recorder.
3	Observe the settling response of the process value and manipulated variable.	

Manual/Automatic Switchover

Step	Activity	Result:
1	Switch to manual in the Commissioning dialog and click the Send button.	The value of the manipulated variable remains constant in the curve recorder.
2	Set a different value for the manual value and click the Send button.	You can now see the new manual value set for the manipulated variable.
3	Return to the automatic mode and click the Send button.	Based on the value of the manipulated variable in the curve recorder, you can see how the controller operates again in automatic mode.

Changing Between PID and PI Parameters

Step	Activity	Result:
1	Switch to manual in the Commissioning dialog and click the Send button.	The value of the manipulated variable remains constant in the curve recorder.
2	Open the VAT_StructPar variable table in the SIMATIC Manager and click on Variable Monitor.	
3	Under "PID/PI Parameter Setting" in the Commissioning dialog, select PI parameters and click the Download button.	In the VAT_StructPar variable table, you can see how the parameters of PI_CON were transferred to the effective parameters.
4	Under "PID/PI Parameter Setting", select PID parameters and click the Download button.	In the VAT_StructPar variable table, you can see how the parameters of PID_CON were transferred to the effective parameters.
5	Return to the automatic mode and click the Send button.	Based on the value of the manipulated variable in the curve recorder, you can see how the controller operates again in automatic mode.

Uploading and Saving the Controller Parameters

Step	Activity	Result:
1	Switch to manual in the Commissioning dialog and click the Send button.	The value of the manipulated variable remains constant in the curve recorder.
2	Open the VAT_StructPar variable table in the SIMATIC Manager and click on Variable Monitor.	
3	Click the Download button for the "Saved PID and control zone parameters" option.	In the VAT_StructPar variable table, you can see how the parameters of PAR_SAVE were transferred to the effective parameters.
4	Change values in the effective parameters to be able to recognize later that the values are transferred.	
5	Click the Save button for the "PID and control zone parameters" option.	In the VAT_StructPar variable table, you can see how the effective parameters are transferred to PAR_SAVE.
6	Return to the automatic mode and click the Send button.	Based on the value of the manipulated variable in the curve recorder, you can see how the controller operates again in automatic mode.

6 Examples for the Temperature Controllers

6.1 Introduction

Overview

This chapter contains executable application samples for the temperature controllers FB 58 "TCONT_CP" and FB 59 "TCONT_S" with simulation of the process.

You will find the examples in the folder ...**STEP7EXAMPLES**.

Requirements

- You have set up and wired an S7 station consisting of a power supply and a CPU.
- STEP 7 (>= V5.1 + Service Pack 3) is installed on your programming device.
- The programming device is connected to the CPU.

Preparing the Samples

1. Open the sample project **zEn01_13_STEP7__PID-Temp** in the ...**STEP7EXAMPLES** folder with the SIMATIC Manager and copy it to your project folder with a suitable name (**File > Save As**). Use the **View > Details** option to display all the information.
2. Insert a station in your project to match your hardware configuration.
3. Select a sample program and copy the program to the station.
4. Configure the hardware with **HW Config**.
5. Save the hardware configuration and download it to the CPU.
6. Download the block folder to the CPU.

Code of the Samples

The samples are written in STL. You can view them directly in the LAD/STL/FBD editor. In this editor, select **View > Display with** "Symbolic Representation", "Symbol Selection" and "Comment". If you have enough space on the screen, you can also display the "Symbol Information".

Using a Sample

The sample programs include variable declaration tables (VAT) with which you can see and change the values. With the curve recorder in the parameter assignment user interface, you can also check the curves.

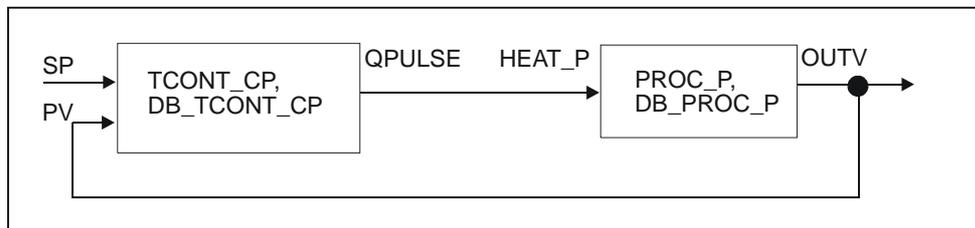
Adapting a Sample

You can use the code of the samples directly as a user program, however, the samples are not tuned for a real process.

6.2 Example with FB 58 "TCONT_CP" (pulse control)

The "pulse controller" sample contains a simple control loop with the FB 58 "TCONT_CP" temperature controller and temperature process simulated with PROC_P. The controller is set up as a pulse controller. PROC_P represents a 3rd order lag with a binary input.

The following schematic shows the control loop of the sample:



Program Structure

The controller and process block are called in OB35 with a cyclic interrupt time of 20 ms. The slower controller stage operates with CYCLE = 400 ms. To achieve the required accuracy, PER_TM > CYCLE was selected (1 s).

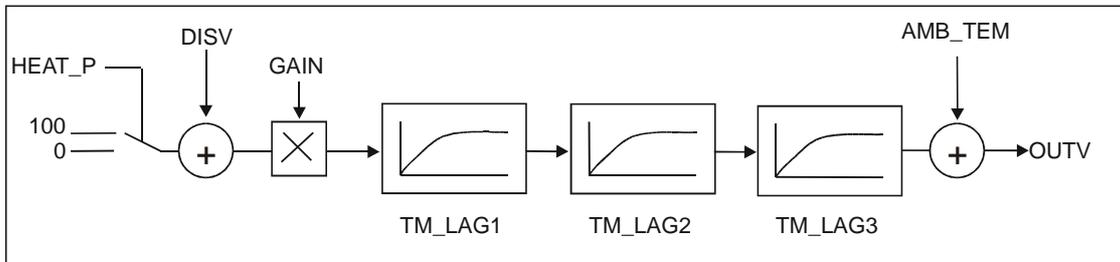
When OB100 starts, the restart bits of the controller and process are set.

The pulse generator for the controller is activated in OB100.

Process Block for Simulation of a Temperature Heating Zone

The block simulates a typical temperature process for heating that can occur as a control zone in an extruder, an injection molding machine, an annealing machine or as a separate furnace.

The following schematic shows the block diagram of the process PROC_P:



Parameter

Parameter	Comment	Description
HEAT_P	Heating pulse	Binary heating input signal
DISV	Disturbance variable	
GAIN	Process gain	
TM_LAG1	Time lag 1	
TM_LAG2	Time lag 2	
TM_LAG3	Time lag 3	
AMB_TEM	Ambient temperature	
OUTV	Output variable	temperature of the control zone

The binary input signals are converted to continuous floating-point values (0 or 100). After adding the disturbance variable and multiplying by the process gain, the process values pass through three 1st order time lag elements. Finally, the value of the ambient temperature is added. If the controller is initialized with COM_RST = TRUE, the output variable is set to $OUTV = DISV * GAIN + AMB_TEM$.

Operator Control and Monitoring

You can make your operator input in the VAT_LoopControl variable table.

The screenshot below shows the VAT_LoopControl variable table:

	Address	Symbol	Display format	Status value	Modify value
1		//Regler			
2	DB58.DBD 34	"DB_TCONT_CP".SP_INT	FLOATING_POINT	0.0	//70.0
3	DB58.DBD 14	"DB_TCONT_CP".PV	FLOATING_POINT	20.0	
4	DB58.DBD 18	"DB_TCONT_CP".LMN	FLOATING_POINT	0.0	
5	DB58.DBX 24.0	"DB_TCONT_CP".QPULSE	BOOL	false	
6	DB58.DBX 42.1	"DB_TCONT_CP".MAN_ON	BOOL	true	//true
7	DB58.DBD 38	"DB_TCONT_CP".MAN	FLOATING_POINT	0.0	//0.0
8		//Impulsformer			
9	DB58.DBD 26	"DB_TCONT_CP".CYCLE	FLOATING_POINT	0.4	//0.4
10	DB58.DBD 30	"DB_TCONT_CP".CYCLE_P	FLOATING_POINT	0.02	//0.02
11	DB58.DBD 76	"DB_TCONT_CP".PER_TM	FLOATING_POINT	1.0	//1.0
12	DB58.DBD 80	"DB_TCONT_CP".P_B_TM	FLOATING_POINT	0.0	
13		//Parameter			
14	DB58.DBD 162	"DB_TCONT_CP".PFAC_SP	FLOATING_POINT	1.0	//0.8
15	DB58.DBD 166	"DB_TCONT_CP".GAIN	FLOATING_POINT	32.4	//32.4
16	DB58.DBD 170	"DB_TCONT_CP".TI	FLOATING_POINT	6.63	//6.63
17	DB58.DBD 174	"DB_TCONT_CP".TD	FLOATING_POINT	1.65	//1.65
18	DB58.DBD 182	"DB_TCONT_CP".CON_ZONE	FLOATING_POINT	7.8	//7.8
19	DB58.DBX 186.0	"DB_TCONT_CP".CONZ_ON	BOOL	false	//true
20	DB58.DBX 186.6	"DB_TCONT_CP".PID_ON	BOOL	true	
21	DB58.DBX 186.5	"DB_TCONT_CP".LOAD_PID	BOOL	false	
22		//Optimierung			
23	DB58.DBX 186.1	"DB_TCONT_CP".TUN_ON	BOOL	false	//true
24	DB58.DBX 186.2	"DB_TCONT_CP".TUN_ST	BOOL	false	//true
25	DB58.DBD 84	"DB_TCONT_CP".TUN_DLMN	FLOATING_POINT	80.0	//-20.0
26	DB58.DBW 108	"DB_TCONT_CP".PHASE	DEC	0	
27	DB58.DBW 110	"DB_TCONT_CP".STATUS_H	DEC	0	
28	DB58.DBW 112	"DB_TCONT_CP".STATUS_D	DEC	0	
29		//Prozess			
30	DB102.DBD 26	"DB_PROC_P".CYCLE	FLOATING_POINT	0.02	//0.02
31	DB102.DBD 2	"DB_PROC_P".DISV	FLOATING_POINT	0.0	
32	DB102.DBD 6	"DB_PROC_P".AMB_TEM	FLOATING_POINT	20.0	
33	DB102.DBD 10	"DB_PROC_P".GAIN	FLOATING_POINT	1.0	//1.0
34	DB102.DBD 14	"DB_PROC_P".TM_LAG1	FLOATING_POINT	50.0	//50.0
35	DB102.DBD 18	"DB_PROC_P".TM_LAG2	FLOATING_POINT	5.0	//5.0
36	DB102.DBD 22	"DB_PROC_P".TM_LAG3	FLOATING_POINT	0.0	
37					

The controller can be changed to manual at the MAN_ON switch. The manual value can be set at MAN.

After a warm restart on the CPU, the controller is in manual with the heating off.

If you want to tune the controller, set the TUN_ON bit and enter a setpoint at SP. You can monitor the tuning at the PHASE parameter.

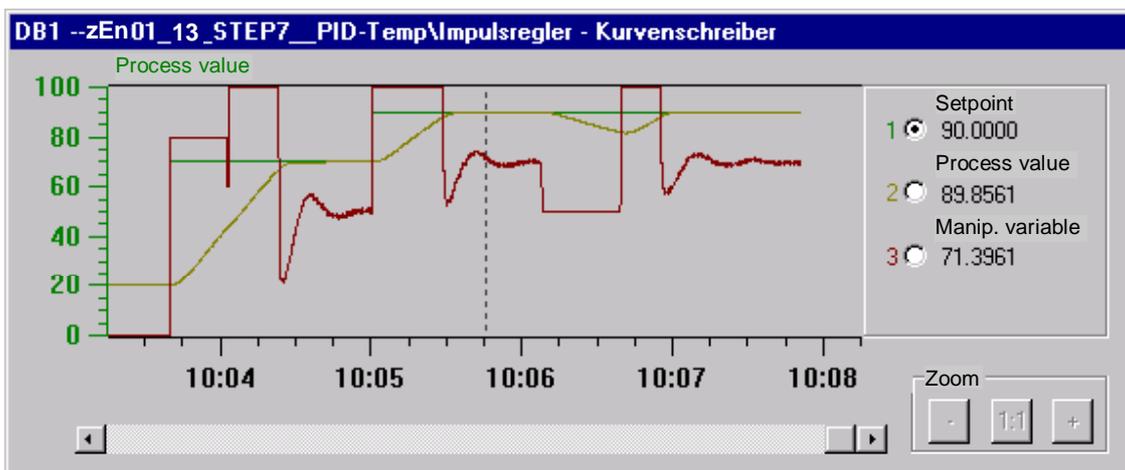
The result of the tuning can be seen at the status words STATUS_H and STATUS_D.

Putting the Sample into Operation

To put the sample into operation, follow the steps outlined below:

1. Copy the sample to a CPU.
2. In HW Config, set the cycle time of OB35 to 20 ms.
If a time error occurs in the cyclic interrupt level, you must extend the cycle time. In this case, the simulation will run more slowly. When you control the real process, the cycle time of OB35 and the sampling time CYCLE_P or CYCLE of DB_PROC_P must match.

The screenshot below illustrates controller tuning with FB58:



This screenshot shows the controller tuning when heating from ambient temperature 20°C to the operating point (70°C). Following this, a setpoint step change was controlled with a control zone. At the new operating point of 90°C, the controller was tuned again with negative manipulated variable excitation.

6.3 Samples for FB 58 "TCONT_CP" with Short Pulse Generator Sampling Time

The two samples described here are identical to the "pulse controller" sample described in Section 6.2. The only differences are in the call mechanism as described below.

The FB 58 "TCONT_CP" block contains a mechanism that allows the processing of the controller stage with its intensive calculations and the tuning to be shifted to OB1 or a slower cyclic interrupt OB (for example OB32: 1 s). You can use this mechanism when your CPU has a heavy load and you require high accuracy and therefore a reduction of CYCLE_P to CYCLE.

- The "pulse controller OB35, OB1" sample is suitable for S7-300 since only one cyclic interrupt level is available.

The following figures show the block call with a short pulse generator sampling time on an S7-300:

```

OB1 (free cycle)

A   "DB_TCONT_CP".QC_ACT
JCN M001

Call TCONT_CP, DB_TCONT_CP
    ...
    SELECT = 1,
    ...
M001: NOP 0
    
```

```

OB35 (e.g. 20 ms)

Call TCONT_CP, DB_TCONT_CP
    ...
    SELECT = 2,
    ...
    
```

- The "pulse controller OB35, OB32" sample is suitable for S7-400 since several cyclic interrupt levels are available.

The following figures show the block call with a short pulse generator sampling time on an S7-400:

```

OB32 (e.g. 1 s)

Call TCONT_CP, DB_TCONT_CP
    ...
    SELECT = 3,
    ...
    
```

```

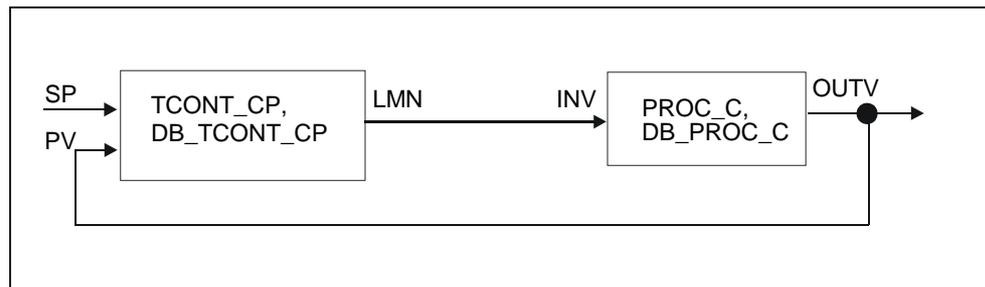
OB35 (e.g. 20 ms)

Call TCONT_CP, DB_TCONT_CP
    ...
    SELECT = 2,
    ...
    
```

6.4 Sample for FB 58 "TCONT_CP" (Continuous)

The "continuous controller" sample contains a simple control loop with the FB 58 "TCONT_CP" temperature controller and temperature process simulated with PROC_C. The controller is set as a continuous controller. PROC_C represents a 3rd order lag with an analog input.

The schematic below illustrates the control loop of the sample:



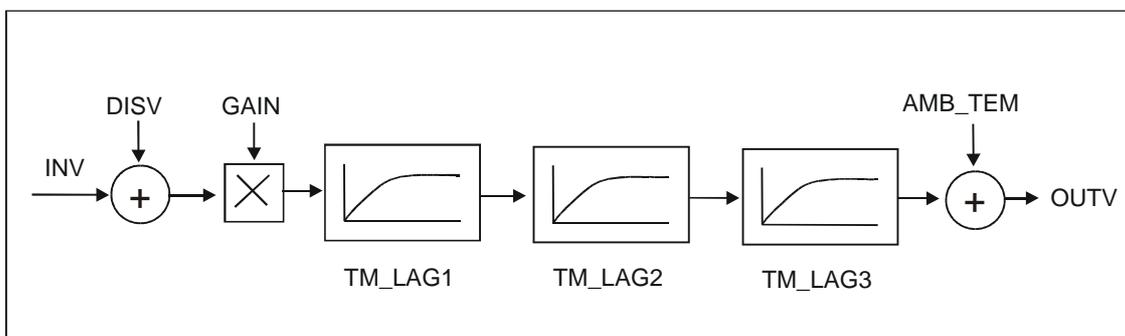
Program Structure

The controller and process block are called in OB35 with a cyclic interrupt time of 100 ms. When OB100 starts, the restart bits of the controller and process are set.

Process Block for Simulation of a Temperature Heating Zone

The block simulates a typical temperature process for heating that can occur as a control zone in an extruder, an injection molding machine, an annealing machine or as a separate furnace.

The schematic below is the block diagram of the controlled system PROC_C:



Parameter

Parameter	Comment	Description
INV	Input variable	Value of the manipulated variable of the controller
DISV	Disturbance variable	
GAIN	Process gain	
TM_LAG1	Time lag 1	
TM_LAG2	Time lag 2	
TM_LAG3	Time lag 3	
AMB_TEM	Ambient temperature	
OUTV	Output variable	Temperature of the control zone

After adding the analog input signal and a load (disturbance variable) and then multiplying with the process gain, the process values pass through three 1st order time lags. Finally, the value of the ambient temperature is added.

If the controller is initialized with `COM_RST = TRUE`, the output variable is set to $OUTV = (INV + DISV) * GAIN + AMB_TEM$.

Operator Control and Monitoring

You can make your operator input in the VAT_LoopControlC variable table.

	Address	Symbol	Display format	Status value	Modify value
1		//Regler			
2	DB58.DBD 34	"DB_TCONT_CP".SP_INT	FLOATING_POINT	0.0	//60.0
3	DB58.DBD 14	"DB_TCONT_CP".PV	FLOATING_POINT	20.0	
4	DB58.DBD 18	"DB_TCONT_CP".LMN	FLOATING_POINT	0.0	
5	DB58.DBX 42.1	"DB_TCONT_CP".MAN_ON	BOOL	true	//true
6	DB58.DBD 38	"DB_TCONT_CP".MAN	FLOATING_POINT	0.0	//0.0
7	DB58.DBD 26	"DB_TCONT_CP".CYCLE	FLOATING_POINT	0.1	
8		//Parameter			
9	DB58.DBD 162	"DB_TCONT_CP".PFAC_SP	FLOATING_POINT	1.0	//0.6
10	DB58.DBD 166	"DB_TCONT_CP".GAIN	FLOATING_POINT	6.48	//6.48
11	DB58.DBD 170	"DB_TCONT_CP".TI	FLOATING_POINT	3.16	//3.16
12	DB58.DBD 174	"DB_TCONT_CP".TD	FLOATING_POINT	0.79	//0.79
13	DB58.DBD 182	"DB_TCONT_CP".CON_ZONE	FLOATING_POINT	38.6	//38.6
14	DB58.DBX 186.0	"DB_TCONT_CP".CONZ_ON	BOOL	false	//true
15	DB58.DBX 186.6	"DB_TCONT_CP".PID_ON	BOOL	true	//true
16	DB58.DBX 186.5	"DB_TCONT_CP".LOAD_PID	BOOL	false	
17					
18		//Optimierung			
19	DB58.DBX 186.1	"DB_TCONT_CP".TUN_ON	BOOL	false	//true
20	DB58.DBX 186.2	"DB_TCONT_CP".TUN_ST	BOOL	false	//true
21	DB58.DBD 84	"DB_TCONT_CP".TUN_DLMN	FLOATING_POINT	20.0	//20.0
22					
23	DB58.DBW 108	"DB_TCONT_CP".PHASE	DEC	0	
24	DB58.DBW 110	"DB_TCONT_CP".STATUS_H	DEC	0	
25	DB58.DBW 112	"DB_TCONT_CP".STATUS_D	DEC	0	
26					
27		//Prozess			
28	DB100.DBD 28	"DB_PROC_C".CYCLE	FLOATING_POINT	0.1	
29	DB100.DBD 4	"DB_PROC_C".DISV	FLOATING_POINT	0.0	
30	DB100.DBD 12	"DB_PROC_C".GAIN	FLOATING_POINT	10.0	//10.0
31	DB100.DBD 16	"DB_PROC_C".TM_LAG1	FLOATING_POINT	50.0	//50.0
32	DB100.DBD 20	"DB_PROC_C".TM_LAG2	FLOATING_POINT	2.0	//2.0
33	DB100.DBD 24	"DB_PROC_C".TM_LAG3	FLOATING_POINT	0.0	
34	DB100.DBD 8	"DB_PROC_C".AMB_TEM	FLOATING_POINT	20.0	
35	DB100.DBX 36.0	"DB_PROC_C".COM_RST	BOOL	false	//true
36					

Ka_Test1\SIMATIC 400(1)\... \Kontinuierlicher Regler RUN

The controller can be changed to manual at the MAN_ON switch. The manual value can be set at MAN. After a warm restart on the CPU, the controller is in manual with the heating off.

If you want to tune the controller, set the TUN_ON bit and enter a setpoint at SP. You can monitor the tuning at the PHASE parameter.

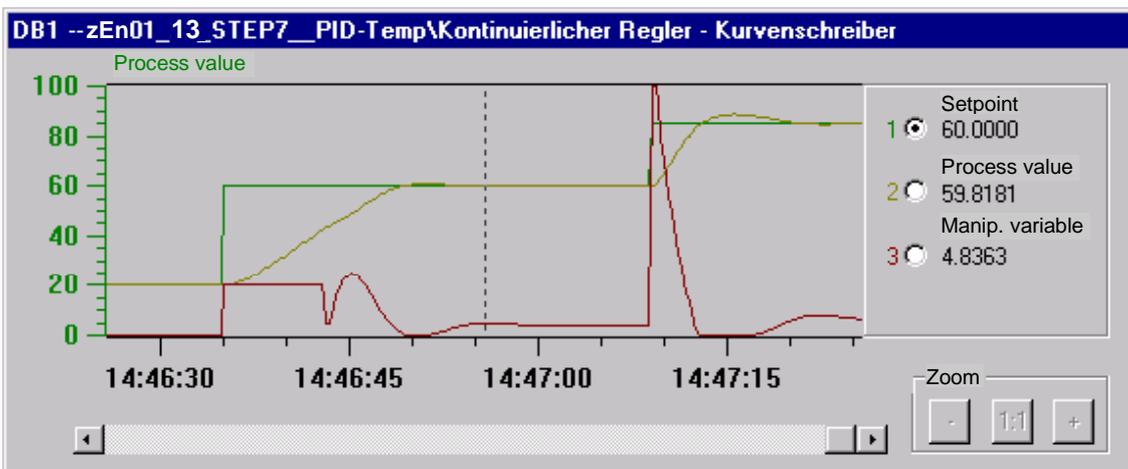
The result of the tuning can be seen at the status words STATUS_H and STATUS_D.

Putting the Sample into Operation

To put the sample into operation, follow the steps outlined below:

1. Copy the sample to a CPU.
2. If the default cycle time of OB35 (100 ms) no longer exists, set the cycle time of OB35 to 100 ms in HW Config. If a time error occurs in the cyclic interrupt priority class, you must extend the cycle time. In this case, the simulation will run more slowly. When you control the real process, the cycle time of OB35 and the sampling times CYCLE of DB_TCONT_CP and DB_PROC_C must match.
3. To run the controller tuning, set TUN_DLMN to 20%.

The screenshot below illustrates controller tuning with TCONT_CP:

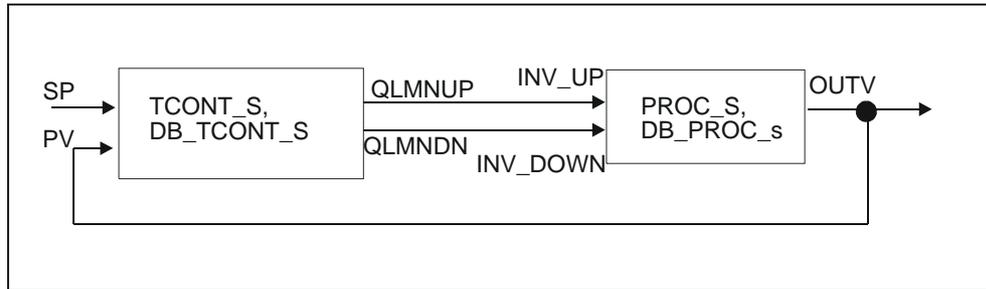


The screenshot above shows the controller tuning when heating from ambient temperature 20°C to the operating point (60°C). Following this, a setpoint step change from 60°C to 85°C is made within the control zone. The overshoot can be eliminated by reducing PFAC_SP from 0.8 to 0.6.

6.5 Sample for FB 59 "TCONT_S" (Step Controller)

The "step controller" sample contains a simple control loop consisting of a PI step controller and a third-order lag with an integrating actuator as the model for a temperature process.

The schematic below illustrates the control loop of the sample:



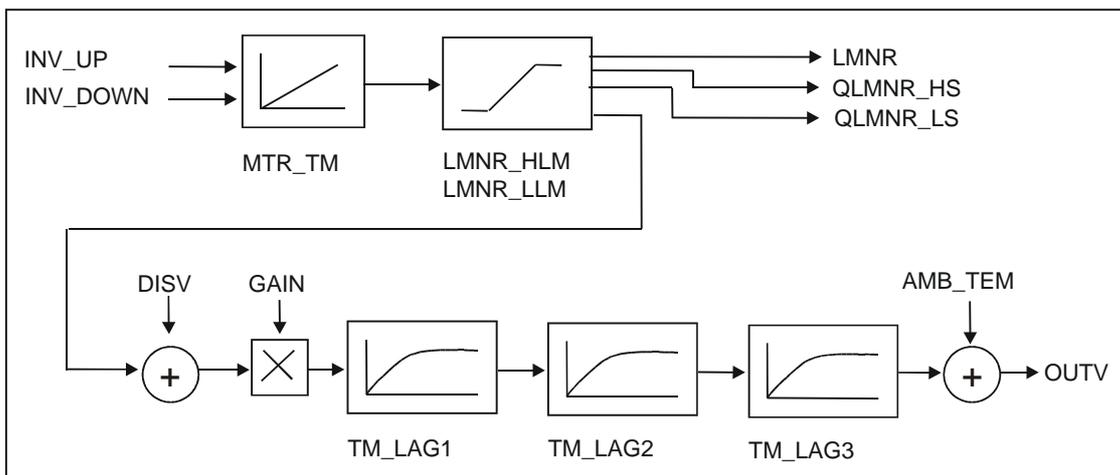
Program Structure

The controller and process are called in OB35. When OB100 starts, the restart bits of the controller and process are set.

Process Block for Simulation of a Temperature Process

The block simulates a process with a 3rd order time lag. For temperature processes select 2nd order time lag response with one large and one small time constant ($TM_LAG1 = 10 \times TM_LAG2$ and $TM_LAG3 = 0$ s).

The following schematic is the block diagram of the PROC_S controlled process with actuator:



Parameter

Parameter	Comment	Description
INV_UP	input variable up	
INV_DOWN	input variable down	
DISV	disturbance variable	
GAIN	process gain	
MTR_TM	motor actuating time	
LMNR_HLM	actuator value high limit	
LMNR_LLM	actuator value low limit	
TM_LAG1	time lag 1	
TM_LAG2	time lag 2	with temperature processes: TM_LAG1 = 10...100 × TM_LAG2)
TM_LAG3	time lag 3	= 0 with temperature processes
AMB_TEM	ambient temperature	
OUTV	output variable	temperature
LMNR	position feedback signal	
QLMNR_HS	high limit stop signal	
QLMNR_LS	low limit stop signal	

Depending on the input signals INV_UP and INV_DOWN, the position feedback signal LMNR is calculated by an integrator. The position feedback signal is limited to LMNR_HLM and LMNR_LLM. When the limit is reached, the limit stop signals QLMNR_HS or QLMNR_LS are set.

After adding a disturbance variable and then multiplying by the process gain, the process values pass through three 1st order time lag elements.

If the controller is initialized with COM_RST = TRUE, the output variable is set to $OUTV = (LMNR + DISV) * GAIN + AMB_TEM$.

Operator Control and Monitoring

You can make your operator input in the VAT_LoopControls variable table.

	Address	Symbol	Display format	Status value	Modify value
1		//Regler			
2	DB59.DBD 4	"DB_TCONT_S".SP_INT	FLOATING_POINT	20.0	//36.0
3	DB59.DBD 8	"DB_TCONT_S".PV_IN	FLOATING_POINT	20.0	
4		//Hand-Automatikumschaltung			
5	DB101.DBD 46	"DB_PROC_S".LMNR	FLOATING_POINT	0.0	//0.0
6	DB59.DBX 18.2	"DB_TCONT_S".LMNS_ON	BOOL	true	//false
7	DB59.DBX 18.3	"DB_TCONT_S".LMNUP	BOOL	false	//false
8	DB59.DBX 18.4	"DB_TCONT_S".LMNDN	BOOL	false	//false
9	DB59.DBX 30.0	"DB_TCONT_S".COM_RST	BOOL	false	//true
10		//Parameter			
11	DB59.DBD 44	"DB_TCONT_S".PFAC_SP	FLOATING_POINT	1.0	//0.7
12	DB59.DBD 48	"DB_TCONT_S".GAIN	FLOATING_POINT	5.8	//5.8
13	DB59.DBD 52	"DB_TCONT_S".TI	FLOATING_POINT	20.0	//20.0
14	DB59.DBD 56	"DB_TCONT_S".MTR_TM	FLOATING_POINT	2.0	//2.0
15	DB59.DBD 60	"DB_TCONT_S".PULSE_TM	FLOATING_POINT	0.0	//0.0
16	DB59.DBD 64	"DB_TCONT_S".BREAK_TM	FLOATING_POINT	0.0	//0.0
17	DB59.DBD 0	"DB_TCONT_S".CYCLE	FLOATING_POINT	0.02	//0.02
18		//Prozess			
19	DB101.DBD 2	"DB_PROC_S".CYCLE	FLOATING_POINT	0.02	//0.02
20	DB101.DBD 6	"DB_PROC_S".DISV	FLOATING_POINT	0.0	//0.0
21	DB101.DBD 10	"DB_PROC_S".AMB_TEM	FLOATING_POINT	20.0	//20.0
22	DB101.DBD 14	"DB_PROC_S".GAIN	FLOATING_POINT	1.5	//1.5
23	DB101.DBD 18	"DB_PROC_S".MTR_TM	FLOATING_POINT	2.0	//2.0
24	DB101.DBD 30	"DB_PROC_S".TM_LAG1	FLOATING_POINT	50.0	//50.0
25	DB101.DBD 34	"DB_PROC_S".TM_LAG2	FLOATING_POINT	5.0	//5.0
26	DB101.DBD 38	"DB_PROC_S".TM_LAG3	FLOATING_POINT	0.0	//0.0
27	DB101.DBX 52.0	"DB_PROC_S".COM_RST	BOOL	false	//true
28					

Ka_Test1\SIMATIC 400(1)\...\Schrittregler RUN

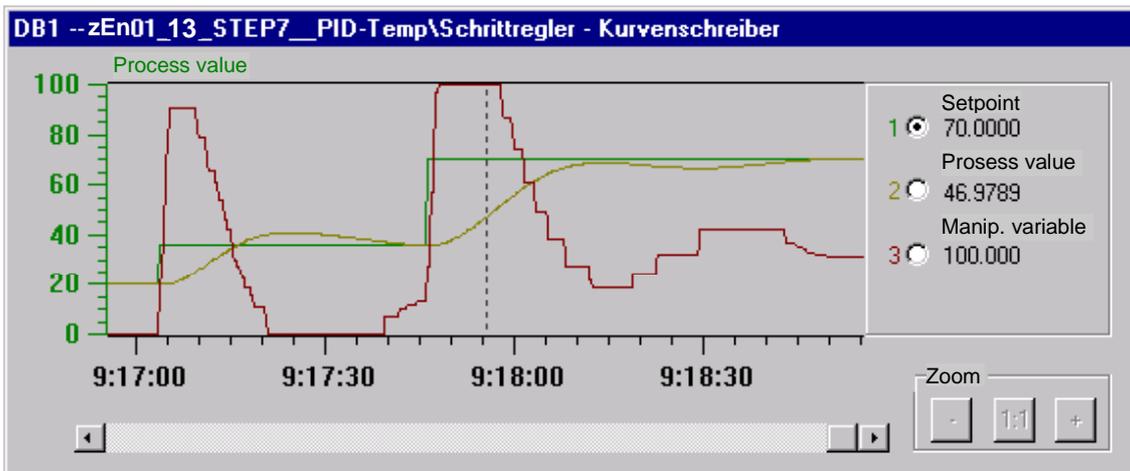
The controller can be changed to manual at the LMNS_ON switch. After a warm restart on the CPU, the controller is in manual. If LMNS_ON is set, the outputs QLMNUP or QLMNDN can be controlled at the inputs LMNUP or LMNDN in manual mode.

Putting the Sample into Operation

To put the sample into operation, follow the steps outlined below:

1. Copy the sample to a CPU.
2. In HW Config, set the cycle time of OB35 to 20 ms. If a time error occurs in the cyclic interrupt level, you must extend the cycle time. In this case, the simulation will run more slowly. When you control the real process, the cycle time of OB35 and the sampling time of FB 59 "TCONT_S" must match.

The screenshot below illustrates controller tuning with TCONT_S:



The schematic above first shows a setpoint step change from 20°C to 36°C. The manipulated variable limit is not reached, the temperature overshoots by approximately 5°C (30%). With the subsequent setpoint step change from 36°C to 70°C, the upper limit of the manipulated variable is reached. This avoids overshoot of the process variable.

If you want to avoid overshoot with small setpoint changes as well, you must reduce PFAC_SP (for example, from 1.0 to 0.8).

A Appendix

A.1 Technical Specifications

The following tables indicate the memory requirements of the temperature blocks:

Block Name	FB No.	Load Memory Req.	Work Memory Req.	Local Data
TCONT_CP	FB 58	10866 Bytes	9910 Bytes	144
TCONT_S	FB 59	2282 Bytes	1966 Bytes	64

Instance DB	Load Memory Req.	Work Memory Req.
Instance DB for TCONT_CP	1068 Bytes	532 Bytes
Instance DB for TCONT_S	298 Bytes	134 Bytes

A.2 Execution Times

Block Name	FB No.	Configuration	Execution Time (in ms) CPU 314	Execution Time (in ms) CPU 416
TCONT_CP	FB 58	Continuous controller with typical parameter settings	4.7	0.14
TCONT_CP	FB 58	Continuous controller with typical parameter settings + controller tuning	6.2	0.19
TCONT_CP	FB 58	Only pulse generator processed	0.87	0.025
TCONT_S	FB 59	Step controller with typical parameter settings	2.8	0.095

Measured with:

CPU 314: 6ES7 314-1AE84-0AB0; 0.3 ms/kAW

CPU 416: 6ES7 416-1XJ02-0AB0; 0.08 ms/kAW

A.3 DB Assignment

A.3.1 Instance DB for FB 58 "TCONT_CP"

Parameters:

Addr	Parameter	Decl.	Data Type	Range of values	Initial Value	Description
0.0	PV_IN	INPUT	REAL	Dependent on the sensors used	0.0	PROCESS VARIABLE IN An initialization value can be set at the "Process Variable In" input or an external process variable in floating-point format can be connected.
4.0	PV_PER	INPUT	INT		0	PROCESS VARIABLE PERIPHERY The process variable in the peripheral I/O format is connected to the controller at the "Process Variable Peripheral" input.
6.0	DISV	INPUT	REAL		0.0	DISTURBANCE VARIABLE For feedforward control, the disturbance variable is connected to the "Disturbance Variable" input.
10.0	INT_HPOS	INPUT	BOOL		FALSE	INTEGRAL ACTION HOLD IN POSITIVE DIRECTION The output of the integral action can be blocked in a positive direction. To achieve this, the INT_HPOS input must be set to TRUE. In a cascade control, the INT_HPOS of the primary controller is interconnected to QLMN_HLM of the secondary controller.
10.1	INT_HNEG	INPUT	BOOL		FALSE	INTEGRAL ACTION HOLD IN NEGATIVE DIRECTION The output of the integral action can be blocked in a negative direction. To achieve this, the INT_HNEG input must be set to TRUE. In a cascade control, the INT_HNEG of the primary controller is interconnected to QLMN_LLM of the secondary controller.

Addr	Parameter	Decl.	Data Type	Range of values	Initial Value	Description
12.0	SELECT	INPUT	INT	0 to 3	0	<p>SELECTION OF CALL PID AND PULSE GENERATOR</p> <p>If the pulse generator is activated, there are several ways of calling the PID algorithm and pulse generator:</p> <ul style="list-style-type: none"> • SELECT =0: The controller is called in a fast cyclic interrupt level and the PID algorithm and pulse generator are processed. • SELECT =1: The controller is called in OB1 and only the PID algorithm is processed. • SELECT =2: The controller is called in a fast cyclic interrupt level and only the pulse generator is processed. • SELECT =3: The controller is called in a slow cyclic interrupt level only the PID algorithm is processed.
14.0	PV	OUTPUT	REAL	Dependent on the sensors used	0.0	<p>PROCESS VARIABLE</p> <p>The effective process variable is output at the "Process Variable" output.</p>
18.0	LMN	OUTPUT	REAL		0.0	<p>MANIPULATED VARIABLE</p> <p>The effective value of the manipulated variable is output in floating-point format at the "Manipulated Variable" output.</p>
22.0	LMN_PER	OUTPUT	INT		0	<p>MANIPULATED VARIABLE PERIPHERY</p> <p>The value of the manipulated variable in the peripheral format is connected to the controller at the "Manipulated Variable Periphery" output.</p>
24.0	QPULSE	OUTPUT	BOOL		FALSE	<p>OUTPUT PULSE SIGNAL</p> <p>The value of the manipulated variable is output pulse duration modulated at the QPULSE output.</p>
24.1	QLMN_HLM	OUTPUT	BOOL		FALSE	<p>HIGH LIMIT OF MANIPULATED VARIABLE REACHED</p> <p>The value of the manipulated variable is always limited to an upper and lower limit. The QLMN_HLM output indicates when the upper limit is exceeded.</p>

Addr	Parameter	Decl.	Data Type	Range of values	Initial Value	Description
24.2	QLMN_LLM	OUTPUT	BOOL		FALSE	LOW LIMIT OF MANIPULATED VARIABLE REACHED The value of the manipulated variable is always limited to an upper and lower limit. The QLMN_LLM output indicates when the lower limit is exceeded.
24.3	QC_ACT	OUTPUT	BOOL		TRUE	NEXT CYCLE, THE CONTINUOUS CONTROLLER IS WORKING This parameter indicates whether or not the continuous controller stage will be executed at the next block call (relevant only when SELECT has the value 0 or 1).
26.0	CYCLE	INPUT/ OUTPUT	REAL	≥ 0.001 s	0.1 s	SAMPLE TIME OF CONTINUOUS CONTROLLER [s] This sets the sampling time for the PID algorithm. The tuner calculates the sampling time in Phase 1 and enters this in CYCLE.
30.0	CYCLE_P	INPUT/ OUTPUT	REAL	≥ 0.001 s	0.02 s	SAMPLE TIME OF PULSE GENERATOR [s] At this input, you enter the sampling time for the pulse generator stage. FB 58 "TCONT_CP" calculates the sampling time in Phase 1 and enters it in CYCLE_P.
34.0	SP_INT	INPUT/ OUTPUT	REAL	Dependent on the sensors used	0.0	INTERNAL SETPOINT The "Internal Setpoint" input is used to specify a setpoint.
38.0	MAN	INPUT/ OUTPUT	REAL		0.0	MANUAL VALUE The "Manual Value" input is used to specify a manual value. In automatic mode, it is corrected to the manipulated variable.
42.0	COM_RST	INPUT/ OUTPUT	BOOL		FALSE	COMPLETE RESTART The block has an initialization routine that is processed when the COM_RST input is set.
42.1	MAN_ON	INPUT/ OUTPUT	BOOL		TRUE	MANUAL OPERATION ON If the "Manual Operation On" input is set, the control loop is interrupted. The MAN manual value is set as the value of the manipulated variable.

Internal Parameters

Addr	Parameter	Decl.	Data Type	Range of Values	Initial Value	Description
44.0	DEADB_W	INPUT	REAL	Dependent on the sensors used	0.0	DEAD BAND WIDTH The error passes through a dead band. The "dead band width" input decides the size of the dead band.
48.0	I_ITLVAL	INPUT	REAL	0 to 100%	0.0	INITIALIZATION VALUE OF THE INTEGRAL ACTION The output of the integral action can be set at the I_ITL_ON input. The initialization value is applied to the "Initialization value of the integral action" input. During a restart COM_RST = TRUE, the I action is set to the initialization value.
52.0	LMN_HLM	INPUT	REAL	> LMN_LLM	100.0	MANIPULATED VARIABLE HIGH LIMIT The value of the manipulated variable is always limited to an upper and lower limit. The "manipulated variable high limit" input specifies the upper limit.
56.0	LMN_LLM	INPUT	REAL	< LMN_HLM	0.0	MANIPULATED VARIABLE LOW LIMIT The value of the manipulated variable is always limited to an upper and lower limit. The "manipulated variable low limit" input specifies the lower limit.
60.0	PV_FAC	INPUT	REAL		1.0	PROCESS VARIABLE FACTOR The "process variable factor" input is multiplied by the "process value periphery". The input is used to adapt the process variable range.
64.0	PV_OFFS	INPUT	REAL		0.0	PROCESS VARIABLE OFFSET The "process variable offset" input is added to the "process variable periphery". The input is used to adapt the process variable range.
68.0	LMN_FAC	INPUT	REAL		1.0	MANIPULATED VARIABLE FACTOR The "manipulated variable factor" input is multiplied by the manipulated variable. The input is used to adapt the manipulated variable range.

Addr	Parameter	Decl.	Data Type	Range of Values	Initial Value	Description
72.0	LMN_OFFS	INPUT	REAL		0.0	MANIPULATED VARIABLE OFFSET The "manipulated variable offset" input is added to the value of the manipulated variable. The input is used to adapt the manipulated variable range.
76.0	PER_TM	INPUT	REAL	≥ CYCLE	1.0 s	PERIOD TIME [s] The pulse repetition period of the pulse duration modulation is entered at the PER_TM parameter. The relationship of the pulse repetition period to the sampling time of the pulse generator decides the accuracy of the pulse duration modulation.
80.0	P_B_TM	INPUT	REAL	≥ 0.0 s	0.02 s	MINIMUM PULSE/BREAK TIME [s] A minimum pulse or minimum break time can be set at the "minimum pulse/break time" parameter. P_B_TM is limited internally to > CYCLE_P.
84.0	TUN_DLMN	INPUT	REAL	-100.0 to 100.0 %	20.0	DELTA MANIPULATED VARIABLE FOR PROCESS EXCITATION Process excitation for controller tuning results from a setpoint step change at TUN_DLMN.
88.0	PER_MODE	INPUT	INT	0, 1, 2	0	PERIPHERY MODE You can enter the type of the I/O module at this switch. The process variable at input PV_PER is then normalized to °C at the PV output. <ul style="list-style-type: none"> • PER_MODE =0: standard • PER_MODE =1: climate • PER_MODE =2: current/voltage
90.0	PVPER_ON	INPUT	BOOL		FALSE	PROCESS VARIABLE PERIPHERY ON If you want the process variable to be read in from the I/O, the PV_PER input must be connected to the I/O and the "process variable periphery" input must be set.

Addr	Parameter	Decl.	Data Type	Range of Values	Initial Value	Description
90.1	I_ITL_ON	INPUT	BOOL		FALSE	INITIALIZATION OF THE INTEGRAL ACTION ON
90.2	PULSE_ON	INPUT	BOOL		FALSE	PULSE GENERATOR ON If PULSE_ON = TRUE is set, the pulse generator is activated
90.3	TUN_KEEP	INPUT	BOOL		FALSE	KEEP TUNING ON The mode changes to automatic only when TUN_KEEP changes to FALSE.
92.0	ER	OUTPUT	REAL	Dependent on the sensors used	0.0	ERROR SIGNAL The effective error is output at the "error signal" output.
96.0	LMN_P	OUTPUT	REAL		0.0	PROPORTIONALITY COMPONENT The "proportionality component" contains the proportional action of the manipulated variable.
100.0	LMN_I	OUTPUT	REAL		0.0	INTEGRAL COMPONENT The "integral component" contains the integral action of the manipulated variable.
104.0	LMN_D	OUTPUT	REAL		0.0	DERIVATIVE COMPONENT The "derivative component" contains the derivative action of the manipulated variable.
108.0	PHASE	OUTPUT	INT	0, 1, 2, 3, 4, 5, 7	0	PHASE OF SELF TUNING The current phase of the controller tuning is indicated at the PHASE output (0..7).
110.0	STATUS_H	OUTPUT	INT		0	STATUS HEATING OF SELF TUNING STATUS_H indicates the diagnostic value of the search for the point of inflection when heating.
112.0	STATUS_D	OUTPUT	INT		0	STATUS CONTROLLER DESIGN OF SELF TUNING STATUS_D indicated the diagnostic value of the controller design when heating.
114.0	QTUN_RUN	OUTPUT	BOOL		0	TUNING IS ACTIVE (PHASE 2) The tuning manipulated variable has been applied, tuning has started and is still in phase 2 (locating the point of inflection).

Addr	Parameter	Decl.	Data Type	Range of Values	Initial Value	Description
116.0	PI_CON	OUTPUT	STRUCT			PI CONTROLLER PARAMETERS
+0.0	GAIN	OUTPUT	REAL	%/phys.unit	0.0	PI PROPORTIONAL GAIN
+4.0	TI	OUTPUT	REAL	≥ 0.0 s	0.0 s	PI RESET TIME [s]
124.0	PID_CON	OUTPUT	STRUCT			PID CONTROLLER PARAMETERS
+0.0	GAIN	OUTPUT	REAL		0.0	PID PROPORTIONAL GAIN
+4.0	TI	OUTPUT	REAL	≥ 0.0 s	0.0 s	PID RESET TIME [s]
+8.0	TD	OUTPUT	REAL	≥ 0.0 s	0.0 s	PID DERIVATIVE TIME [s]
136.0	PAR_SAVE	OUTPUT	STRUCT			SAVED CONTROLLER PARAMETERS The PID parameters are saved in this structure.
+0.0	PFAC_SP	INPUT/ OUTPUT	REAL	0.0 to 1.0	1.0	PROPORTIONAL FACTOR FOR SETPOINT CHANGES
+4.0	GAIN	OUTPUT	REAL	%/phys.unit	0.0	PROPORTIONAL GAIN
+8.0	TI	INPUT/ OUTPUT	REAL	≥ 0.0 s	40.0 s	RESET TIME [s]
+12.0	TD	INPUT/ OUTPUT	REAL	≥ 0.0 s	10.0 s	DERIVATIVE TIME [s]
+16.0	D_F	OUTPUT	REAL	5.0 to 10.0	5.0	DERIVATIVE FACTOR
+20.0	CON_ZONE	OUTPUT	REAL	≥ 0.0	100.0	CONTROL ZONE ON
+24.0	CONZ_ON	OUTPUT	BOOL		FALSE	CONTROL ZONE
162.0	PFAC_SP	INPUT/ OUTPUT	REAL	0.0 to 1.0	1.0	PROPORTIONAL FACTOR FOR SETPOINT CHANGES 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"> • 1: P action has full effect if the setpoint changes. • 0: P action has no effect if the setpoint changes.
166.0	GAIN	INPUT/ OUTPUT	REAL	%/phys. unit	2.0	PROPORTIONAL GAIN The "proportional gain" input specifies the controller gain. The direction of control can be reversed by giving GAIN a negative sign.

Addr	Parameter	Decl.	Data Type	Range of Values	Initial Value	Description
170.0	TI	INPUT/ OUTPUT	REAL	≥ 0.0 s	40.0 s	RESET TIME [s]
174.0	TD	INPUT/ OUTPUT	REAL	≥ 0.0 s	10.0 s	DERIVATIVE TIME [s] The "derivative time" input decides the derivative action response.
178.0	D_F	INPUT/ OUTPUT	REAL	5.0 to 10.0	5.0	DERIVATIVE FACTOR The derivative factor decides the lag of the D action. <ul style="list-style-type: none"> D_F = derivative time/ "lag of the D action"
182.0	CON_ZONE	INPUT/ OUTPUT	REAL	Dependent on the sensors used	100.0	CONTROL ZONE If the error is greater than the control zone width, the upper manipulated variable limit is output as the manipulated variable. If the error is less than the negative control zone width, the lower manipulated variable limit is output as the manipulated variable.
186.0	CONZ_ON	INPUT/ OUTPUT	BOOL		FALSE	CONTROL ZONE ON CONZ_ON =TRUE activates the control zone.
186.1	TUN_ON	INPUT/ OUTPUT	BOOL		FALSE	SELF TUNING ON If TUN_ON=TRUE is set, the manipulated value is averaged until the manipulated variable excitation TUN_DLMN is activated either by a setpoint step change or by TUN_ST=TRUE.
186.2	TUN_ST	INPUT/ OUTPUT	BOOL		FALSE	START SELF TUNING If the setpoint is to remain constant during controller tuning at the operating point, a manipulated variable step change by the amount of TUN_DLMN is activated by TUN_ST=1.

Addr	Parameter	Decl.	Data Type	Range of Values	Initial Value	Description
186.3	UNDO_PAR	INPUT/ OUTPUT	BOOL		FALSE	UNDO CHANGE OF CONTROLLER PARAMETERS
186.4	SAVE_PAR	INPUT/ OUTPUT	BOOL		FALSE	SAVE CURRENT CONTROLLER PARAMETERS Saves the controller parameters PFAC_SP, GAIN, TI, TD, D_F CONZ_ON and CON_ZONE in the data structure PAR_SAVE.
186.5	LOAD_PID	INPUT/ OUTPUT	BOOL		FALSE	LOAD OPTIMIZED PI/PID PARAMETERS Loads the controller parameters GAIN, TI, TD depending on PID_ON from the data structure PI_CON or PID_CON (only in manual mode)
186.6	PID_ON	INPUT/ OUTPUT	BOOL		TRUE	PID MODE ON 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"> • PID controller: PID_ON = TRUE • PI controller: PID_ON = FALSE It is nevertheless possible that with certain process types, only a PI controller will be designed despite PID_ON = TRUE.
188.0	GAIN_P	OUTPUT	REAL		0.0	PROCESS PROPORTIONAL GAIN Identified process gain. For the process type I, GAIN_P tends to be estimated too low.
192.0	TU	OUTPUT	REAL	$\geq 3 \cdot \text{CYCLE}$	0.0	DELAY TIME [s] Identified delay of the process.
196.0	TA	OUTPUT	REAL		0.0	RECOVERY TIME [s] Identified system time constant of the process. For the process type I, TA tends to be estimated too low.
200.0	KIG	OUTPUT	REAL		0.0	MAXIMAL ASCENT RATIO OF PV WITH 100 % LMN CHANGE $\text{GAIN_P} = 0.01 \cdot \text{KIG} \cdot \text{TA}$
204.0	N_PTN	OUTPUT	REAL	1.01 to 10.0	0.0	PROCESS ORDER The parameter specifies the order of the process. "Non-integer values" are also possible.

Addr	Parameter	Decl.	Data Type	Range of Values	Initial Value	Description
208.0	TM_LAG_P	OUTPUT	REAL		0.0	TIME LAG OF PTN MODEL [s] Time lag of a PTN model (practical values only for N_PTN >= 2).
212.0	T_P_INF	OUTPUT	REAL		0.0	TIME TO POINT OF INFLECTION [s] Time from process excitation until the point of inflection.
216.0	P_INF	OUTPUT	REAL	Dependent on the sensors used	0.0	PV AT POINT OF INFLECTION - PV0 Process variable change from process excitation until the point of inflection.
220.0	LMN0	OUTPUT	REAL	0 to 100%	0.0	MANIPULATED VAR. AT BEGIN OF TUNING Detected in phase 1 (mean value).
224.0	PV0	OUTPUT	REAL	Dependent on the sensors used	0.0	PROCESS VALUE AT BEGIN OF TUNING
228.0	PVDT0	OUTPUT	REAL		0.0	RATE OF CHANGE OF PV AT BEGIN OF TUNING [1/s] Sign adapted.
232.0	PVDT	OUTPUT	REAL		0.0	CURRENT RATE OF CHANGE OF PV [1/s] Sign adapted.
236.0	PVDT_MAX	OUTPUT	REAL		0.0	MAX. RATE OF CHANGE OF PV PER SECOND [1/s] Maximum rate of change of the process variable at the point of inflection at the (sign adapted, always > 0), used to calculate TU and KIG.
240.0	NOI_PVDT	OUTPUT	REAL		0.0	RATIO OF NOISE IN PVDT_MAX IN % The higher the proportion of noise, less accurate (less aggressive) the control parameters.
244.0	NOISE_PV	OUTPUT	REAL		0.0	ABSOLUTE NOISE IN PV Difference between maximum and minimum process variable in phase 1.
248.0	FIL_CYC	OUTPUT	INT	1 ... 1024	1	NO OF CYCLES FOR MEAN-VALUE FILTER The process variable is averaged over FIL_CYC cycles. When necessary, FIL_CYC is increased automatically from 1 to a maximum of 1024.

Addr	Parameter	Decl.	Data Type	Range of Values	Initial Value	Description
250.0	POI_CMAX	OUTPUT	INT		2	MAX NO. OF CYCLES AFTER POINT OF INFLECTION This time is used to find a further (in other words better) point of inflection when measurement noise is present. The tuning is completed only after this time.
252.0	POI_CYCL	OUTPUT	INT		0	NUMBER OF CYCLES AFTER POINT OF INFLECTION

A.3.2 Instance DB for FB 59 "TCONT_S"

Parameter:

Addr	Parameter	Decl.	Data Type	Range of Values	Initial Value	Description
0.0	CYCLE	INPUT	REAL	≥ 0.001	0.1 s	SAMPLE TIME OF STEP CONTROLLER [s] At this input, you enter the sampling time for the controller.
4.0	SP_INT	INPUT	REAL	Dependent on the sensors used	0.0	INTERNAL SETPOINT The "Internal Setpoint" input is used to specify a setpoint.
8.0	PV_IN	INPUT	REAL	Dependent on the sensors used	0.0	PROCESS VARIABLE IN An initialization value can be set at the "Process Variable In" input or an external process variable in floating-point format can be connected.
12.0	PV_PER	INPUT	INT		0	PROCESS VARIABLE PERIPHERY The process variable in the peripheral I/O format is connected to the controller at the "Process Variable Peripheral" input.
14.0	DISV	INPUT	REAL		0.0	DISTURBANCE VARIABLE For feedforward control, the disturbance variable is connected to the "Disturbance Variable" input.
18.0	LMNR_HS	INPUT	BOOL		FALSE	HIGH LIMIT SIGNAL OF REPEATED MANIPULATED VALUE The signal "valve at upper limit stop" is connected to the "Upper limit stop signal of the position feedback signal". <ul style="list-style-type: none"> LMNR_HS=TRUE: The valve is at the upper limit stop.

Addr	Parameter	Decl.	Data Type	Range of Values	Initial Value	Description
18.1	LMNR_LS	INPUT	BOOL		FALSE	LOW LIMIT SIGNAL OF REPEATED MANIPULATED VALUE
18.2	LMNS_ON	INPUT	BOOL		TRUE	MANIPULATED SIGNALS ON The processing of the controller output signal is set to manual at the "manipulated signals on" input.
18.3	LMNUP	INPUT	BOOL		FALSE	MANIPULATED SIGNALS UP With the controller output signals set to manual, the QLMNUP output signal is applied to the "manipulated signals up" input.
18.4	LMNDN	INPUT	BOOL		FALSE	MANIPULATED SIGNALS DOWN With the controller output signals set to manual, the QLMNDN output signal is applied to the "manipulated signals down" input.
20.0	QLMNUP	OUTPUT	BOOL		FALSE	MANIPULATED SIGNAL UP If the "manipulated signal up" output is set, the valve will be opened.
20.1	QLMNDN	OUTPUT	BOOL		FALSE	MANIPULATED SIGNAL DOWN If the "manipulated signal down" output is set, the valve will be closed.
22.0	PV	OUTPUT	REAL		0.0	PROCESS VARIABLE The effective process variable is output at the "Process Variable" output.
26.0	ER	OUTPUT	REAL		0.0	ERROR SIGNAL The effective error is output at the "error signal" output.
30.0	COM_RST	INPUT/ OUTPUT	BOOL		FALSE	COMPLETE RESTART The block has an initialization routine that is processed when the COM_RST input is set.

Internal Parameters

Addr	Parameter	Decl	Data Type	Range of values	Initial Value	Description
32.0	PV_FAC	INPUT	REAL		1.0	PROCESS VARIABLE FACTOR The "process variable factor" input is multiplied by the "process value". The input is used to adapt the process variable range.
36.0	PV_OFFS	INPUT	REAL	Dependent on the sensors used	0.0	PROCESS VARIABLE OFFSET The "process variable offset" input is added to the process variable. The input is used to adapt the process variable range.
40.0	DEADB_W	INPUT	REAL	≥ 0.0	0.0	DEAD BAND WIDTH The error passes through a dead band. The "dead band width" input decides the size of the dead band.
44.4	PFAC_SP	INPUT	REAL	0.0 to 1.0	1.0	PROPORTIONAL FACTOR FOR SETPOINT CHANGES [0..1] 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"> • 1: P action has full effect if the setpoint changes. • 0: P action has no effect if the setpoint changes.
48.0	GAIN	INPUT	REAL	%/phys. unit	2.0	PROPORTIONAL GAIN The "proportional gain" input specifies the controller gain. The direction of control can be reversed by giving GAIN a negative sign.
52.0	TI	INPUT	REAL	≥ 0.0 s	40.0 s	RESET TIME [s] The "reset time" input (integral time) decides the integral action response.
56.0	MTR_TM	INPUT	REAL	\geq CYCLE	30 s	MOTOR ACTUATING TIME The run time of the valve from limit stop to limit stop is entered in the "motor actuating time" parameter.
60.0	PULSE_TM	INPUT	REAL	≥ 0.0 s	0.0 s	MINIMUM PULSE TIME [s] A minimum pulse time can be set with the "minimum pulse time" parameter.
64.0	BREAK_TM	INPUT	REAL	≥ 0.0 s	0.0 s	MINIMUM BREAK TIME [s] A minimum break time can be set with the "minimum break time" parameter.

Addr	Parameter	Decl	Data Type	Range of values	Initial Value	Description
68.0	PER_MODE	INPUT	INT	0, 1, 2	0	<p>PERIPHERY MODE</p> <p>You can enter the type of the I/O module at this switch. The process variable at input PV_PER is then normalized to °C at the PV output.</p> <ul style="list-style-type: none"> • PER_MODE =0: standard • PER_MODE =1: climate • PER_MODE =2: current/voltage
70.0	PVPER_ON	INPUT	BOOL		FALSE	<p>PROCESS VARIABLE PERIPHERY ON</p> <p>If you want the process variable to be read in from the I/O, the PV_PER input must be connected to the I/O and the "process variable periphery" input must be set.</p>

A.4 List of Possible Messages during Tuning

STATUS_H	Description	Remedy
0	Default or no new controller parameters (yet).	
10000	Tuning completed and suitable controller parameters found	
2xxxx	Tuning completed and controller parameters uncertain	
2xx2x	Point of inflection not reached (only with excitation by setpoint step change)	If the controller is oscillating, weaken the controller parameters and repeat the attempt with a lower manipulated variable difference TUN_DLMN.
2x1xx	Estimation error ($TU < 3 \cdot CYCLE$)	Reduce CYCLE and repeat the attempt Special case, purely PT1 process: Do not repeat, possibly weaken controller parameters.
2x3xx	TU estimated too high	Repeat the attempt under better conditions
21xxx	Estimation error $N_PTN < 1$	Repeat the attempt under better conditions
22xxx	Estimation error $N_PTN > 10$	Repeat the attempt under better conditions
3xxxx	Tuning terminated in Phase 1 due to bad parameter settings:	
30002	Effective manip. var. difference $< 5\%$	Correct manipulated variable difference 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 level and note call distributors which may exist. Check the CPU utilization. If your CPU is highly utilized, you will have prolonged sampling times which may not match CYCLE or CYCLE_P.

Note

If you stop the tuning in phase 1 or 2, STATUS_H is set to 0. STATUS_D, however, continues to indicate the status of the last controller calculation.

The higher the value of STATUS_D, the higher the order of the controlled process, the higher the ratio TU/TA and the softer the controller parameters.

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 (intermediate 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 were corrected during Phase 7.

B Abbreviations and Acronyms

Abbr./Acronym	Explanation
BREAK_TM	Minimum break time [s]
COM_RST	Restart
CON_ZONE	Control zone width
CONZ_ON	Activate control zone
CYCLE	Sampling time [s]
CYCLE_P	Sampling time of the pulse generator [s]
D_F	Derivative factor
DEADB_W	Deadband width
DISV	Disturbance variable
ER	Error
FIL_CYC	Number of cycles of the mean value filter
GAIN	Controller gain
GAIN_P	Process gain
I_ITL_ON	Set I-action
I_ITLVAL	Initialization value for I-action
INT_HNEG	Block I-action in negative direction
INT_HPOS	Block I-action in positive direction
KIG	Maximum process value rate of rise following a manipulated variable change from 0 to 100 % [1/s]
LMN	Value of the manipulated variable
LMN_D	D-action
LMN_FAC	Manipulated variable factor
LMN_HLM	Manipulated variable high limit
LMN_I	I-action
LMN_LLM	Manipulated variable low limit
LMN_OFFS	Manipulated variable offset
LMN_P	P-action
LMN_PER	Manipulated variable periphery
LMN0	Manipulated variable at start of tuning
LMNDN	Actuating signal down
LMNR_HS	Upper limit stop signal of the position feedback signal

Abbr./Acronym	Explanation
LMNR_LS	Lower limit stop signal of the position feedback signal
LMNS_ON	Activate manual mode for actuating signals
LMNUP	Actuating signal up
LOAD_PID	Load tuned PID parameters
MAN	Manual value
MAN_ON	Activate manual mode
MTR_TM	Motor actuating time [s]
N_PTN	Process order
NOI_PVDT	Noise proportion in PVDT_MAX in %
NOISE_PV	Absolute noise in process value
P_B_TM	Minimum pulse/minimum break time [s]
P_INF	Process value at point of inflection – PVO
PAR_SAVE	Saved PID controller parameters
PER_MODE	Periphery mode
PER_TM	Pulse repetition period [s]
PFAC_SP	Proportional factor for setpoint changes
PHASE	Phase indicator for controller tuning
PI_CON	PI controller parameter
PID_CON	PID controller parameter
PID_ON	Activate PID mode
POI_CMAX	Maximum no. of cycles after the point of inflection
POI_CYCL	Number of cycles after the point of inflection
PULSE_ON	Activate pulse generator
PULSE_TM	Minimum pulse time [s]
PV	Actual value
PV_FAC	Process value factor
PV_IN	Process value input
PV_OFFS	Process value offset
PV_PER	Process value periphery
PV0	Process value at start of tuning
PVDT	Current process value rate of rise [1/s]
PVDT_MAX	Max. change of process value per second [1/s]
PVDT0	Process value rate of rise at start of tuning [1/s]
PVPER_ON	Activate process value periphery
QC_ACT	Continuous controller action will be processed at the next call
QLMN_HLM	High limit of the manipulated variable reached
QLMN_LLM	Low limit of the manipulated variable reached

Abbr./Acronym	Explanation
QLMNDN	Actuating signal down
QLMNUP	Actuating signal up
QPULSE	Pulse output
QTUN_RUN	Tuning active (phase 2)
SAVE_PAR	Save current controller parameters
SELECT	Selection of the call for PID and pulse generator
SP_INT	Internal setpoint
STATUS_D	Status controller design of the controller tuning
STATUS_H	Status heating of the controller tuning
T_P_INF	Time to point of inflection [s]
TA	Time response of the process (recovery time) [s]
TD	Derivative time constant [s]
TI	Integral time constant or reset time [s]
TM_LAG_P	Time constant of a PTN model [s]
TU	Time lag of the process (delay time) [s]
TUN_DLMN	Delta manipulated variable for process excitation
TUN_KEEP	Keep tuning mode
TUN_ON	Activate controller tuning
TUN_ST	Start controller tuning
UNDO_PAR	Undo controller parameter change

Index

C	
Control	6-1
Controller sampling time	2-15, 2-16, 4-7
Controller tuning.....	3-1
improving	3-16
messages	A-17
phases	3-4
problems	3-12
result	3-11
starting	3-8
stopping	3-11
strong heat coupling.....	3-19
weak heat coupling	3-19
Cooling.....	1-3
Cooling process	1-3
CYCLE	2-15
CYCLE_P	2-15, 2-17
D	
Disturbances.....	3-3
E	
Example with FB 58 "TCONT_CP"	6-2
F	
FB 58 "TCONT_CP"	
application.....	1-3
block diagram.....	2-13
block diagram error formation	2-1
block diagram PID algorithm.....	2-4, 2-9, 2-12,
.....	3-2, 3-4, 3-5
block diagram saving and reloading controller parameters	2-9
control zone	2-6
controller sampling time	2-15, 2-16
controller tuning	3-1
cooling process.....	1-3
deadband.....	2-3
description	1-4
error formation	2-1, 2-3
example	6-2, 6-6, 6-7
feedforward control	2-6
initialization	2-14, 2-15, 2-18
instance DB.....	A-2
integrator.....	2-5
manipulated variable calculation	2-9, 2-12,
.....	3-2, 3-4, 3-5
manipulated variable limitation.....	2-8
manipulated variable normalization.....	2-8
manual value processing	2-7
PID algorithm.....	2-4
preparations	3-6
process value format conversion.....	2-2
process value normalization.....	2-2
process value normalization, example.....	2-3
process value options.....	2-1
pulse generator	2-11
reloading controller parameters.....	2-9
sampling time of the pulse generator.....	2-15
saving controller parameters	2-9
setpoint branch.....	2-1
structure outline.....	1-3
weakening the P action	2-5
FB 59 "TCONT_S".....	1-4
application	1-4
block diagram	4-5
deadband	4-3
description	1-4
error formation.....	4-1, 4-3
initialization.....	4-7
instance DB.....	A-13
PI step controller algorithm.....	4-4
process value conversion	4-2
process value normalization	4-2
process value options.....	4-1
sampling time	4-7
setpoint branch.....	4-1
Fine tuning in control mode	3-16
G	
Getting Started	5-1
I	
Instance DB for FB 58 "TCONT_CP"	A-2
Instance DB for FB 59 "TCONT_S"	A-13
L	
Linearity.....	3-3
M	
Messages during tuning	A-17
O	
Operating range	3-3

P	
PI step controller algorithm	4-4
feedforward control	4-4
Process type	
checking.....	3-10
Process types	3-2
Product structure.....	1-1
Pulse duration modulation	2-11
Pulse generator	2-11
PULSEGEN	2-11
R	
Readme file.....	1-2
S	
Sample for FB 58 "TCONT_CP"	6-6, 6-7
Sample for FB 59 "TCONT_S"	6-11
Sampling time of the pulse generator	2-15
Software	
installing	1-1
STATUS_H.....	A-17
T	
Transient response.....	3-3