SIEMENS

Ingenuity for life

Multi-Zone Control with "PID_Temp”

SIMATIC S7-1200/S7-1500 and STEP 7 (TIA Portal)

https://support.industry.siemens.com/cs/ww/en/view/109740463

Siemens Industry Online Support

Legal information

Use of application examples

Application examples illustrate the solution of automation tasks through an interaction of several components in the form of text, graphics and/or software modules. The application examples are a free service by Siemens AG and/or a subsidiary of Siemens AG ("Siemens"). They are non-binding and make no claim to completeness or functionality regarding configuration and equipment. The application examples merely offer help with typical tasks; they do not constitute customer-specific solutions. You yourself are responsible for the proper and safe operation of the products in accordance with applicable regulations and must also check the function of the respective application example and customize it for your system.

Siemens grants you the non-exclusive, non-sublicensable and non-transferable right to have the application examples used by technically trained personnel. Any change to the application examples is your responsibility. Sharing the application examples with third parties or copying the application examples or excerpts thereof is permitted only in combination with your own products. The application examples are not required to undergo the customary tests and quality inspections of a chargeable product; they may have functional and performance defects as well as errors. It is your responsibility to use them in such a manner that any malfunctions that may occur do not result in property damage or injury to persons.

Disclaimer of liability

Siemens shall not assume any liability, for any legal reason whatsoever, including, without limitation, liability for the usability, availability, completeness and freedom from defects of the application examples as well as for related information, configuration and performance data and any damage caused thereby. This shall not apply in cases of mandatory liability, for example under the German Product Liability Act, or in cases of intent, gross negligence, or culpable loss of life, bodily injury or damage to health, non-compliance with a guarantee, fraudulent non-disclosure of a defect, or culpable breach of material contractual obligations. Claims for damages arising from a breach of material contractual obligations shall however be limited to the foreseeable damage typical of the type of agreement, unless liability arises from intent or gross negligence or is based on loss of life, bodily injury or damage to health. The foregoing provisions do not imply any change in the burden of proof to your detriment. You shall indemnify Siemens against existing or future claims of third parties in this connection except where Siemens is mandatorily liable.

By using the application examples you acknowledge that Siemens cannot be held liable for any damage beyond the liability provisions described.

Other information

Siemens reserves the right to make changes to the application examples at any time without notice. In case of discrepancies between the suggestions in the application examples and other Siemens publications such as catalogs, the content of the other documentation shall have precedence.

The Siemens terms of use (https://support.industry.siemens.com) shall also apply.

Security information

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

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

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

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

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

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

Table of contents

1 Task

1.1 Overview

Introduction

For a multi-zone control several subsections, so called zones, of a plant are simultaneously controlled at different temperatures. The mutual impact on the temperature zones by thermal coupling is characteristic for multi-zone control. This means that the actual value of a zone can influence the actual value in another zone by heat coupling. It depends on the structure of the plant and the selected operating points of the zones how heavily the zones influence each other.

An example of a multi-zone control is an extrusion plant, as it used in plastics processing, among others.

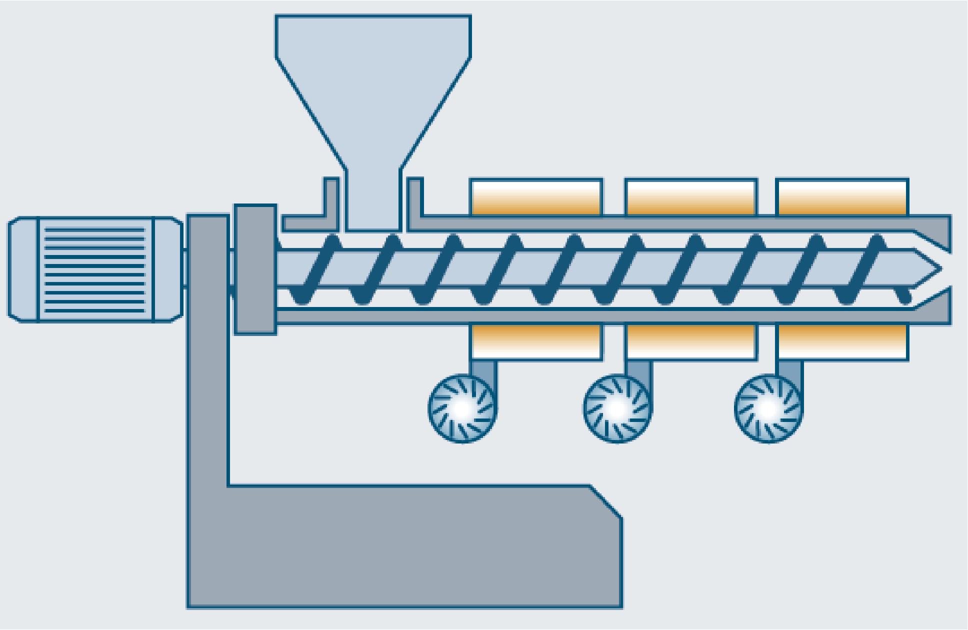
Mode of Operation

The material is filled in with the help of a funnel (for example, plastic granulate). The screw conveyor transports the material through the heated and/or cooled screw cylinder. The material is melted via different temperature zones (by heating, friction and shearing). In parts, this creates so much heat that the material has to be cooled. A forming aperture at the end of the extruder (outlet nozzle) presses the melt into shape. Before shaping, the material has to be cooled.

Overview of the application example

The following figure gives you an overview of the automation task.

Figure 1-1



The control parameters of the individual zone controllers in the application example are determined simultaneously, despite the temperature influence on the zones.

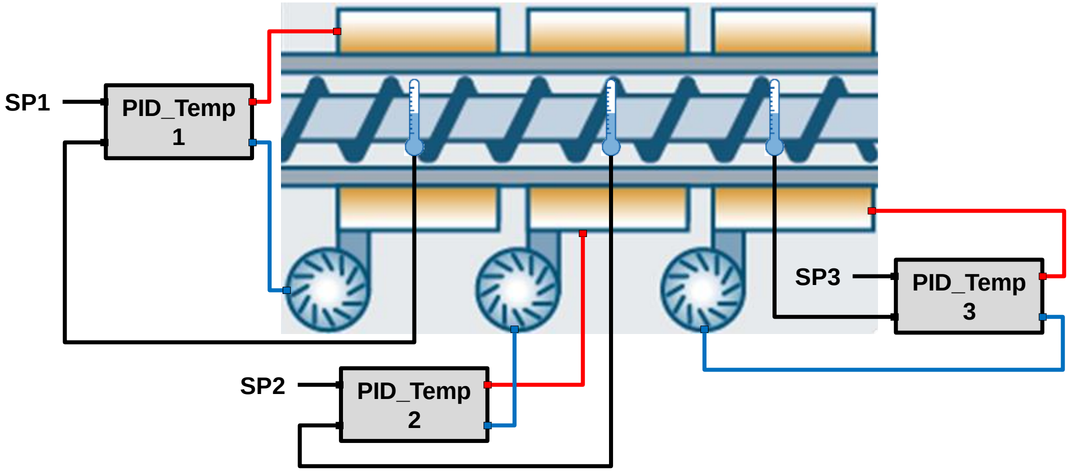
2 Solution

2.1 Overview

Schematic layout

The figure below shows you a schematic illustration of the main components of this solution.

Figure 2-1



Configuration

In order to control the individual zones, the technology object "PID_Temp” of the SIMATIC S7-1200 controller family or SIMATIC S7-1500 is used.

This technology object presents a continuous PID controller with integrated tuning and is designed especially for temperature control. "PID_Temp” is suitable for heating or heating/cooling applications. Two outputs are available for this purpose, one for heating and one for cooling.

When using the technology object "PID_Temp” in multi-zone controllers, each temperature zone is controlled by its own "PID_Temp” instance.

To keep the influence of neighboring zones as low as possible, you can synchronize the individual "PID_Temp” controller instances in the two tuning types "pretuning” and "fine tuning”.

Note

Pure cooling control is realized with the technology object "PID_Compact” and the "inverting the control direction” option.

More information on the control types can be found in the function manual "SIMATIC S7-1200, S7-1500 PID control” \3\.

https://support.industry.siemens.com/cs/ww/en/view/108210036

Advantages

The solution presented here offers the following advantages:

- Overview of factors you have to observe for a multi-zone control.
- Overview of options to use the technology object "PID_Temp" and to determine the control parameters.
- Time and cost savings through synchronized tuning.
- Expandability: The solution is introduced on the example of a multi-zone controller with three zones. However, you can expand the application as desired.

Topics not covered by this application

This application example relates particularly to the multi-zone control with the technology object "PID_Temp". This technology object can only be used in SIMATIC S7-1200 or S7-1500.

This application example introduces mechanisms that optimize the control parameters of a multi-zone controller with three zones.

The application example provides you with the following blocks:

- Simulation of the coupled controlled system
- Synchronized pretuning for heating and cooling
- Synchronized fine tuning with selectable energy type (heating or cooling)

These blocks are designed for three zones. Chapter 5.2 and the comments in the program code show you how to expand zones.

Zone 1 in the present example is designed as pure heating controllers.
With the configuration, you can select whether the technology object "PID_Temp" is used as heating or as heating/cooling controller.
The program code included can be used for both controller designs.

For the real operation, you have to adjust the application example to your actuators used and the actual value sensors. Basic knowledge is assumed.

Note

More information on the technology object "PID_Temp" can be found in Chapter 6 of the function manual "SIMATIC S7-1200, S7-1500 PID control" \3\.

https://support.industry.siemens.com/cs/ww/en/view/108210036

Assumed knowledge

The following basic knowledge is required and is not explicitly discussed in this description:

- Control engineering
- STEP 7 (TIA Portal)
- The SCL/LAD/FBD programming languages.

2.2 Hardware and software components

2.2.1 Validity

This application example is valid for the following components:

- STEP 7 V15.1 or higher
- S7-1200 CPU firmware as of V4.2 or S7-1500 CPU firmware as of V2.0
- Technology object "PID_Temp” V1.1

2.2.2 Components used

The application example has been created with the following components.

Hardware components

Table 2-1

Component	Qty.	Article number	Note
SIMATIC S7-1200 POWER MODULE PM1207 CONTROLLED POWER SUPPLY INPUT: 120/230 V AC OUTPUT: 24V/2.5 A DC	1	6EP1332-1SH71	Any other SIMATIC power supply with 24V DC can be used.
SIMATIC S7-1200, CPU 1215C, compact CPU, DC/DC/DC, 2 PROFINET Port, onboard I/O: 14 DI 24VDC; 10 DO 24VDC; 0,5A; 2 AI 0-10V DC, 2 AO 0-20mA DC, power supply: 20.4-28.8V DC, 125 kB program/data storage	1	6ES7215-1AG40-0XB0	As of firmware V4.2. Alternatively, any other S7-1200 CPU as of firmware V4.2 can also be used.
SIMATIC S7-1500 compact CPU 1511C-1 PN, central module with 175 kB main memory for program and 1 MByte for data, 16 digital inputs, 16 digital outputs, 5 analog inputs, 2 analog outputs, 6 fast counter, 1 interface: PROFINET IRT with 2 port switch, 60 NS bit performance, incl. push-in front plug, SIMATIC memory card required	1	6ES7511-1CK00-0AB0	As of firmware V2.0. Alternatively, any other S7-1500 CPU as of firmware V2.0 can also be used.
SIMATIC S7, memory card for S7-1x 00 CPU/SINAMICS, 3, 3 V flash, 24 MByte	1	6ES7954-8LF02-0AA0	Required when using a S7-1500 CPU.

Note Other hardware components for controlling real actuators and temperature acquisition can be found in:

- System Manual "SIMATIC S7 S7-1200 Programmable controller" in Chapter A "Technical data" (\4\)
  https://support.industry.siemens.com/cs/ww/en/view/109764129
- Manual "SIMATIC S7-1500/ET 200MP Automation system In a nutshell" (\10\)
  https://support.industry.siemens.com/cs/ww/en/view/109481357
- TIA Selection Tool (\5\)
  http://w3.siemens.com/mcms/topics/en/simatic/tia-selection-tool/Pages/tab.aspx
- Hardware catalog in the TIA Portal

Software components

Table 2-2

Component	Qty.	Article number	Note
STEP 7 Basic V15.1	1	6ES7822-0A.05-..	Minimal license for configuring the S7-1200
STEP 7 Professional V15.1	1	6ES7822-1..05-..	Enables configuration of S7-1200 and S7-1500

Example files and projects

The following table contains all files and projects used in this example.

Table 2-3

Component	Note
109740463_PidTemp_MultiZone_PROJ_v11.zip	This zip file contains the STEP 7 project.
109740463_PidTemp_MultiZone_DOC_v11_en.pdf	This document.

3 Basics

This chapter discusses the tuning types of the technology object "PID_Temp” and how to use them, in order to achieve a stable setting of your multi-zone controller on the basis of the function manual "SIMATIC S7-1200, S7-1500 PID control (\3\), chapter "Multi-zone controlling with PID_Temp”.

3.1 Pretuning

The initial commissioning of a plant usually starts by carrying out a pretuning to perform a first setting of the PID parameters and to control the operating point. The pretuning for multi-zone controllers is often done simultaneously for all zones.

Carry out the pretuning for heating separate from the pretuning for cooling in order to decrease the mutual influence of thermal coupling between the zones during the tuning.

Only start the pretuning for cooling the controllers with active cooling and PID parameter switching when all zones have completed the pretuning for heating and their operating point has been reached.

3.1.1 Adjusting the delay time

With the pretuning for heating, a jump on the output value heating is returned, the PID parameters for heating are calculated and afterwards the setpoint is controlled in automatic mode.

The "AdaptDelayTime” tag determines the adjustment of the delay time for heating on the operating point.

With the "AdaptDelayTime” = 0 default, the delay time is only determined during the step "Determine turning point heating ("PIDSelfTune.SUT.State” = 300).

Only for "AdaptDelayTime” = 1 the delay time is additionally checked by temporarily switching off the heating on the operating point and, if necessary, adjusted ("PIDSelfTune.SUT.State” = 1000).

This is not desired for the multi-zone control because the adjustment of the delay time in this phase may be false due to the thermal coupling of neighboring zones.

This is why you have to make sure that the adjustment of the delay time is disabled for all PID_Temp instances ("PIDSelfTune.SUT.AdaptDelayTime” = 0).

3.1.2 Temporarily switching off cooling

For controllers with active cooling ("Config.ActivateCooling” = TRUE), you can temporarily disable the cooling in automatic mode with "PID_Temp”. To do this, set "DisableCooling” = TRUE.

This is how you can prevent that this controller cools in automatic mode during commissioning, whilst the controllers of the neighboring zones have not yet completed the tuning of the heating.

3.1.3 Procedure

For the synchronized pretuning of multi-zone controllers, proceed as follows:

1. Disable the adjustment of the delay time for all controllers ("PIDSelfTune.SUT.AdaptDelayTime” = 0).
2. Disable the cooling ("DisableCooling” = TRUE) for all controllers with active cooling ("Config.ActivateCooling” = TRUE).

3. Specify the desired setpoint ("Setpoint” parameter) and simultaneously start the pretuning for heating for all controllers ("Heat.EnableTuning” = TRUE, "Cool.EnableTuning” = FALSE, "Mode” = 1, "ModeActivate” = TRUE) from the "Inactive” ("State” = 0) mode.
4. Wait until all controllers have successfully completed the heating pretuning.
5. Enable the cooling ("DisableCooling” = FALSE) for all controllers with active cooling ("Config.ActivateCooling” = TRUE).
6. Wait until all zones have reached the operating point in automatic mode ("State” = 3).

Note The heating or cooling actuator is too weak, if the setpoint cannot be reached permanently for a zone!

7. Start the cooling pretuning ("Config.ActivateCooling” = TRUE) simultaneously for all controllers with active cooling ("Heat.EnableTuning” = FALSE, "Cool.EnableTuning” = TRUE, "Mode” = 1).
8. Wait until all controllers have successfully completed the cooling pretuning.

CAUTION

Exceeding the limit value of the actual value

When you are disabling the cooling in automatic mode ("DisableCooling” = TRUE), the actual value can exceed the setpoint and the actual value limits.

If you are using "DisableCooling”, monitor the actual values and if required take action.

CAUTION

Multi-zone controllers

The thermal coupling between the zones in multi-zone controllers can cause increased overshoots or the temporary exceeding of the limit values and permanent or temporary control deviations during commissioning and during operation.

Monitor the actual values and, if required, take action.
Depending on the plant it may be necessary to deviate from the approach described above.

3.2 Fine tuning

The fine tuning generates a constant, limited oscillation of the actual value. From the amplitude and frequency of this oscillation, the PID parameters for the operating point are optimized. The PID parameters are recalculated from the results. The PID parameters from the fine tuning mostly show a better management and disturbance behavior than the parameters from the pretuning. You get the best PID parameters when you do the pretuning and the fine tuning.

3.2.1 Temporary tuning offset for heating/cooling controller

If "PID_Temp” is used as heating and cooling controller ("Config.ActivateCooling” = TRUE), the PID output value ("PidOutputSum”) must fulfil the following prerequisites so that an actual value oscillation is created and fine tuning can be performed successfully:

- Positive PID output value for heating fine tuning
- Positive PID output value for cooling fine tuning

If this prerequisite is not fulfilled, you can specify a temporary offset for the fine tuning that is returned on the output that has the opposite effect.

- Offset for cooling output ("PIDSelfTune.TIR.OutputOffsetCool”) for fine tuning heating
  Before starting the tuning, specify a negative tuning offset cooling that is smaller than that of the PID output value ("PidOutputSum”) on the setpoint in the stationary state.
- Offset for heating output ("PIDSelfTune.TIR.OutputOffsetHeat”) for fine tuning cooling
  Before starting the tuning, specify a positive tuning offset heating that is larger than that of the PID output value ("PidOutputSum”) in the stationary state.

The specified offset is than balanced out by the PID algorithm so that the actual value on the setpoint remains. For the PID output value to fulfil the above mentioned prerequisites, you can increase the offset. With the level of the offset, the PID output value can therefore be adjusted accordingly so that it fulfils the above mentioned prerequisites.

In order to avoid larger overshoots of the actual value when the offset is specified, it can also be increased in several steps.

If "PID_Temp” leaves the "Fine tuning” mode, the tuning offset is reset.

3.2.2 Synchronizing several fine tunings

If the fine tuning is started in automatic mode with "PIDSelfTune.TIR.RunIn” = FALSE, "PID_Temp” tries to reach the setpoint of the current PID parameters. Only when the setpoint is reached, does the actual tuning start.
The time that is required to reach the setpoint may vary for the individual zones of a multi-zone controller.
If you want to start the fine tuning for several zones at the same time, you can synchronize the fine tuning with "PID_Temp”. To do this, the "PID_Temp” waits until all zones have reached the setpoint before it carries out the next tuning step.

Procedure

This is how you can make sure that all controllers have reached their setpoint before the actual tuning steps start. This prevents mutual influence of thermal coupling between the zones during the tuning.

For controllers, where you want to carry out the fine tuning of their zones simultaneously, proceed as follows:

1. Set "PIDSelfTune.TIR.WaitForControlIn" = TRUE for all controllers. These controllers have to be in automatic mode with "PIDSelfTune.TIR.RunIn" = FALSE.
2. Set the desired setpoints with the "Setpoint" parameter and start the fine tuning for all controllers.
3. Wait until "PIDSelfTune.TIR.ControlInReady" = TRUE is set for all controllers.
4. Set "PIDSelfTune.TIR.FinishControlIn" = TRUE for all controllers.

Thus, all controllers start the actual tuning simultaneously with the calculation of the standard deviation (filtering of noise of actual value signal).

4 Mode of Operation

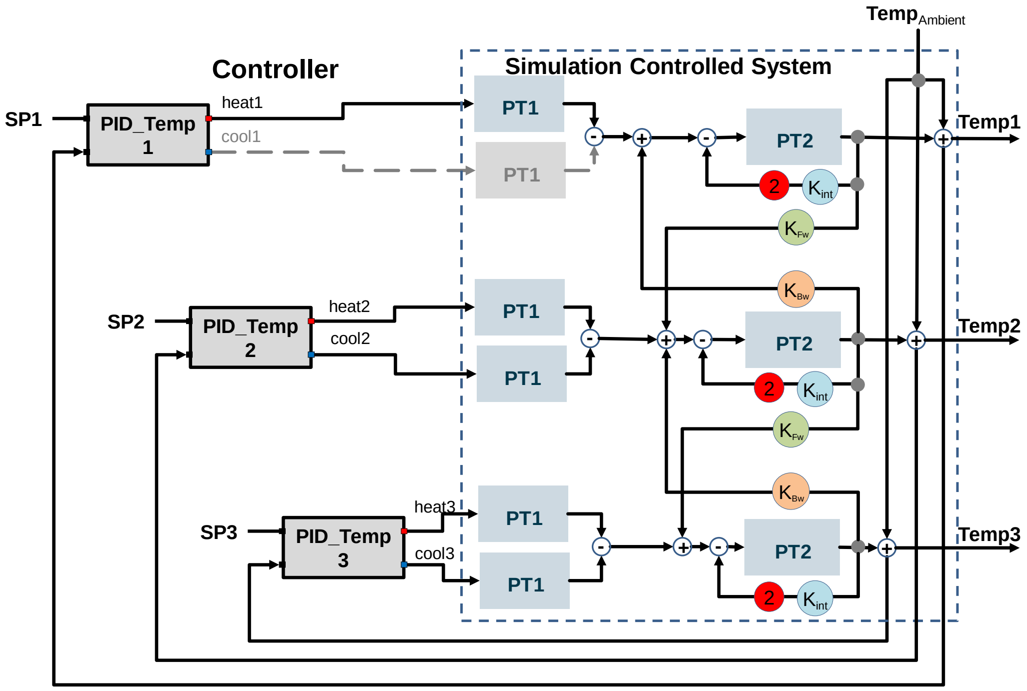
4.1 General overview

Figure 4-1 shows the block diagram of a multi-zone controller with three zones. Each zone is controlled by an individual instance of the technology object "PID_Temp". The individual zone controller "PID_Temp" in the application example is configured as follows:

- Zone 1: Heating controller
- Zone 2: Heating/cooling controller
- Zone 3: Heating/cooling controller

The application example includes the simulation of the controlled system that replicates the thermal coupling of the zones. The controlled system simulation is described in more detail in chapter 4.2.

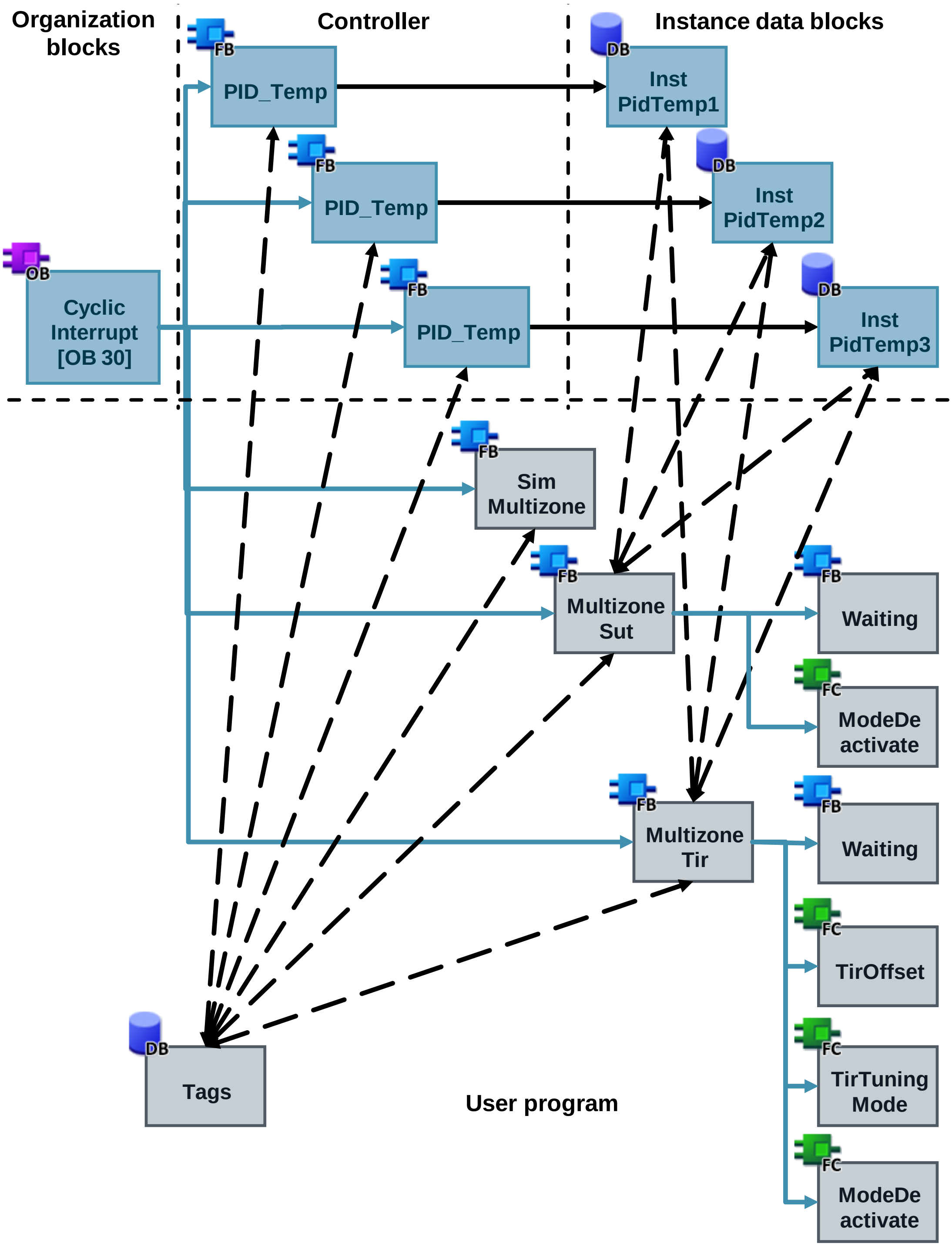
Figure 4-1



Program overview

Figure 4-2 shows the call structure of the blocks for a multi-zone controller with three zones, a controlled system simulation and a synchronized sequence for pretuning and fine tuning.

Figure 4-2



All instructions and functions are called in the interval of the "CyclicInterrupt" interrupt OB. For each controller zone the technology object "PID_Temp" with individual instance DB ("InstPidTempX") is called.

Note The configuration and commissioning interface is only available to you when you call the technology object "PID_Temp" as individual instance.

The FB "SimMultizone" simulates the temperature curves of the coupled zones and is switched according to the specification of Figure 4-1 with the calls of "PID_Temp".

As of STEP 7 the instance DBs of the "PID_Temp" are transferred to the InOut interface of the function blocks as parameter instance.
This feature is used for the synchronized tuning processes:

- FB "MultizoneSut" starts a simultaneous pretuning heating for all zones, followed by a simultaneous pretuning cooling (if configured).
- FB "MultizoneTir" synchronizes the fine tuning of all selected zones (according to energy type specification heating or cooling, depending on zone).

The following subfunctions are used by these optimization blocks:

- FB "Waiting" has the effect that all zones wait until the control deviations are within a specified tolerance range and a waiting period has lapsed.
- FC "TirOffset" calculates heating or cooling according to the specified energy type for the fine tuning depending on zone of the temporary offset on the opposite output.
- FC "TirTuningMode" implements the energy type specification heating or cooling for the fine tuning depending on zone.
- FC "DeactivateMode" resets the input bit "ModeActivate" of all "PID_Temp" calls.

The DB "Tags" includes the parameters for transfer to the block interfaces.

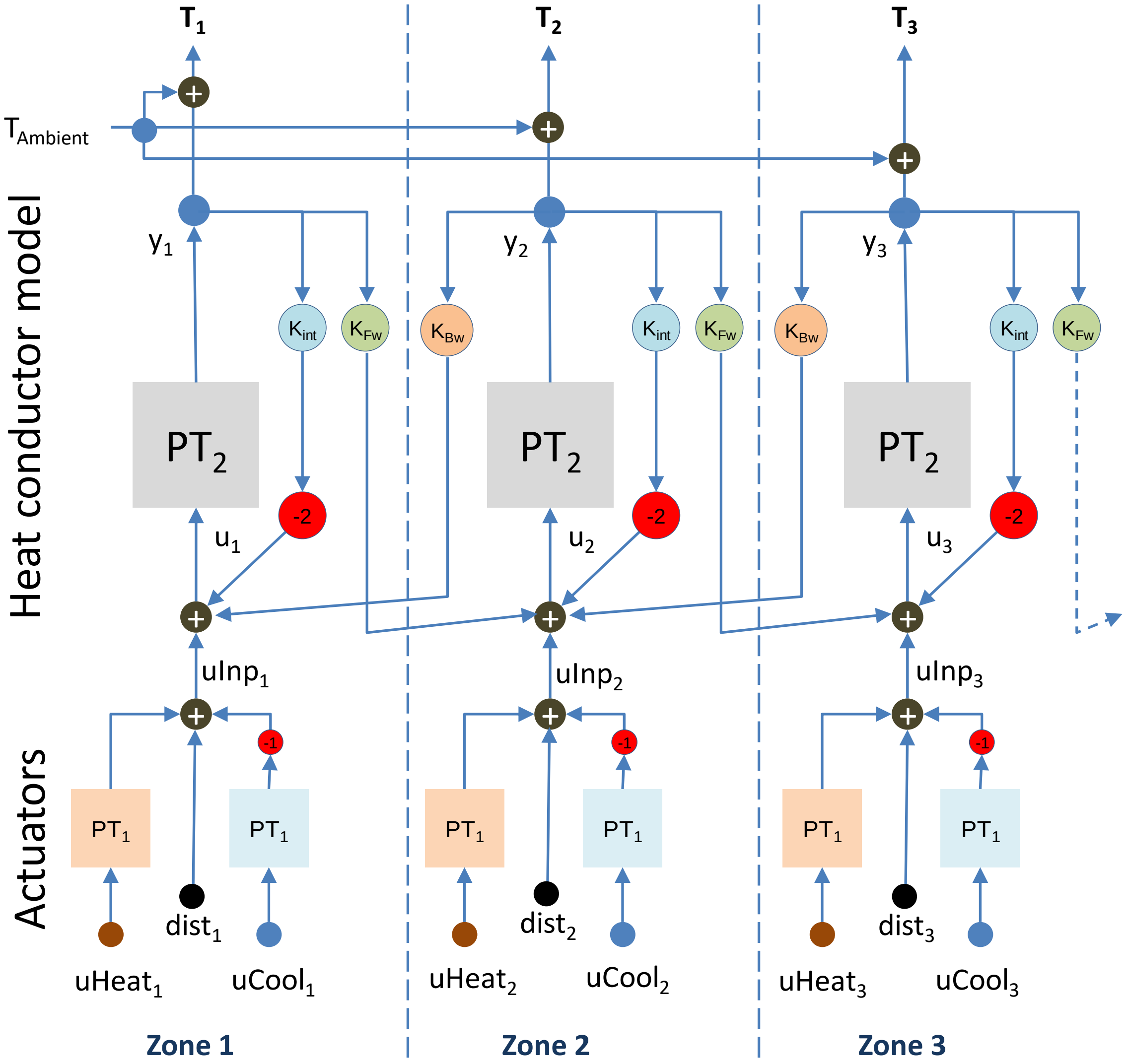
4.2 FB "SimMultizone"

The FB "SimMultizone" simulates the temperature curves of the coupled zones and is switched according to the specification of Figure 4-1 with the calls of "PID_Temp".

4.2.1 Simulation of a coupled thermal stretch

As mathematic model, the discrete solution of a partial differential equation of a simple heat conductor with n grid points is used (\7\).

Figure 4-3: Flow chart of a coupled thermal controlled system



The dynamic of a gird point is exemplary as second-order aperiodic delay element. However, depending on the desired controlled system behavior, you can also use other elements here.

The simulated zone temperature T_z is the result of equation 4.2.1:

Equation 4.2.1: Calculating the zone temperature $\boldsymbol{T_z}$:

$$T_z = y_z + T_{Ambient}$$

$T_{Ambient}$ = ambient temperature

z = Zone

Equation 4.2.2: Transfer function of the PT2 delay element

$$y_z = \frac{gain_{PT2}}{(tmLag1_{PT2} \, * p + 1) * (tmLag2_{PT2} * p + 1)} * u_z$$

Equation 4.2.3: Equation for calculating the PT2 input

$$u_z = kFW_{z-1} \, * y_{z-1} - 2 * kINT_z \, * y_z + kBW_{z+1} \, * y_{z+1} + uInp_z$$

kFW = coupling factor forward

$kINT$ = coupling factor internal

kBW = coupling factor backward

Equation 4.2.4: Summation of the manipulated variable delay by PT1

$$uInp_z = \frac{gain_{Heat}}{tmLag1_{Heat} * p + 1} * uHeat_z - \frac{gain_{Cool}}{tmLag1_{Cool} * p + 1} * uCool_z + dist_z$$

$uHeat$ = manipulated variable heating

$uCool$ = manipulated variable cooling

$dist$ = disturbance variable

The model can be expanded to any number of zones by copying the pattern (Figure 4-3).

In equation 4.2.3 it has to be observed that for the peripheral zones, the previous zone or the following zone is omitted and the terms are therefore to be set to z-1 or z+1 here = 0.

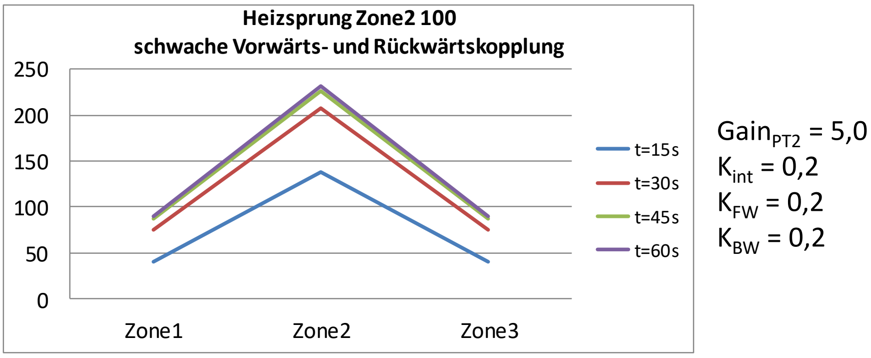
Through the coupling factors K_{FW} and K_{BW} (0 to 1), the thermal influence to neighboring zones can be varied.

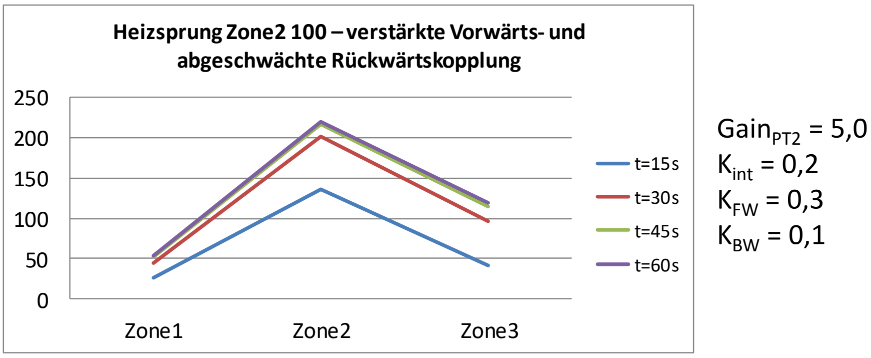
You can vary the static end value of the zone temperatures through the internal coupling factors K_{int} (0 to 1) and the gain factors of the PTx delay elements.

Chronological sequence

In terms of time, a test jump of 100% in zone 2, for example, is divided according to t seconds to the neighboring zones as follows:

Figure 4-4





4.2.2 Configuration explanations

The FB "SimMultizone” is called in the interrupt OB in which the controllers "PID_Temp” are also called.

Table 4-1: Parameters of SimMultizone

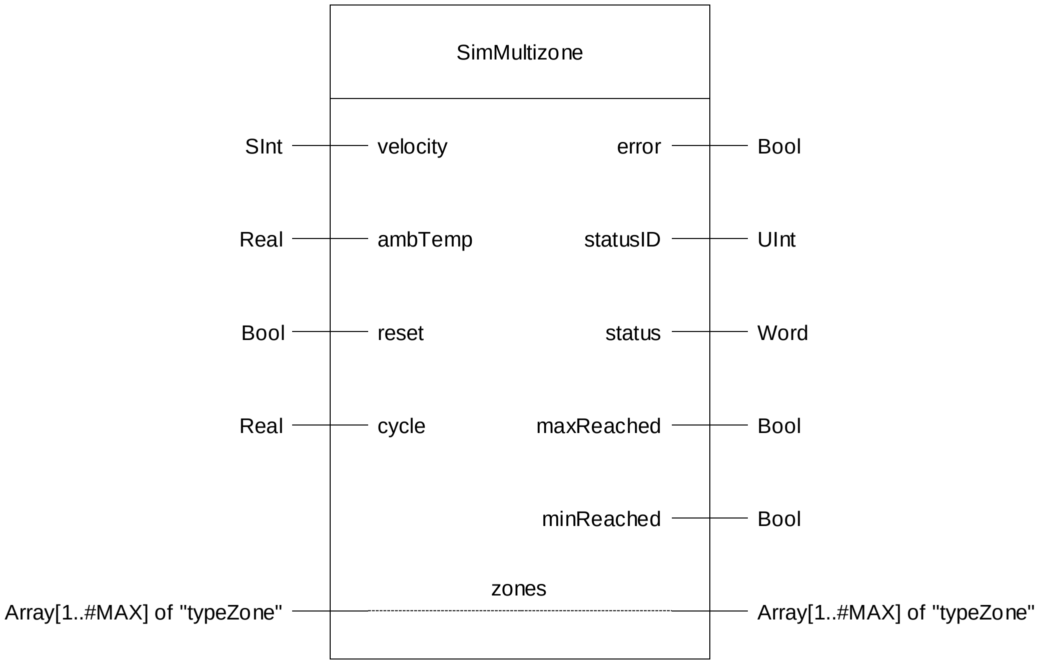
Name	P type	Data type	Comment
velocity	IN	SInt	Transport velocity of the extruder screw in %
ambTemp	IN	Real	Ambient temperature $T_{Ambient}$
reset	IN	Bool	Resets all relevant parameters including "outputX” to ‘0’.
cycle	IN	Real	Cycle time of the calling cyclic interrupt OB in seconds
error	OUT	Bool	FALSE: No errors TRUE: Block error, "statusID” specifies the error source, "status” specifies the error code.
statusID	OUT	UInt	Error source: Ten digits specify the zone, One digit specifies the subfunction: 1 = instZxHeat (LSim_PT1) 2 = instZxCool (LSim_PT1) 3 = instZxPT2 (LSim_PT2aper)
status	OUT	Word	Error code of the respective subfunction "LSim_PT1” or "LSim_PT2aper”
maxReached	OUT	Bool	For "maxReached” = TRUE at least one output tag

Name	P type	Data type	Comment
			"output” of the subfunctions "LSim_PT1” or "LSim_PT2aper” has been limited by the respective static parameter "maxOut”.
minReached	OUT	Bool	For "minReached” = TRUE at least one output tag "output” of the subfunctions "LSim_PT1” or "LSim_PT2aper” has been limited by the appropriate static parameter "minOut”.
zones	IN_OUT	Array[1..#MAX][1] of "typeZone"	PLC data type with the required interface parameters for each temperature zone (The size of the field is specified via the local constant "MAX”.)

Note

Detailed information on the subfuntions "LSim_PT1” and "LSim_PT2aper”used and their error codes ("status”) can be found in the Closed-Loop Control of Simulated Controlled Systems in the S7-1500 with PID_Compact V2 (\6\).

https://support.industry.siemens.com/cs/ww/en/view/79047707



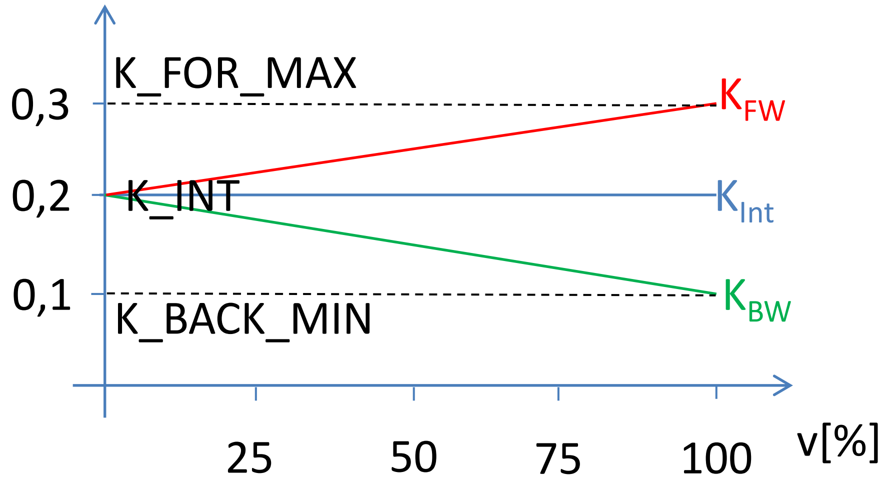
[1] The local constant “MAX” has to match the number of zones of the multi-zone controller.

Transport velocity

The coupling factors in the application example depend on the transport "velocity” (0..100%) of the extruder granulate.

Due to the increase of the transport velocity, an increase of the forward coupling and a reduction of the backward coupling is assumed.

Figure 4-5



Through the constants "K_BACK_MIN”, "K_INT” and "K_FOR_MAX” the influence of the transport velocity to coupling factors and therefore the energy flow, can be changed accordingly.

4.2.3 Configuring the FB "SimMultizone”

The controlled system properties of the simulated coupled temperature curves can be adjusted via the local constants of the FB "SimMultizone”.

Table 4-2

Name	Data type	Default value	Comment
MAX	DInt	3	Number of the zones
TMLAG_HEATER	Real	5.0	Time constant of the actuator delay heating (PT1) in seconds
TMLAG_COOLER	Real	10.0	Time constant of the actuator delay cooling (PT1) in seconds
GAIN_ZONE	Real	5.0	Gain factor temperature curve (PT2)
TMLAG1_ZONE	Real	20.0	Time constant 1 temperature curve (PT2)
TMLAG2_ZONE	Real	3.0	Time constant 2 temperature curve (PT2)
K_BACK_MIN	Real	0.1	Minimum coupling factor backward (to previous zone)
K_FOR_MAX	Real	0.3	Maximum coupling factor forward (to next zone)
K_INT	Real	0.2	Internal coupling factor (feedback)
PT2_MAX_OUT	Real	1000.0	Maximum output limit temperature curve (PT2)
PT2_MIN_OUT	Real	-1000.0	Minimum output limit temperature curve (PT2)

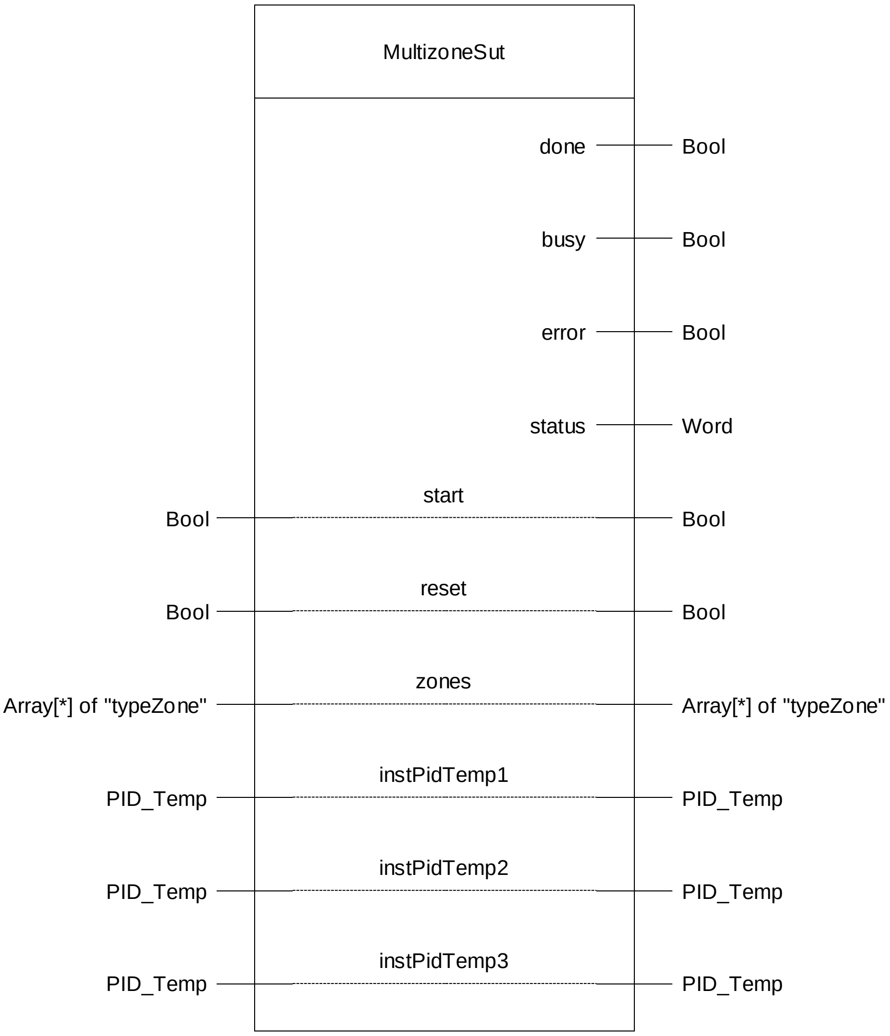
4.3 FB "MultizoneSut”

The FB "MultizoneSut” enables the simultaneous pretuning of all zones (first heating, then cooling with heating/cooling controllers).

It is called in the interrupt OB in which the controllers "PID_Temp” are also called.

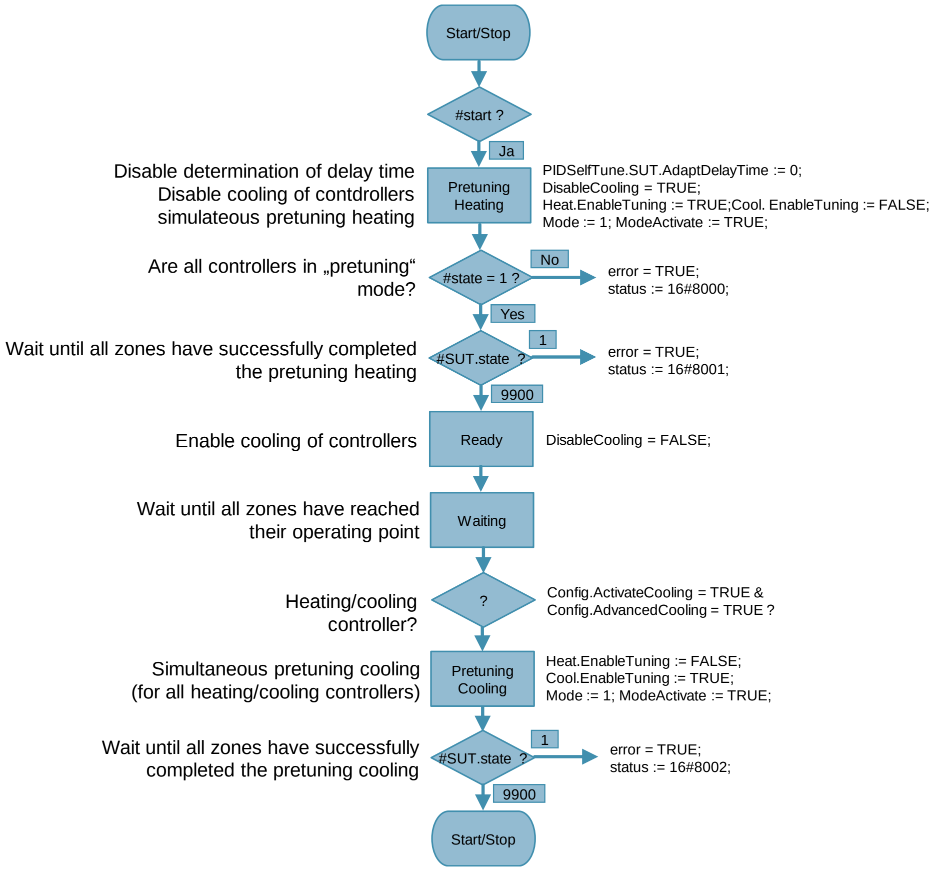
Table 4-3: Parameters of MultizoneSut

Name	P type	Data type	Comment
done	OUT	Bool	Error free processing of the block
busy	OUT	Bool	Block being processed
error	OUT	Bool	FALSE: No errors TRUE: Error in block, "status” specifies the error code.
status	OUT	Word	Error code: 0: no error 16#8000: At least for one zone controller the pretuning heating cannot be enabled. 16#8001: At least one zone controller has completed the pretuning heating with error. 16#8002: At least one zone controller has completed the pretuning cooling with error.
start	IN_OUT	Bool	Starts the processing and is reset after completion.
reset	IN_OUT	Bool	Resets the block and is set to FALSE after completion.
zones	IN_OUT	Array[*] of "typeZone"	PLC data type with the required interface parameters for each temperature zone (the size of the field is read.) Only the parameters "*[x].pidTemp” are accessed.
instPidTemp1	IN_OUT	PID_Temp	Technology instance transfer for zone 1
instPidTemp2	IN_OUT	PID_Temp	Technology instance transfer for zone 2
instPidTemp3	IN_OUT	PID_Temp	Technology instance transfer for zone 3


MultizoneSut
done
Bool
busy
Bool
error
Bool
status
Word
start
Bool
Bool
reset
Bool
Bool
zones
Array[*] of "typeZone"
Array[*] of "typeZone"
instPidTemp1
PID_Temp
PID_Temp
instPidTemp2
PID_Temp
PID_Temp
instPidTemp3
PID_Temp
PID_Temp

4.3.1 Program flow chart

Figure 4-6


Start/Stop
#start ?
Ja
Disable determination of delay time
Disable cooling of contdrollers
simulateous pretuning heating
Pretuning Heating
PIDSelfTune.SUT.AdaptDelayTime := 0;
DisableCooling = TRUE;
Heat.EnableTuning := TRUE;Cool. EnableTuning := FALSE;
Mode := 1; ModeActivate := TRUE;
Are all controllers in „pretuning“ mode?
#state = 1 ?
No
error = TRUE;
status := 16#8000;
Yes
Wait until all zones have successfully completed the pretuning heating
#SUT.state ?
1
error = TRUE;
status := 16#8001;
9900
Enable cooling of controllers
Ready
DisableCooling = FALSE;
Wait until all zones have reached their operating point
Waiting
Heating/cooling controller?
?
Config.ActivateCooling = TRUE &
Config.AdvancedCooling = TRUE ?
Simultaneous pretuning cooling (for all heating/cooling controllers)
Pretuning Cooling
Heat.EnableTuning := FALSE;
Cool.EnableTuning := TRUE;
Mode := 1; ModeActivate := TRUE;
Wait until all zones have successfully completed the pretuning cooling
#SUT.state ?
1
error = TRUE;
status := 16#8002;
9900
Start/Stop

Prerequisite for the pretuning heating is that the actual value is not too near the setpoint. This is why it is a good idea to start FB "MultizoneSut" when all controller instances "PID_Temp” are in the "inactive” operating state.

When setting the "start” input the simultaneous pretuning heating of the multi-zone controller is started with the required preconditions from chapter 3.1:

- Disabling the specification of the delay time
  PIDSelfTune.SUT.AdaptDelayTime:= 0
- Disabling the cooling of the controller: "DisableCooling” = TRUE
- Simultaneous pretuning for heating:
  Heat.EnableTuning := TRUE; Cool.EnableTuning := FALSE;
  Mode := 1; ModeActivate := TRUE;

When the controller instances do not report an error and the pretuning of all zones has been completed successfully, the cooling for all heating/cooling controllers is re-enabled ("DisableCooling” = FALSE).

Now it is waited until all zones have reached their operating point.

For all heating/cooling controllers the synchronous pretuning cooling is then started.

The synchronized pretuning of the multi-zone controller is stopped when it is successfully completed. This is continuously signaled by the "done” bit, until "start” is reset or the message is deleted with "reset”.

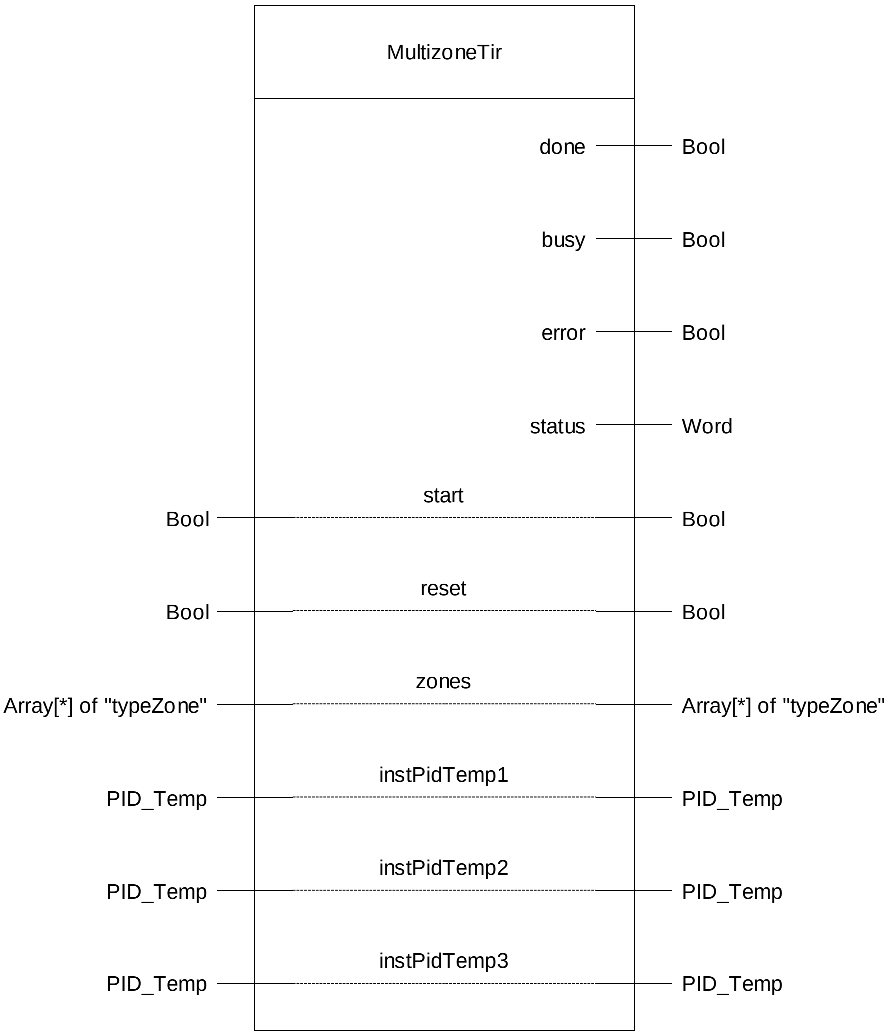
4.4 FB "MultizoneTir”

The FB "MultizoneTir” enables the synchronized fine tuning of selected zones with energy type specification (heating or cooling).

It is called in the interrupt OB in which the controllers "PID_Temp” are also called.

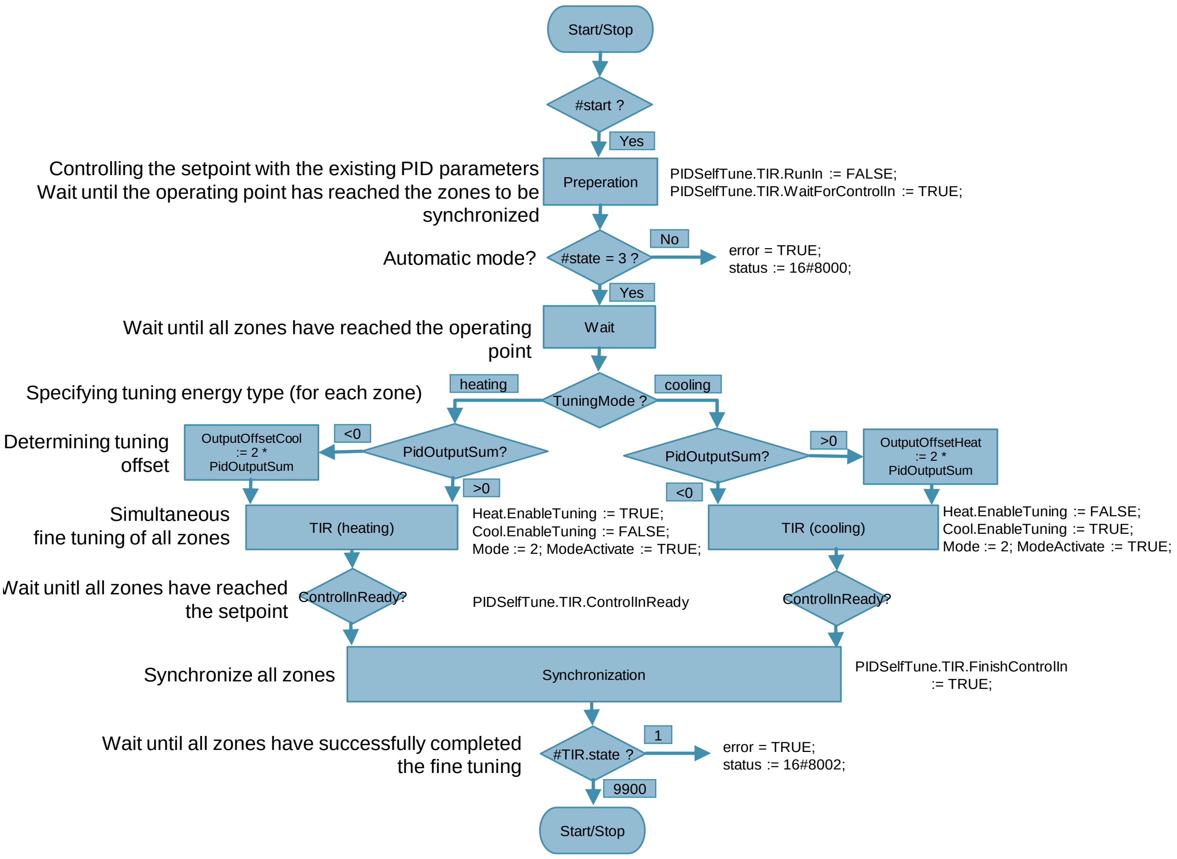
Table 4-4: Parameters of MultizoneTir

Name	P type	Data type	Comment
done	OUT	Bool	error free processing of the block
busy	OUT	Bool	Block being processed
error	OUT	Bool	FALSE: no error TRUE: Error in block, "status” specifies the error code.
status	OUT	Word	Error code: 0 no error, 16#8000: at least one zone controller to be optimized is in "automatic” mode 16#8001: at least one zone controller reports an error 16#8002: at least one zone controller has stopped the fine tuning with error. 16#8003: at least one zone controller did not return to automatic mode after fine tuning
start	IN_OUT	Bool	starts the processing and is reset after completion.
reset	IN_OUT	Bool	resets the block and is set to FALSE after completion
zones	IN_OUT	Array[*] of "typeZone"	PLC data type with the required interface parameters for controller calls
instPidTemp1	IN_OUT	PID_Temp	Technology instance transfer for zone 1
instPidTemp2	IN_OUT	PID_Temp	Technology instance transfer for zone 2
instPidTemp3	IN_OUT	PID_Temp	Technology instance transfer for zone 3


MultizoneTir
done
Bool
busy
Bool
error
Bool
status
Word
start
Bool
Bool
reset
Bool
Bool
zones
Array[*] of "typeZone"
Array[*] of "typeZone"
instPidTemp1
PID_Temp
PID_Temp
instPidTemp2
PID_Temp
PID_Temp
instPidTemp3
PID_Temp
PID_Temp

4.4.1 Program flow chart

Figure 4-7


Start/Stop
#start ?
Yes
Controlling the setpoint with the existing PID parameters
Wait until the operating point has reached the zones to be synchronized
Preperation
PIDSelfTune.TIR.RunIn := FALSE;
PIDSelfTune.TIR.WaitForControlIn := TRUE;
Automatic mode?
#state = 3 ?
No
error = TRUE;
status := 16#8000;
Yes
Wait until all zones have reached the operating point
Wait
Specifying tuning energy type (for each zone)
heating
TuningMode ?
cooling
Determining tuning offset
OutputOffsetCool := 2 * PidOutputSum
<0
PidOutputSum?
>0
PidOutputSum?
>0
OutputOffsetHeat := 2 * PidOutputSum
<0
Simultaneous fine tuning of all zones
TIR (heating)
Heat.EnableTuning := TRUE;
Cool.EnableTuning := FALSE;
Mode := 2; ModeActivate := TRUE;
TIR (cooling)
Heat.EnableTuning := FALSE;
Cool.EnableTuning := TRUE;
Mode := 2; ModeActivate := TRUE;
Wait unitl all zones have reached the setpoint
ControlInReady?
PIDSelfTune.TIR.ControlInReady
ControlInReady?
Synchronize all zones
Synchronization
PIDSelfTune.TIR.FinishControlIn := TRUE;
Wait until all zones have successfully completed the fine tuning
#TIR.state ?
1
error = TRUE;
status := 16#8002;
9900
Start/Stop

Before starting the synchronous fine tuning, specify the energy type to be optimized for each zone controller via the InOut tag "zones[x].tuningMode”:

- 0: No fine tuning
- 1: Heating
- 2: Cooling

All controllers where fine tuning ("tuningMode” = 1 or 2) is to be performed have to be in automatic mode as a precondition.

Setting the "start” input creates the required preconditions for synchronizing several fine tunings from chapter 3.2.2:

- It is controlled to the setpoint with the exiting PID parameters: "PIDSelfTune.TIR.RunIn” = FALSE
- During the fine tuning it is waited that the setpoints of the other controllers to be optimized are reached before the standard deviation is calculated: "PIDSelfTune.TIR.WaitForControlIn” = TRUE

When the controllers to be optimized are in automatic mode, it is waited until they have reached their operating point. This is required to determine the energy type of the controller in the operating point.

If a controller is not in the selected energy type, an according offset is calculated and output on the opposite output of the controller in order to get to the selected energy type.

Example:
The controller is in heating mode, but it is to carry out the fine tuning cooling. Therefore an additional offset is set on the controller’s heating output so that the controller has to cool to reach the setpoint.

Afterwards the fine tunings for each zone are started.

When all controllers to be optimized have reached their setpoint, the actual fine tuning of all zones start synchronously with the respective calculation of the standard deviation.

The synchronized fine tuning of the multi-zone controller is concluded when it is successfully completed. This is continuously signaled by the "done” bit, until "start” is reset or the message is deleted with "reset”.

4.5 FB "Waiting”

When the specified controller difference tolerance of the "PID_Temp” instances is maintained, the FB "Waiting” waits for a specified time (number of cycles).

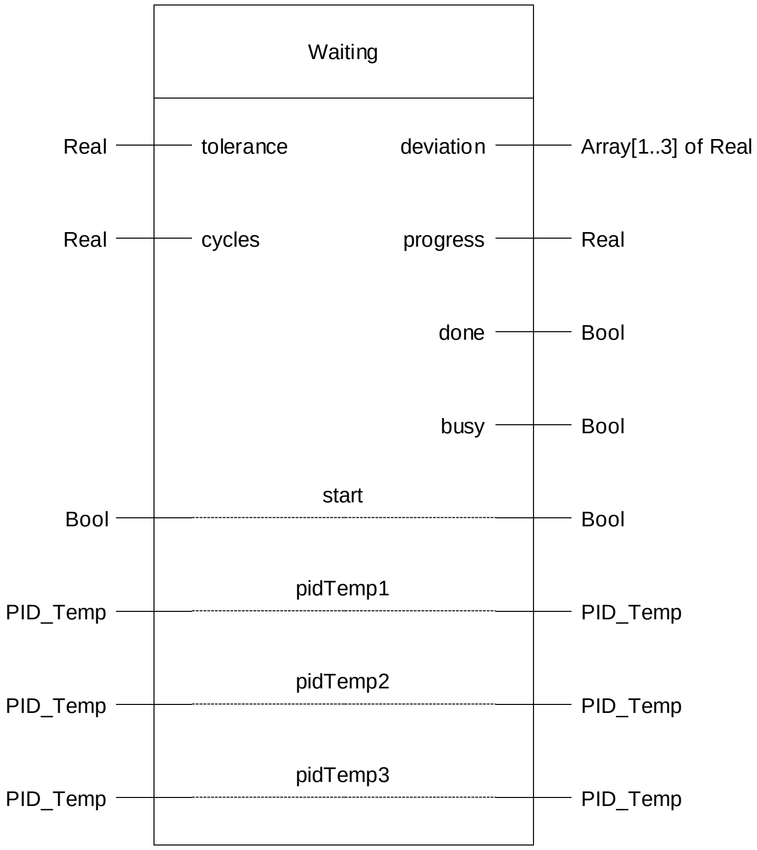
It is used in FB "MultizoneSut” and in FB "MultizoneTir”.

Table 4-5: Parameters of waiting

Name	P type	Data type	Comment
tolerance	IN	Real	Tolerance specification of the control differences in %
cycles	IN	Real	Specification of the number of waiting cycles
deviation	OUT	Array[1..#MAX][2] of Real	Control differences of the "PID_Temp” instances (the size of the field is specified via the local constant "MAX”.)
progress	OUT	Real	Progress of the wait time in %
done	OUT	Bool	Error free processing of the block
busy	OUT	Bool	Block being processed
start	IN_OUT	Bool	Starts the processing and is reset after completion.
pidTemp1	IN_OUT	PID_Temp	Technology instance transfer for zone 1
pidTemp2	IN_OUT	PID_Temp	Technology instance transfer for zone 2
pidTemp3	IN_OUT	PID_Temp	Technology instance transfer for zone 3

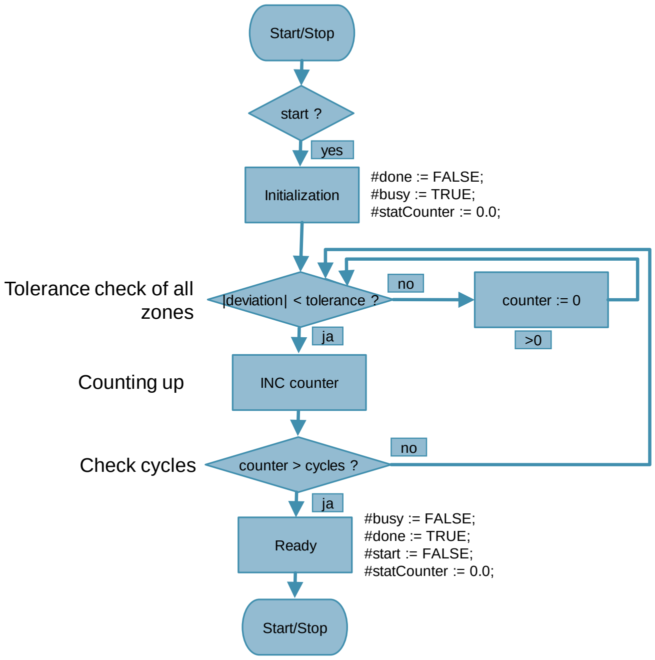
[2] The local constant “MAX” has to match the number of zones of the multi-zone controller.

4.5 FB "Waiting”



4.5.1 Program flow chart

Figure 4-8



When starting the FB "Waiting” the initialization is performed:

- #done := FALSE;
- #busy := TRUE;
- #statCounter := 0.0;

In each cycle of the interrupt the percentage of the control deviation for each zone z is calculated via the following formula:

$$deviation_z = \frac{setpoint_z - input_z}{setpoint_z} * 100$$

The cycle counter "counter” is incremented if all the amounts of the control deviations are within the proportional "tolerance” specification. Otherwise the counter is reset.

FB "Waiting” is completed if the specified number of cycles "cycles” are exceeded. This is continuously signaled by the "done” bit, until "start” is reset.

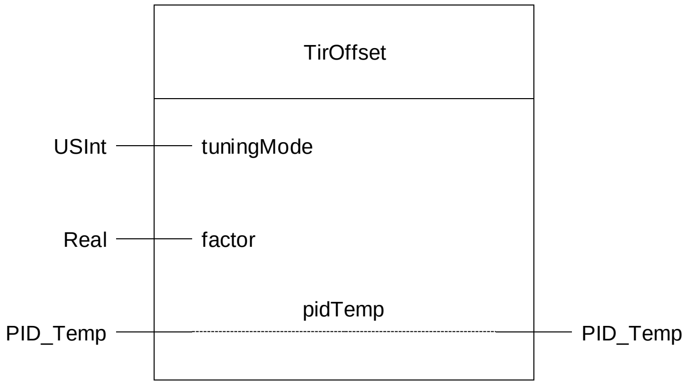
4.6 FC "TirOffset”

The FC "TirOffset” calculates the tuning offset for heating/cooling controller "PID_Temp” to enable fine tuning for the opposite energy type (heating or cooling).

It is called in FB "MultizoneTir” for each zone.

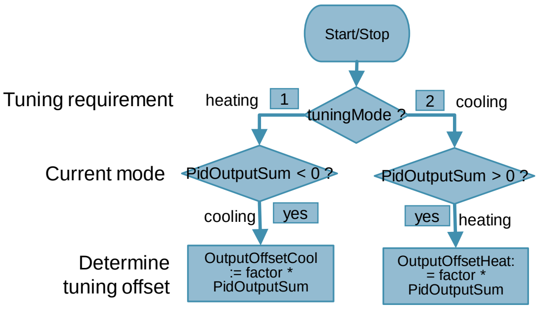
Table 4-6: Parameters of TirOffset

Name	P type	Data type	Comment
tuningMode	IN	USInt	Fine tuning energy type: 0 = none, 1 = heating, 2 = cooling
factor	IN	Real	Multiplier (Offset = factor * PidOutputSum); must be >1
pidTemp	IN_OUT	PID_Temp	Technology instance transfer for the respective zone



4.6.1 Program flow chart

Figure 4-9



Example:
The controller is to carry out the fine tuning heating, but is in cooling mode. This is why a higher offset is returned in the cooling output so that the controller has to heat in order to reach the setpoint and to determine the parameters for heating from the oscillation.
The offset on the controller is reset after completing the fine tuning.

4.7 FC "TirTuningMode”

FC "TirTuningMode” sets the tuning bits according to the desired energy type of the fine tuning according to Table 4-8.

It is called in FB "MultizoneTir” for each zone.

Table 4-7: Parameter of TirTuningMode

Name	P type	Data type	Comment
tuningMode	IN	USInt	Fine tuning energy type: 0 = none, 1 = heating, 2 = cooling
pidTemp	IN_OUT	PID_Temp	Technology instance transfer for the respective zone

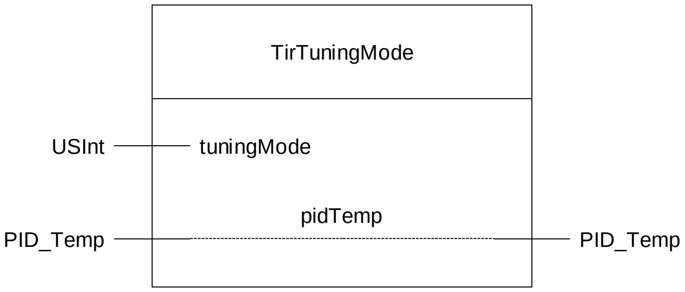


Table 4-8

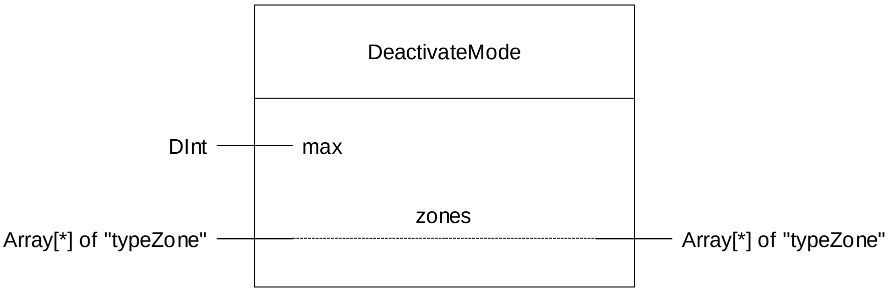
tuningMode	Heat.EnableTuning	Cool.EnableTuning
0 (none)	FALSE	FALSE
1 (heating)	TRUE	FALSE
2 (cooling)	FALSE	TRUE

4.8 FC "DeactivateMode”

FC "DeactivateMode” resets the input bit "ModeActivate” of all "PID_Temp” calls. It is used in FB "MultizoneSut” as well as in FB "MultizoneTir”.

Table 4-9: Parameters of DeactivateMode

Name	P type	Data type	Comment
max	IN	DInt	Number of the zones
zones	IN_OUT	Array[*] of "typeZone"	PLC data type with the required interface parameters for each temperature zone (the size of the field is read) Only the parameters "*[x].pidTemp.modeActivate” are accessed.



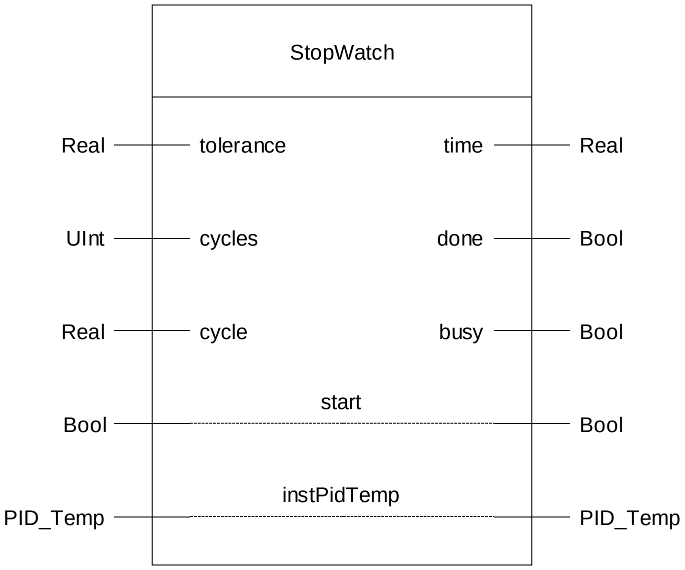
4.9 FB "StopWatch”

The auxiliary function FB "StopWatch” measures the recovery time of disturbance variables.

It is called in the interrupt OB in which the controller "PID_Temp” is also called.

Table 4-10: Parameters of StopWatch

Name	P type	Data type	Comment
tolerance	IN	Real	Tolerance specification of the control differences in %
cycles	IN	UInt	Specification of the number of waiting cycles
cycle	IN	Real	Cycle time of the calling cyclic interrupt OB (in seconds)
time	OUT	Real	Recovery time in seconds
done	OUT	Bool	Error free processing of the block
busy	OUT	Bool	Block being processed
start	IN_OUT	Bool	Starts the processing and is reset after completion
instPidTemp	IN_OUT	PID_Temp	Technology instance transfer



After starting the FB "StopWatch” the block will wait until the proportional control deviation leaves the tolerance band "tolerance”. From this time on the "recoveryCycles” are counted until the control deviation is again within the tolerance band for the number of cycles waited after presenting the FB "Waiting”.

As a result, the recovery time "time” is calculated:

$$time = (recoveryCycles \ - cycles) \ * cycle$$

4.10 Data block/PLC data types

4.10.1 DB "Tags”

The data block "Tags” includes the parameters to transfer the block interfaces and looks as follows:

Table 4-11

Name	Data type	Description
simMultizone	"typeSimMultizoneIn"	Includes the individual parameters for transferring the interface to the FB "SimMultizone”
multizoneSut	"typeMultizoneIn"	Includes the input parameters for the pretuning of the multi-zone controller (FB "MultizoneSut")
multizoneTir	"typeMultizoneIn"	Includes the input parameters for the fine tuning of the multi-zone controller (FB "MultizoneTir")
zones	Array[1..3] [3] of "typeZone"	Includes interface parameters for the transfer to the block calls, depending on the number of zones.
stopWatch	"typeWaitingIn"	Includes the interface parameters for the recovery time calculation

4.10.2 "typeSimMultizoneIn"

The PLC data type "typeSimMultizoneIn” includes the individual parameters for the interface transfer to the FB "SimMultizone” (independent from the number of zones).

Table 4-12

Name	Data type	Description
velocity	SInt	Transport velocity of the extruder screw in %
ambTemp	Real	Ambient temperature
reset	Bool	Resets all relevant parameters

4.10.3 "typeMultizoneIn"

The PLC data type "typeMultizoneIn” includes the input parameters for transferring the interface to the FB "MultizoneSut” or FB "MultizoneTir”.

Table 4-13

Name	Data type	Description
start	Bool	Starts the processing and is reset after completion.
reset	Bool	Resets the block and is set to FALSE after completion.

[3] The upper limit of the field has to match the number of zones of the multi-zone controller.

4.10.4 "typeZone”

PLC data type "typeZone" includes the parameters of the block interfaces for each temperature zone.

Table 4-14

Name	Data type	Description
pidTemp	"typePidTemp"	PLC data type with the required interface parameters for the "PID_Temp” controller call.
simMultizone	"typeSimMultizoneArray"	PLC data type with the input parameters for temperature simulation (FB "SimMultizone”).
tuningMode	USInt	Energy type specification for fine tuning (0 = none, 1 = heating, 2 = cooling).

4.10.5 "typePidTemp"

PLC data type "typePidTemp” includes the required interface parameters for the "PID_Temp” controller call.

Table 4-15

Name	Data type	Description
setpoint	Real	Setpoint specification
mode	Int	Mode selection
modeActivate	Bool	Release mode
outputHeat	Real	Output value heating in REAL format
outputCool	Real	Output value cooling in REAL format

4.10.6 "typeSimMultizoneArray"

PLC data type "typeSimMultizoneArray” includes interface parameters for the temperature simulation (FB "SimMultizone”) depending on the number of zones.

Table 4-16

Name	Data type	Description
disturbance	Real	Disturbance variable on the input
output	Real	Simulated zone temperature

4.10.7 "typeWaitingIn”

PLC data type "typeWaitingIn" includes the input parameters for the FBs "Waiting” and "StopWatch”.

Table 4-17

Name	Data type	Description
start	Bool	Starts the processing and is reset after completion.
tolerance	Real	Tolerance specification of the control differences in %
cycles	UInt	Specification of the number of waiting cycles

5 Configuration and Settings

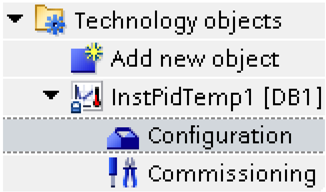
This chapter describes the steps necessary to tailor the example project to your applications.

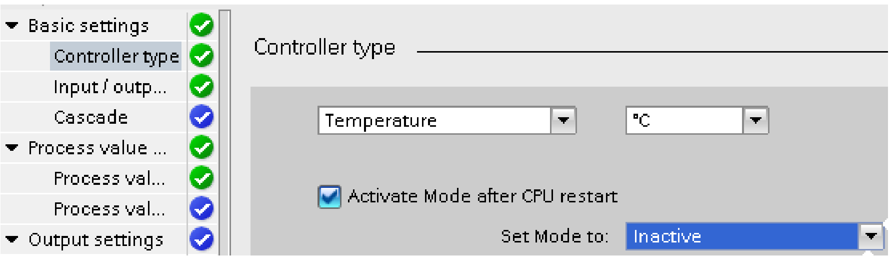
5.1 Configuring the technology "PID_Temp”

The storage locations of the controller block calls "PID_Temp” are created as individual instances. As a result, each controller can be conveniently configured as follows, via the appropriate wizard in the technology object.

Table 5-1

<table>
<tr><th>No.</th><th>Action</th></tr>
<tr><td>1.</td><td>Open the configuration editor of the "PID_Temp” individual instance to be configured in "Technology objects” in STEP 7 (TIA Portal).


</td></tr>
<tr><td>2.</td><td>Open the "Controller type” menu in the basic settings and select the temperature unit for the display in the commissioning window.



Temperature °C

☑ Activate Mode after CPU restart

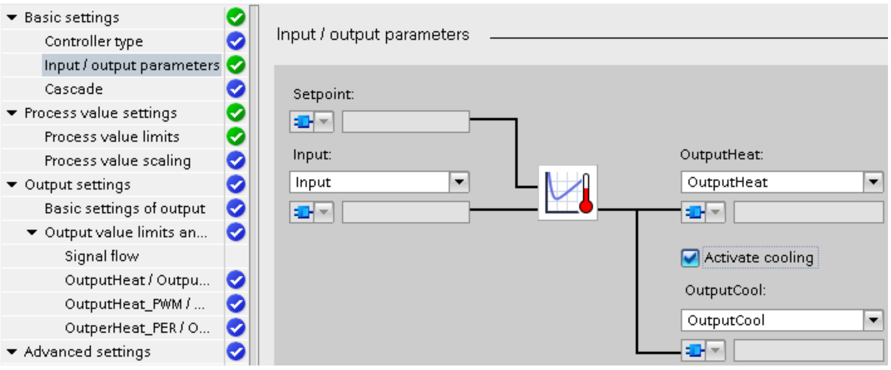
Set Mode to: Inactive

Enable the specification of the mode for the first start und select "Inactive”.</td></tr>
<tr><td>3.</td><td>Open the input/output parameter and specify whether the controller is a heating or heating/cooling controller ("Activate cooling”).

Select the signal course for each interface (floating-point number/analog/PWM).

For more information see Input or Output Value Heating or Cooling (\3\).

Function Manual: SIMATIC S7-1200, S7-1500 PID control
https://support.industry.siemens.com/cs/ww/en/view/108210036



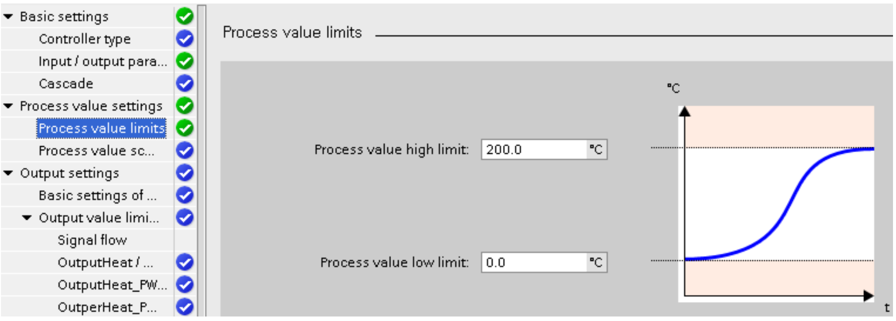
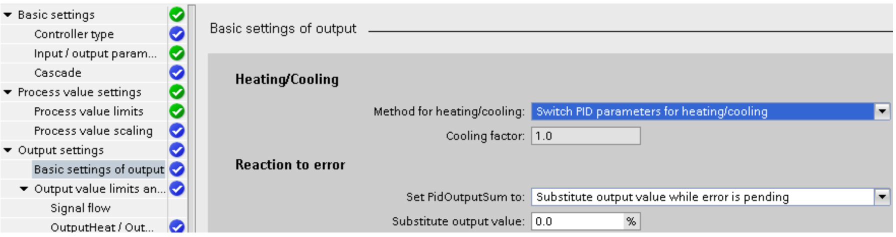
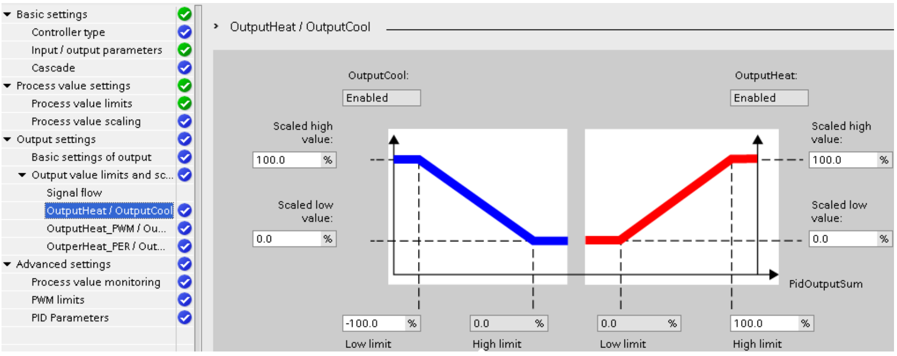
Input: Input

OutputHeat: OutputHeat

☑ Activate cooling

OutputCool: OutputCool

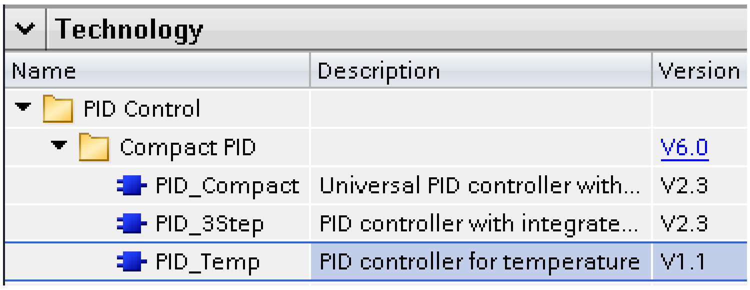
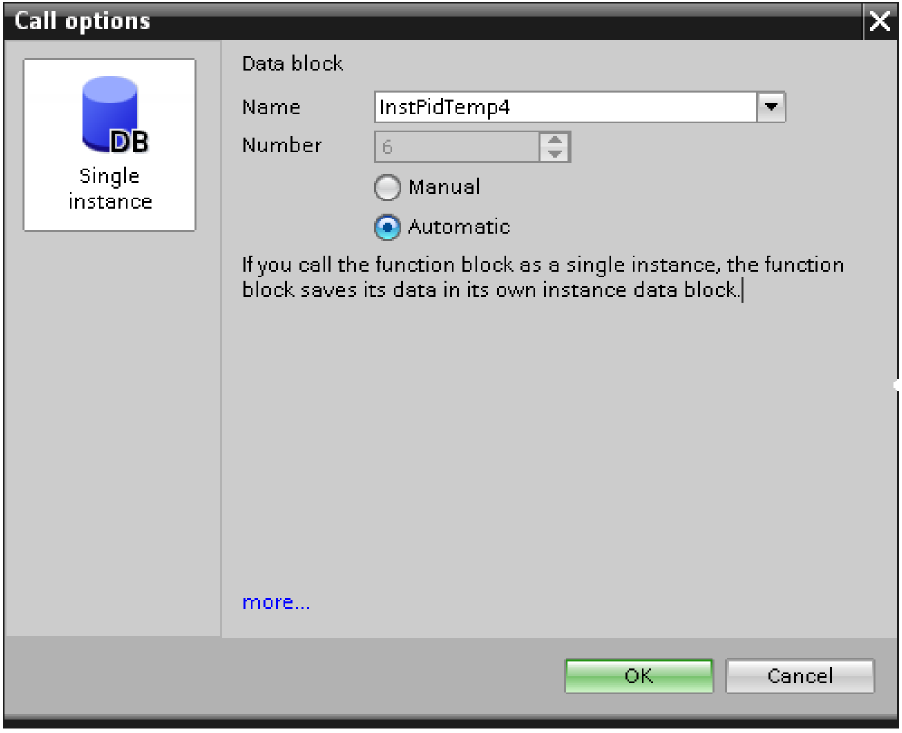
When using the simulation block "SimMultizone” use input as floating-point number ("Input”). The manipulated variable as floating-point number is also output when selecting "OutputX_PWM” or "OutputX_PER (analog)”.</td></tr>
</table>

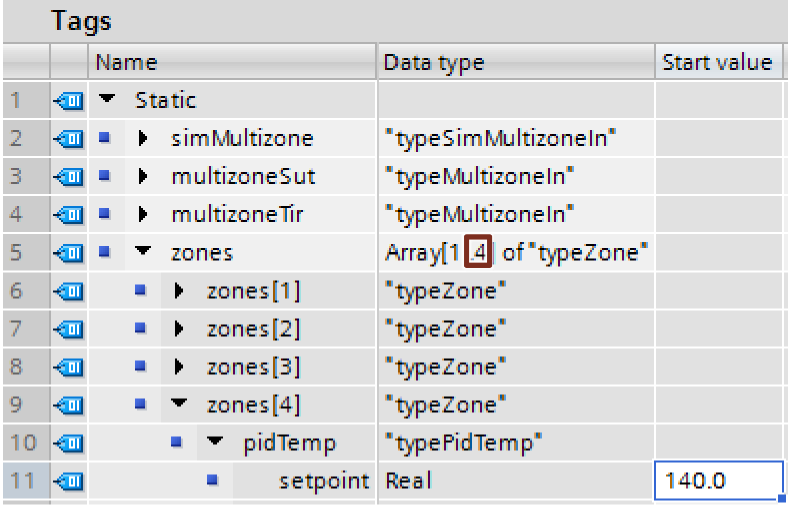
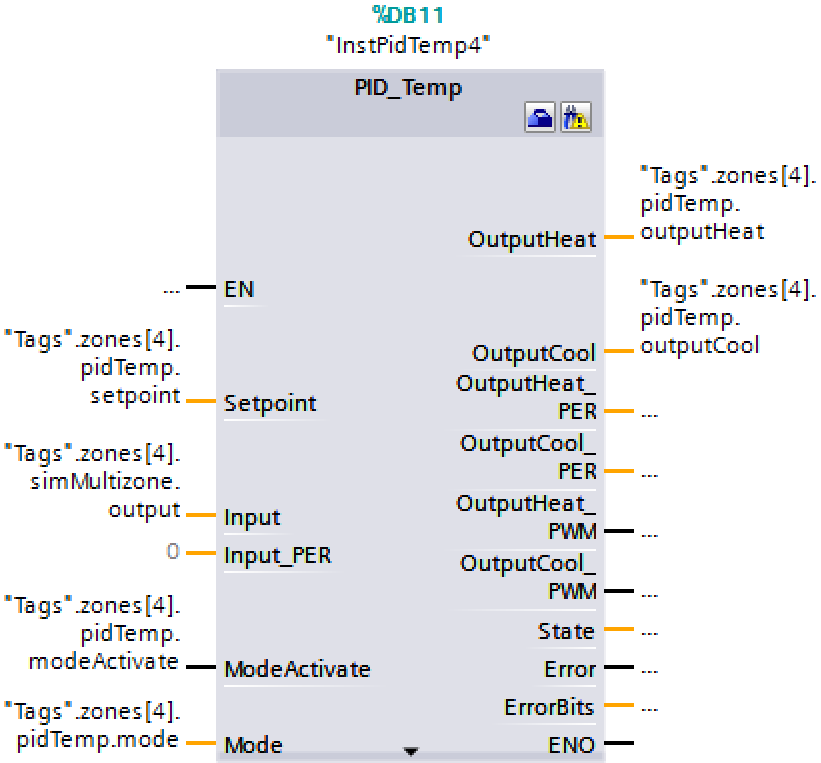
No.	Action
4.	Specify the actual value limits in the actual value settings in such a way that they are not violated. Otherwise, in the event of a violation, the tuning is cancelled.  Basic settings Controller type Input / output para... Cascade Process value settings Process value limits Process value sc... Output settings Basic settings of ... Output value limi... Signal flow OutputHeat / ... OutputHeat_PW... OutperHeat_P... °C t Process value limits Process value high limit: 200.0 °C Process value low limit: 0.0 °C Specify the actual value scaling if you are using the analog input signal "Input_PER".
5.	Open the Basic settings of output and select the "Switch PID parameter for heating/cooling" as method for heating/cooling controllers. Only when this method is selected, is pretuning and fine tuning for cooling made available.  Basic settings Controller type Input / output param... Cascade Process value settings Process value limits Process value scaling Output settings Basic settings of output Output value limits an... Signal flow OutputHeat / Out... Basic settings of output Heating/Cooling Method for heating/cooling: Switch PID parameters for heating/cooling Cooling factor: 1.0 Reaction to error Set PidOutputSum to: Substitute output value while error is pending Substitute output value: 0.0 % Specify the response in the event of an error
6.	Make the settings for Output value limits and scaling when selecting "OutputX_PWM" or "OutputX_PER (analog)".  Basic settings Controller type Input / output parameters Cascade Process value settings Process value limits Process value scaling Output settings Basic settings of output Output value limits and sc... Signal flow OutputHeat / OutputCool OutputHeat_PWM / Ou... OutperHeat_PER / Out... Advanced settings Process value monitoring PWM limits PID Parameters PidOutputSum OutputHeat / OutputCool OutputCool: Enabled OutputHeat: Enabled Scaled high value: 100.0 % / 100.0 % Scaled low value: 0.0 % / 0.0 % Low limit: -100.0 %, High limit: 0.0 % Low limit: 0.0 %, High limit: 100.0 % Make the settings for PWM limits in the advanced settings if you have selected the pulse width modulation output signal "OutputX_PWM".

5.2 Extension of control zones

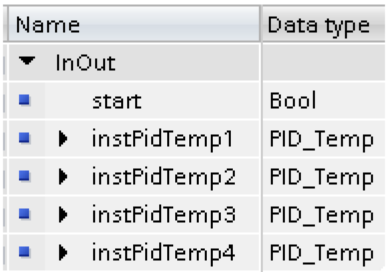
Proceed as follows to expand the number of control zones of the application example.

Table 5-2

No.	Action
1.	Call the technology instruction "PID_Temp" V1.1 in the interrupt.  Technology Name | Description | Version PID Control Compact PID | | V6.0 PID_Compact | Universal PID controller with... | V2.3 PID_3Step | PID controller with integrate... | V2.3 PID_Temp | PID controller for temperature | V1.1
2.	Select the individual instance as call option  Call options Single instance Data block Name: InstPidTemp4 Number: 6 ○ Manual ● Automatic If you call the function block as a single instance, the function block saves its data in its own instance data block. more... OK Cancel Configure the technology object according to Table 5-1.

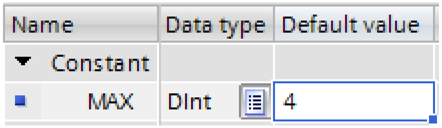
<table>
<tr><th>No.</th><th>Action</th></tr>
<tr><td>3.</td><td>Adjust the upper limit of the "zones" tag in DB "Tags".

Tags
<table><tr><th></th><th>Name</th><th>Data type</th><th>Start value</th></tr><tr><td>1</td><td>Static</td><td></td><td></td></tr><tr><td>2</td><td>simMultizone</td><td>"typeSimMultizoneIn"</td><td></td></tr><tr><td>3</td><td>multizoneSut</td><td>"typeMultizoneIn"</td><td></td></tr><tr><td>4</td><td>multizoneTir</td><td>"typeMultizoneIn"</td><td></td></tr><tr><td>5</td><td>zones</td><td>Array[1..4] of "typeZone"</td><td></td></tr><tr><td>6</td><td>zones[1]</td><td>"typeZone"</td><td></td></tr><tr><td>7</td><td>zones[2]</td><td>"typeZone"</td><td></td></tr><tr><td>8</td><td>zones[3]</td><td>"typeZone"</td><td></td></tr><tr><td>9</td><td>zones[4]</td><td>"typeZone"</td><td></td></tr><tr><td>10</td><td>pidTemp</td><td>"typePidTemp"</td><td></td></tr><tr><td>11</td><td>setpoint</td><td>Real</td><td>140.0</td></tr></table>
Specify the setpoint for the new zone as start value.</td></tr>
<tr><td>4.</td><td>Switch the input interfaces in accordance with the other "PID_Temp" calls.

%DB11
"InstPidTemp4"
PID_Temp
... — EN
"Tags".zones[4].pidTemp.setpoint — Setpoint
"Tags".zones[4].simMultizone.output — Input
0 — Input_PER
"Tags".zones[4].pidTemp.modeActivate — ModeActivate
"Tags".zones[4].pidTemp.mode — Mode
OutputHeat — "Tags".zones[4].pidTemp.outputHeat
OutputCool — "Tags".zones[4].pidTemp.outputCool
OutputHeat_PER — ...
OutputCool_PER — ...
OutputHeat_PWM — ...
OutputCool_PWM — ...
State — ...
Error — ...
ErrorBits — ...
ENO</td></tr>
</table>

5. Add an InOut parameter of data type "PID_Temp" to the FBs "MultizoneSut", "multizoneTir" and "Waiting".



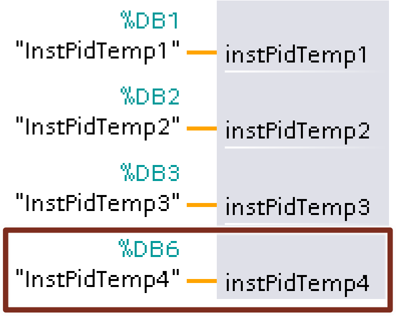
Name	Data type
▼ InOut	
■ start	Bool
■ ▶ instPidTemp1	PID_Temp
■ ▶ instPidTemp2	PID_Temp
■ ▶ instPidTemp3	PID_Temp
■ ▶ instPidTemp4	PID_Temp

6. Adjust the local constant "MAX" of the FBs "Waiting" and "SimMultizone" to the number of zones.



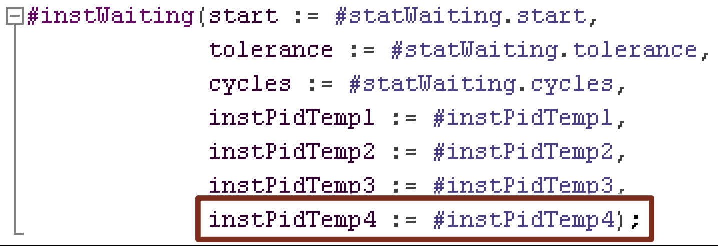
Name	Data type	Default value
▼ Constant		
■ MAX	DInt	4

7. Update the block call and switch the newly created individual instance of the "PID_Temp" to the InOut parameter of the FBs "MultizoneSut" and "MultizoneTir"



```
         %DB1
"InstPidTemp1" — instPidTemp1
         %DB2
"InstPidTemp2" — instPidTemp2
         %DB3
"InstPidTemp3" — instPidTemp3
         %DB6
"InstPidTemp4" — instPidTemp4
```

and loop through the call of the FB "Waiting" to the FBs "MultizoneSut" and "MultizoneTir".



```
#instWaiting(start := #statWaiting.start,
             tolerance := #statWaiting.tolerance,
             cycles := #statWaiting.cycles,
             instPidTemp1 := #instPidTemp1,
             instPidTemp2 := #instPidTemp2,
             instPidTemp3 := #instPidTemp3,
             instPidTemp4 := #instPidTemp4);
```

Furthermore, you have to expand the program code by the added zone, according to the comments in the FBs "MultizoneSut", "MultizoneTir" and "Waiting".

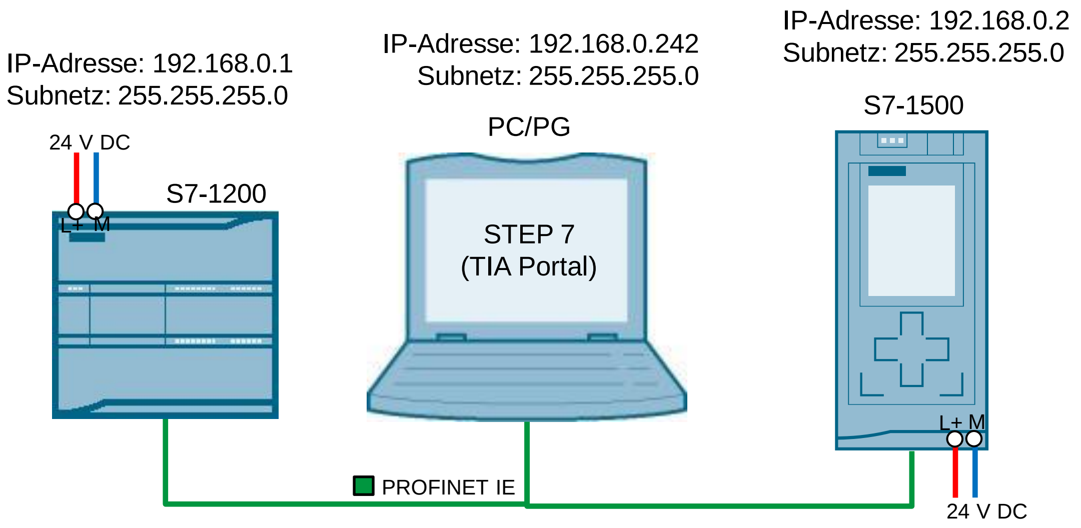
6 Installation and Commissioning

This chapter describes the steps necessary for commissioning the example project with the hardware and software used (from chapter 2.2.2).

6.1 Installing the hardware

The figure below shows the hardware configuration of the application:

Figure 6-1



Note

Always follow the installation guidelines in order to connect all the components.

- Manual SIMATIC S7-1200 (\4\)
- Manual SIMATIC S7-1500 (\8\)

Table 6-1

No.	Action	Remark
8.	Wire and connect the SIMATIC S7-1200 or S7-1500 as described.	See S7-1200 manual (\4\) or S7-1500 manual (\8\)
9.	Plug the empty memory card into the S7-1500 CPU (Table 2-1).	See Chapter "Plug/Pull SIMATIC Memory Card on the CPU" (S7-1500 manual \8\)
10.	Connect the SIMATIC PM 1207 power supply module to the low voltage network (230 V).	-

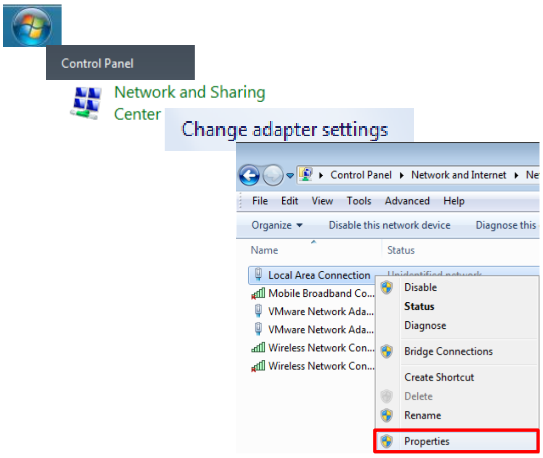
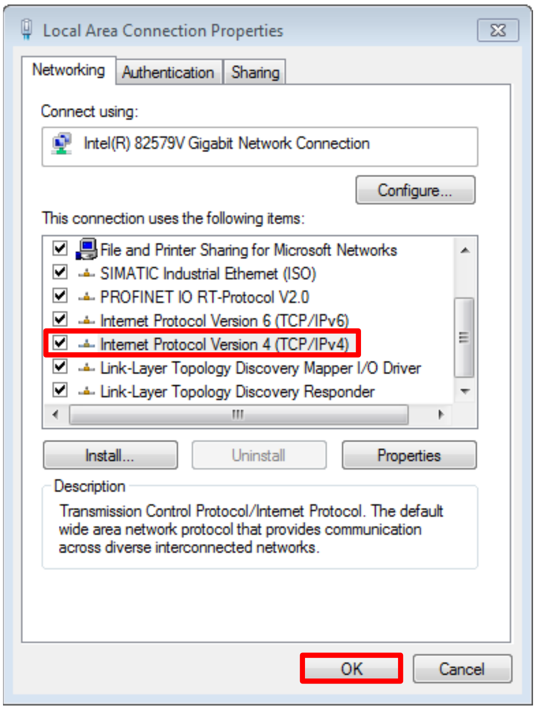
6.2 Commissioning

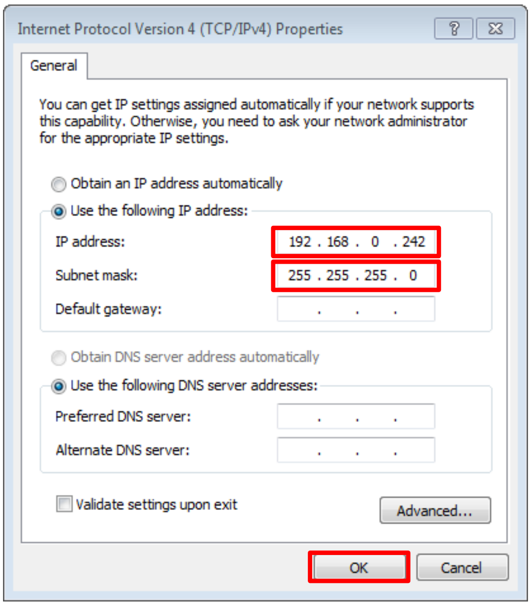
This chapter describes the steps for installing the sample code.

6.2.1 Network connections

The LAN network card of the programming device requires a static IP address to configure the controller.
The configuration of the LAN connection is described below.

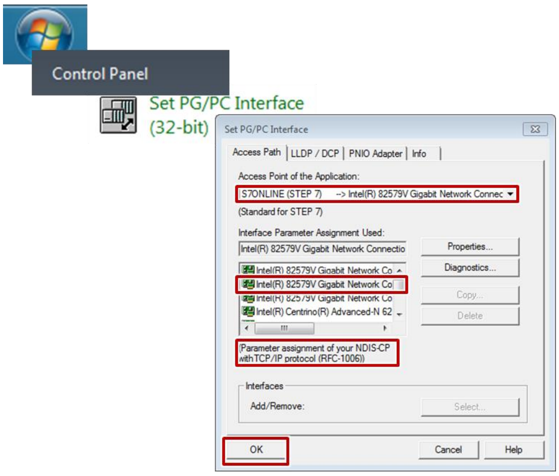
Table 6-2

No.	Action	Remark
11.	Click "Start > Control Panel > Network and Sharing Center > Change adapter settings" to open the network connections. - Select your network connection. - Right-click to open the properties.	 Control Panel Network and Sharing Center Change adapter settings Control Panel ▸ Network and Internet ▸ Ne File Edit View Tools Advanced Help Organize ▾ Disable this network device Diagnose this Name Status Local Area Connection Unidentified network Mobile Broadband Co... VMware Network Ada... VMware Network Ada... Wireless Network Con... Wireless Network Con... Disable Status Diagnose Bridge Connections Create Shortcut Delete Rename Properties
12.	Select the "Internet Protocol Version 4 (TCP/IPv4)" element in "Networking" and open the properties.	 Local Area Connection Properties Networking | Authentication | Sharing Connect using: Intel(R) 82579V Gigabit Network Connection Configure... This connection uses the following items: ☑ File and Printer Sharing for Microsoft Networks ☑ SIMATIC Industrial Ethernet (ISO) ☑ PROFINET IO RT-Protocol V2.0 ☑ Internet Protocol Version 6 (TCP/IPv6) ☑ Internet Protocol Version 4 (TCP/IPv4) ☑ Link-Layer Topology Discovery Mapper I/O Driver ☑ Link-Layer Topology Discovery Responder Install... Uninstall Properties Description Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks. OK Cancel

No.	Action	Remark
13.	- Select "Use the following IP address". - Select an IP address in the CPU's subnet mask. - Confirm the settings with "OK" and "Close".	 Internet Protocol Version 4 (TCP/IPv4) Properties General You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings. ○ Obtain an IP address automatically ● Use the following IP address: IP address: 192 . 168 . 0 . 242 Subnet mask: 255 . 255 . 255 . 0 Default gateway: . . . ○ Obtain DNS server address automatically ● Use the following DNS server addresses: Preferred DNS server: . . . Alternate DNS server: . . . □ Validate settings upon exit Advanced... OK Cancel

6.2.2 Setting PG/PC interface

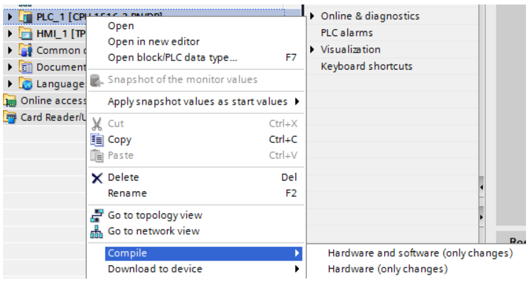
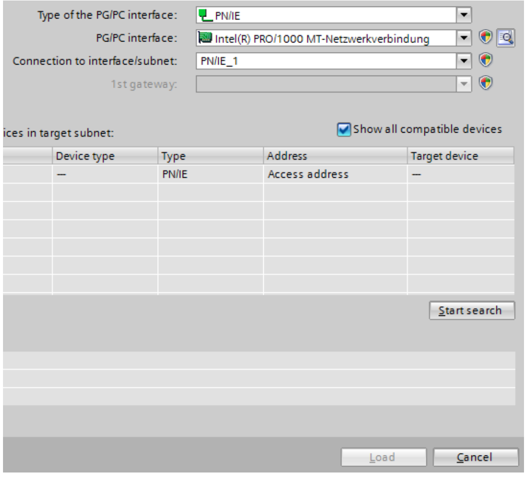
Table 6-3

No.	Action	Remark
14.	Open the PG/PC interface settings via "Start > Control Panel" to set the correct access path for STEP 7. - Select "S7ONLINE (STEP 7)" as the application's access point. - Select your network card with "Parameter assignment of your NDIS CP with TCP/IP protocol (RFC-1006)" as the interface configuration used. Confirm the settings with "OK".	 Control Panel Set PG/PC Interface (32-bit) Set PG/PC Interface Access Path | LLDP / DCP | PNIO Adapter | Info Access Point of the Application: S7ONLINE (STEP 7) --> Intel(R) 82579V Gigabit Network Connec (Standard for STEP 7) Interface Parameter Assignment Used: Intel(R) 82579V Gigabit Network Connectio Intel(R) 82579V Gigabit Network Co Intel(R) 82579V Gigabit Network Co Intel(R) 82579V Gigabit Network Co Intel(R) Centrino(R) Advanced-N 62 Properties... Diagnostics... Copy... Delete (Parameter assignment of your NDIS-CP with TCP/IP protocol (RFC-1006)) Interfaces Add/Remove: Select... OK Cancel Help

6.2.3 Downloading to the controller

Below, the successful installation of STEP 7 (minimum "Basic” license for SIMATIC S7-1200 or "Professional” license for SIMATIC S7-1500) is assumed.

Table 6-4

No.	Action	Remark
15.	Set the IP address via the display if using the S7-1500 CPU.	See Figure 6-1 or "Display of CPU” (\8\) When using the S7-1200, the IP address is transferred with the project.
16.	Unzip the application example from Siemens Industry Online Support (\2\) and open the project.	-
17.	In order to compile the configuration of the S7-1200 or S7-1500 CPU, right-click on the CPU and click on the "Compile > Hardware and software (only changes)” command.	 PLC_1 [CPU 1516-3 PN/DP] HMI_1 [TP Common d Document Language Online access Card Reader/U Open Open in new editor Open block/PLC data type... F7 Snapshot of the monitor values Apply snapshot values as start values Cut Ctrl+X Copy Ctrl+C Paste Ctrl+V Delete Del Rename F2 Go to topology view Go to network view Compile Download to device Online & diagnostics PLC alarms Visualization Keyboard shortcuts Hardware and software (only changes) Hardware (only changes)
18.	Load the project in the S7-1200 or S7-1500 CPU. For this purpose, select the CPU and select the "Online > Download and reset PLC program”. Now select your access point to the S7-1500 CPU and load the project into the CPU.	Type of the PG/PC interface: PN/IE PG/PC interface: Intel(R) PRO/1000 MT-Netzwerkverbindung Connection to interface/subnet: PN/IE_1 1st gateway: ices in target subnet: ☑ Show all compatible devices Device type: — · Type: PN/IE · Address: Access address · Target device: —

Note For more information on the "Loading blocks for S7-1200/1500” topic, please refer to the STEP 7 manual (see \9\).

7 Operating the Application Example

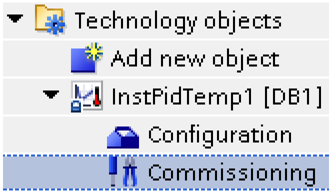
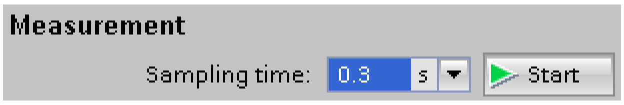
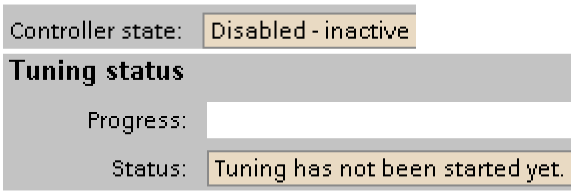
The application example is operated via the "WatchTable".

Note You can also simulate the controller part of the S7-1500 of the application example with PLCSIM. To do this, follow the notes in the function manual (\3\), chapter "Simulating PID_Temp with PLCSIM".

7.1 Preparation

Selecting the technology objects "PID_Temp" as individual instance enables you can to use the commissioning wizard and to watch the courses of the curves.

Table 7-1

No.	Action	Remark
19.	Open the commissioning window of the respective "PID_Temp" instance in "Technology objects".	 Technology objects Add new object InstPidTemp1 [DB1] Configuration Commissioning
20.	Start each measurement with the "Start" button.	 Measurement Sampling time: 0.3 s Start
21.	The measurement is started. The controller is in the "Disabled – inactive" state and the tuning status shows: "Tuning has not been started yet".	 Controller state: Disabled - inactive Tuning status Progress: Status: Tuning has not been started yet.

Note You get to the state before the first commissioning with the start values of all PID parameters, by selecting the controller and via the "Online > Download and reset PLC program (\3\).

7.2 Pretuning

Carry out the following steps for the synchronous pretuning (first heating, then cooling for heating/cooling controllers) of multi-zone controllers.

Table 7-2

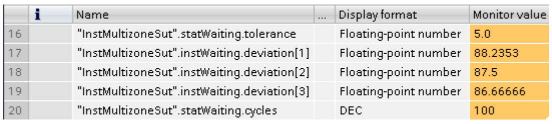
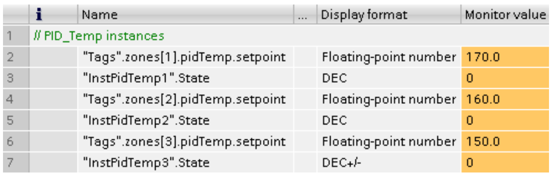
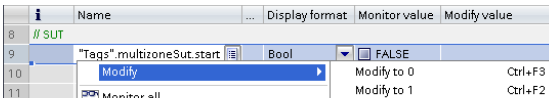
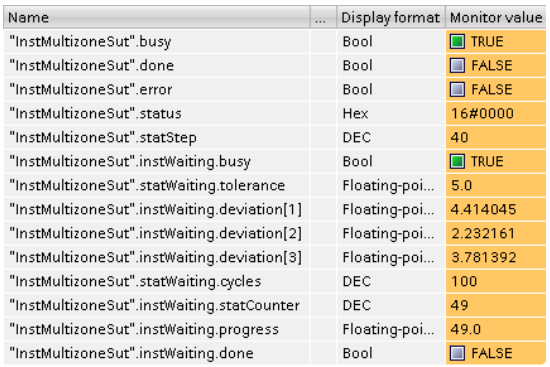
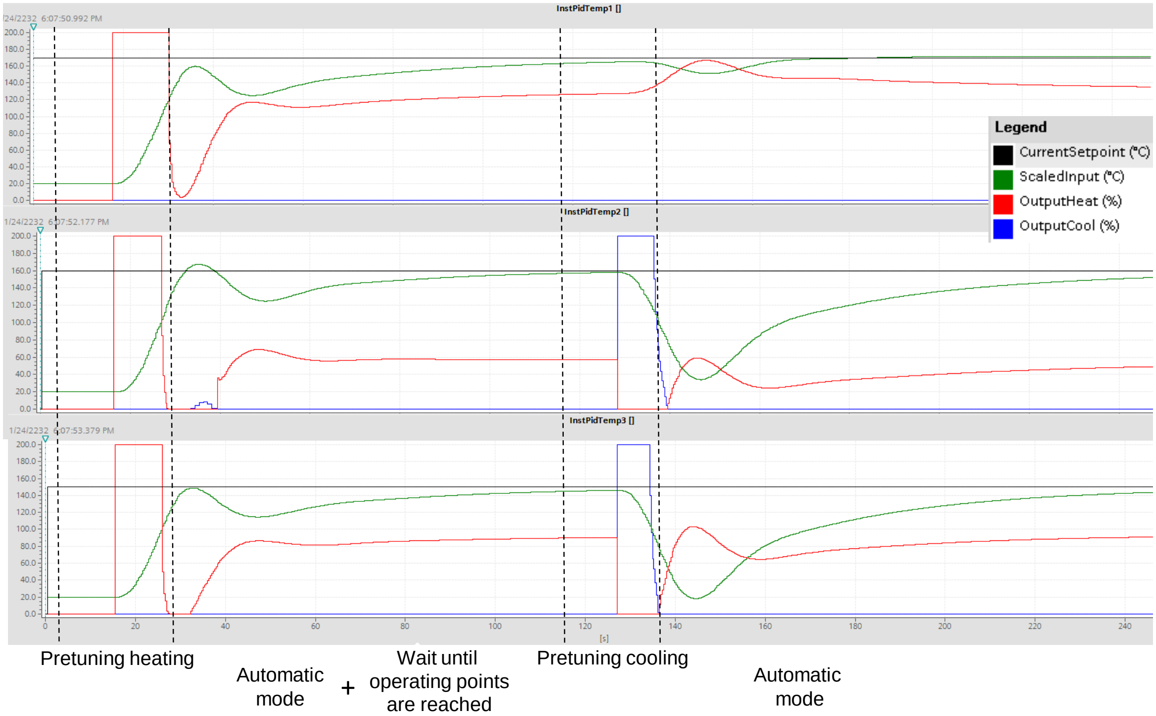
<table>
<tr><th>No.</th><th>Action</th><th>Remark</th></tr>
<tr><td>22.</td><td>Open the "WatchTable” in "Watch and force tables”.</td><td>-</td></tr>
<tr><td>23.</td><td>Click on the "Watch all” button.</td><td></td></tr>
<tr><td>24.</td><td>Between the pretuning heating and the fine tuning cooling, it is waited with FB "Waiting” until all zones are located in the operating point.
Check the proportional tolerance and the number of waited cycles and adjust them if required.</td><td>
<table>
<tr><th></th><th>i</th><th>Name</th><th>...</th><th>Display format</th><th>Monitor value</th></tr>
<tr><td>16</td><td></td><td>"InstMultizoneSut".statWaiting.tolerance</td><td></td><td>Floating-point number</td><td>5.0</td></tr>
<tr><td>17</td><td></td><td>"InstMultizoneSut".instWaiting.deviation[1]</td><td></td><td>Floating-point number</td><td>88.2353</td></tr>
<tr><td>18</td><td></td><td>"InstMultizoneSut".instWaiting.deviation[2]</td><td></td><td>Floating-point number</td><td>87.5</td></tr>
<tr><td>19</td><td></td><td>"InstMultizoneSut".instWaiting.deviation[3]</td><td></td><td>Floating-point number</td><td>86.66666</td></tr>
<tr><td>20</td><td></td><td>"InstMultizoneSut".statWaiting.cycles</td><td></td><td>DEC</td><td>100</td></tr>
</table></td></tr>
<tr><td>25.</td><td>Check the setpoints of the individual zone controllers and make sure that all controllers are in the "Inactive” (state = 0) mode.</td><td>
<table>
<tr><th></th><th>i</th><th>Name</th><th>...</th><th>Display format</th><th>Monitor value</th></tr>
<tr><td>1</td><td colspan="5">// PID_Temp instances</td></tr>
<tr><td>2</td><td></td><td>"Tags".zones[1].pidTemp.setpoint</td><td></td><td>Floating-point number</td><td>170.0</td></tr>
<tr><td>3</td><td></td><td>"InstPidTemp1".State</td><td></td><td>DEC</td><td>0</td></tr>
<tr><td>4</td><td></td><td>"Tags".zones[2].pidTemp.setpoint</td><td></td><td>Floating-point number</td><td>160.0</td></tr>
<tr><td>5</td><td></td><td>"InstPidTemp2".State</td><td></td><td>DEC</td><td>0</td></tr>
<tr><td>6</td><td></td><td>"Tags".zones[3].pidTemp.setpoint</td><td></td><td>Floating-point number</td><td>150.0</td></tr>
<tr><td>7</td><td></td><td>"InstPidTemp3".State</td><td></td><td>DEC+/-</td><td>0</td></tr>
</table></td></tr>
<tr><td>26.</td><td>Start the synchronized pretuning for all zones by setting the "<i>Tags”.multizoneSut.start</i> bit.</td><td>
<table>
<tr><th></th><th>i</th><th>Name</th><th>...</th><th>Display format</th><th>Monitor value</th><th>Modify value</th></tr>
<tr><td>8</td><td colspan="6">// SUT</td></tr>
<tr><td>9</td><td></td><td>"Tags".multizoneSut.start</td><td></td><td>Bool</td><td>□ FALSE</td><td></td></tr>
<tr><td>10</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>11</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>27.</td><td>You can monitor the progress of the pretuning through the courses of the curves in the commissioning windows (Figure 7-1) and in the watch table.
In the figure, FB <i>"MultizoneSut</i>” is in step "<i>statStep" = 40</i> and waits until the control deviations of the zones "<i>deviation[x]”</i> are within the specified proportional "<i>tolerance</i>” and it is waited for the specified "<i>cycles</i>”. The lapse of the waiting period can be read via the proportional "<i>progress</i>”.</td><td>
<table>
<tr><th>Name</th><th>...</th><th>Display format</th><th>Monitor value</th></tr>
<tr><td>"InstMultizoneSut".busy</td><td></td><td>Bool</td><td>🟩 TRUE</td></tr>
<tr><td>"InstMultizoneSut".done</td><td></td><td>Bool</td><td>□ FALSE</td></tr>
<tr><td>"InstMultizoneSut".error</td><td></td><td>Bool</td><td>□ FALSE</td></tr>
<tr><td>"InstMultizoneSut".status</td><td></td><td>Hex</td><td>16#0000</td></tr>
<tr><td>"InstMultizoneSut".statStep</td><td></td><td>DEC</td><td>40</td></tr>
<tr><td>"InstMultizoneSut".instWaiting.busy</td><td></td><td>Bool</td><td>🟩 TRUE</td></tr>
<tr><td>"InstMultizoneSut".statWaiting.tolerance</td><td></td><td>Floating-poi...</td><td>5.0</td></tr>
<tr><td>"InstMultizoneSut".instWaiting.deviation[1]</td><td></td><td>Floating-poi...</td><td>4.414045</td></tr>
<tr><td>"InstMultizoneSut".instWaiting.deviation[2]</td><td></td><td>Floating-poi...</td><td>2.232161</td></tr>
<tr><td>"InstMultizoneSut".instWaiting.deviation[3]</td><td></td><td>Floating-poi...</td><td>3.781392</td></tr>
<tr><td>"InstMultizoneSut".statWaiting.cycles</td><td></td><td>DEC</td><td>100</td></tr>
<tr><td>"InstMultizoneSut".instWaiting.statCounter</td><td></td><td>DEC</td><td>49</td></tr>
<tr><td>"InstMultizoneSut".instWaiting.progress</td><td></td><td>Floating-poi...</td><td>49.0</td></tr>
<tr><td>"InstMultizoneSut".instWaiting.done</td><td></td><td>Bool</td><td>□ FALSE</td></tr>
</table></td></tr>
</table>

Figure 7-1



The pretuning heating is started with the calculations of the standard deviation, followed by simultaneous setpoint jumps heating with the determination of the turning points of the temperature curves. Afterwards the automatic mode of the controllers waits that the respective operating point is reached, before the pretuning cooling is carried for the heating/cooling controllers (zone 2 and 3). During this time the controller of zone 1 remains in automatic mode. With the completion of the last pretuning cooling (zone 2) the FB "MultizoneSut" is successfully processed.

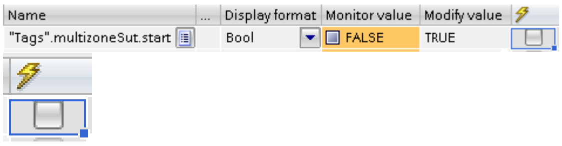
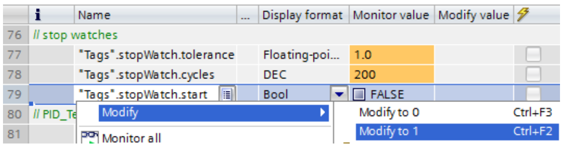
7.2.1 Disturbance variable compensation

A statement regarding quality of the parameters found can be made through the compensation of disturbance variables.

This can take place in real operation, for example, through the switching on of a hot air blower on the extruder housing.

To do this in the simulated application example, proceed as follows:

Table 7-3

No.	Action	Remark
28.	Remove all ticks in the option boxes for the selection of the tags to be controlled in the "WatchTable".	 Name | ... | Display format | Monitor value | Modify value "Tags".multizoneSut.start | Bool | FALSE | TRUE | □
29.	Check the proportional tolerance and the number of waited cycles for the measurement of the recovery times and adjust them if required. Start the measurement of the recovery times via *"Tags".stopWatch.start* tag.	 i | Name | ... | Display format | Monitor value | Modify value 76 // stop watches 77 "Tags".stopWatch.tolerance | Floating-poi... | 1.0 | □ 78 "Tags".stopWatch.cycles | DEC | 200 | □ 79 "Tags".stopWatch.start | Bool | FALSE | □ 80 // PID_Te 81 Modify ▸ | Modify to 0 Ctrl+F3 | Modify to 1 Ctrl+F2 Monitor all

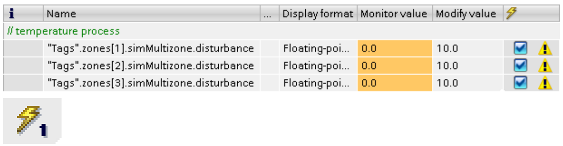
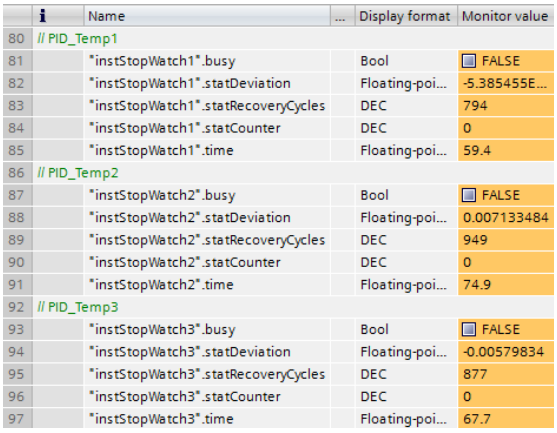
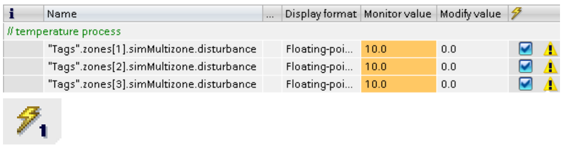
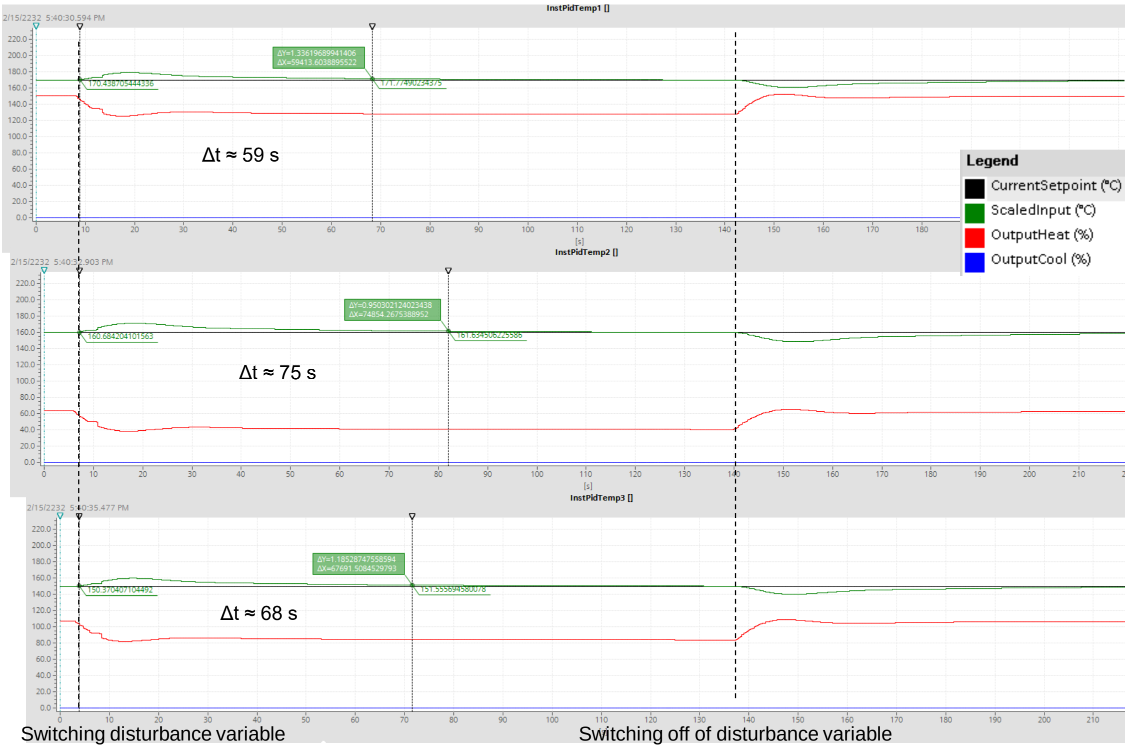
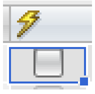
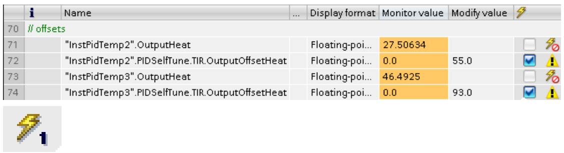
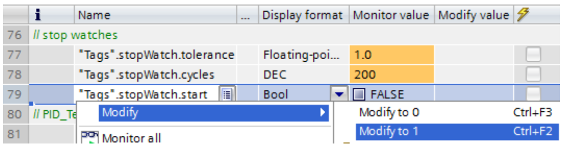
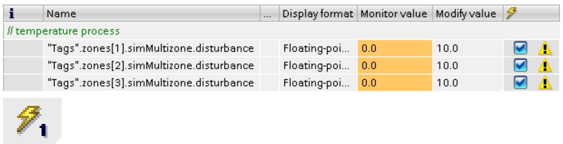
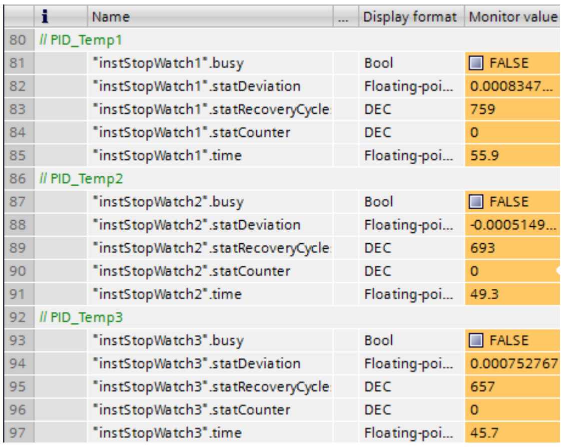
<table>
<tr><th>No.</th><th>Action</th><th>Remark</th></tr>
<tr><td>30.</td><td>For each zone, enter the following as control word for the disturbance variables:
<em>"Tags"zones[x].simMultizone.disturbance</em> = <em>10.0</em>
Click on the "Modify now" button.</td><td>
<table>
<tr><th>i</th><th>Name</th><th>...</th><th>Display format</th><th>Monitor value</th><th>Modify value</th><th>⚡</th></tr>
<tr><td colspan="7">// temperature process</td></tr>
<tr><td></td><td>"Tags".zones[1].simMultizone.disturbance</td><td></td><td>Floating-poi...</td><td>0.0</td><td>10.0</td><td>☑ ⚠</td></tr>
<tr><td></td><td>"Tags".zones[2].simMultizone.disturbance</td><td></td><td>Floating-poi...</td><td>0.0</td><td>10.0</td><td>☑ ⚠</td></tr>
<tr><td></td><td>"Tags".zones[3].simMultizone.disturbance</td><td></td><td>Floating-poi...</td><td>0.0</td><td>10.0</td><td>☑ ⚠</td></tr>
</table>
⚡1</td></tr>
<tr><td>31.</td><td>Monitor the course of the curve and the watch table.
After compensation of the disturbance variables, the required times are displayed via the "<em>instStopWatchX".time</em>.
You can also stop the course of the curve via ■ Stop and then determine the recovery time via the vertical measurement cursor (Figure 7-2).</td><td>
<table>
<tr><th></th><th>i</th><th>Name</th><th>...</th><th>Display format</th><th>Monitor value</th></tr>
<tr><td>80</td><td colspan="5">// PID_Temp1</td></tr>
<tr><td>81</td><td></td><td>"instStopWatch1".busy</td><td></td><td>Bool</td><td>□ FALSE</td></tr>
<tr><td>82</td><td></td><td>"instStopWatch1".statDeviation</td><td></td><td>Floating-poi...</td><td>-5.385455E...</td></tr>
<tr><td>83</td><td></td><td>"instStopWatch1".statRecoveryCycles</td><td></td><td>DEC</td><td>794</td></tr>
<tr><td>84</td><td></td><td>"instStopWatch1".statCounter</td><td></td><td>DEC</td><td>0</td></tr>
<tr><td>85</td><td></td><td>"instStopWatch1".time</td><td></td><td>Floating-poi...</td><td>59.4</td></tr>
<tr><td>86</td><td colspan="5">// PID_Temp2</td></tr>
<tr><td>87</td><td></td><td>"instStopWatch2".busy</td><td></td><td>Bool</td><td>□ FALSE</td></tr>
<tr><td>88</td><td></td><td>"instStopWatch2".statDeviation</td><td></td><td>Floating-poi...</td><td>0.007133484</td></tr>
<tr><td>89</td><td></td><td>"instStopWatch2".statRecoveryCycles</td><td></td><td>DEC</td><td>949</td></tr>
<tr><td>90</td><td></td><td>"instStopWatch2".statCounter</td><td></td><td>DEC</td><td>0</td></tr>
<tr><td>91</td><td></td><td>"instStopWatch2".time</td><td></td><td>Floating-poi...</td><td>74.9</td></tr>
<tr><td>92</td><td colspan="5">// PID_Temp3</td></tr>
<tr><td>93</td><td></td><td>"instStopWatch3".busy</td><td></td><td>Bool</td><td>□ FALSE</td></tr>
<tr><td>94</td><td></td><td>"instStopWatch3".statDeviation</td><td></td><td>Floating-poi...</td><td>-0.00579834</td></tr>
<tr><td>95</td><td></td><td>"instStopWatch3".statRecoveryCycles</td><td></td><td>DEC</td><td>877</td></tr>
<tr><td>96</td><td></td><td>"instStopWatch3".statCounter</td><td></td><td>DEC</td><td>0</td></tr>
<tr><td>97</td><td></td><td>"instStopWatch3".time</td><td></td><td>Floating-poi...</td><td>67.7</td></tr>
</table></td></tr>
<tr><td>32.</td><td>Then remove the disturbance variables again from the control loop:
<em>"Tags"zones[x].simMultizone.disturbance</em> = <em>0.0</em>.
Click on the "Modify now" button.</td><td>
<table>
<tr><th>i</th><th>Name</th><th>...</th><th>Display format</th><th>Monitor value</th><th>Modify value</th><th>⚡</th></tr>
<tr><td colspan="7">// temperature process</td></tr>
<tr><td></td><td>"Tags".zones[1].simMultizone.disturbance</td><td></td><td>Floating-poi...</td><td>10.0</td><td>0.0</td><td>☑ ⚠</td></tr>
<tr><td></td><td>"Tags".zones[2].simMultizone.disturbance</td><td></td><td>Floating-poi...</td><td>10.0</td><td>0.0</td><td>☑ ⚠</td></tr>
<tr><td></td><td>"Tags".zones[3].simMultizone.disturbance</td><td></td><td>Floating-poi...</td><td>10.0</td><td>0.0</td><td>☑ ⚠</td></tr>
</table>
⚡1</td></tr>
</table>

Figure 7-2



For the disturbance variable compensation in the opposite energy type, the controller has to change into this energy type first. Proceed as follows:

Table 7-4

No.	Action	Remark
33.	Remove all ticks in the option boxes for the selection of the tags to be controlled in the "WatchTable".	
34.	Select a higher tuning offset heating than the stationary setpoint heating ("OutputHeat") for each zone of the heating/cooling controller in automatic mode (for example, 2 x "OutputHeat"), in order to force the cooling mode.	 i Name ... Display format Monitor value Modify value 70 // offsets 71 "InstPidTemp2".OutputHeat Floating-poi... 27.50634 ☐ 72 "InstPidTemp2".PIDSelfTune.TIR.OutputOffsetHeat Floating-poi... 0.0 55.0 ☑ 73 "InstPidTemp3".OutputHeat Floating-poi... 46.4925 ☐ 74 "InstPidTemp3".PIDSelfTune.TIR.OutputOffsetHeat Floating-poi... 0.0 93.0 ☑
35.	Watch the course of the curve. Now wait until all zones have reached their stationary operating point.	
36.	Start the measurement of the recovery times via *"Tags".stopWatch.start* tag.	 i Name ... Display format Monitor value Modify value 76 // stop watches 77 "Tags".stopWatch.tolerance Floating-poi... 1.0 ☐ 78 "Tags".stopWatch.cycles DEC 200 ☐ 79 "Tags".stopWatch.start Bool ☐ FALSE ☐ 80 // PID_Te 81 Modify ▸ Monitor all Modify to 0 Ctrl+F3 Modify to 1 Ctrl+F2
37.	As control word for the disturbance value, enter *"Tags"zones[x].simMultizone.disturbance = 10.0 for each zone.* Click on the "Modify now" button.	 i Name ... Display format Monitor value Modify value // temperature process "Tags".zones[1].simMultizone.disturbance Floating-poi... 0.0 10.0 ☑ "Tags".zones[2].simMultizone.disturbance Floating-poi... 0.0 10.0 ☑ "Tags".zones[3].simMultizone.disturbance Floating-poi... 0.0 10.0 ☑
38.	Monitor the course of the curve and the watch table. After compensation of the disturbance variables, the required times are displayed via the "*instStopWatchX".time*. You can also stop the course of the curve with ■ Stop and determine the recovery time via the vertical measurement cursor (Figure 7-3).	 i Name ... Display format Monitor value 80 // PID_Temp1 81 "instStopWatch1".busy Bool ☐ FALSE 82 "instStopWatch1".statDeviation Floating-poi... 0.0008347... 83 "instStopWatch1".statRecoveryCycle DEC 759 84 "instStopWatch1".statCounter DEC 0 85 "instStopWatch1".time Floating-poi... 55.9 86 // PID_Temp2 87 "instStopWatch2".busy Bool ☐ FALSE 88 "instStopWatch2".statDeviation Floating-poi... -0.0005149... 89 "instStopWatch2".statRecoveryCycle DEC 693 90 "instStopWatch2".statCounter DEC 0 91 "instStopWatch2".time Floating-poi... 49.3 92 // PID_Temp3 93 "instStopWatch3".busy Bool ☐ FALSE 94 "instStopWatch3".statDeviation Floating-poi... 0.000752767 95 "instStopWatch3".statRecoveryCycle DEC 657 96 "instStopWatch3".statCounter DEC 0 97 "instStopWatch3".time Floating-poi... 45.7

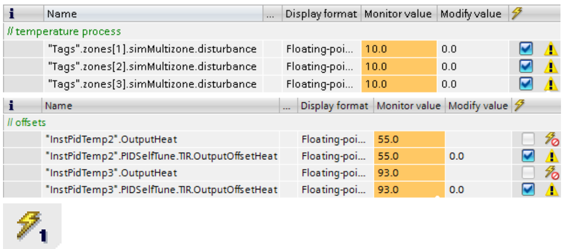
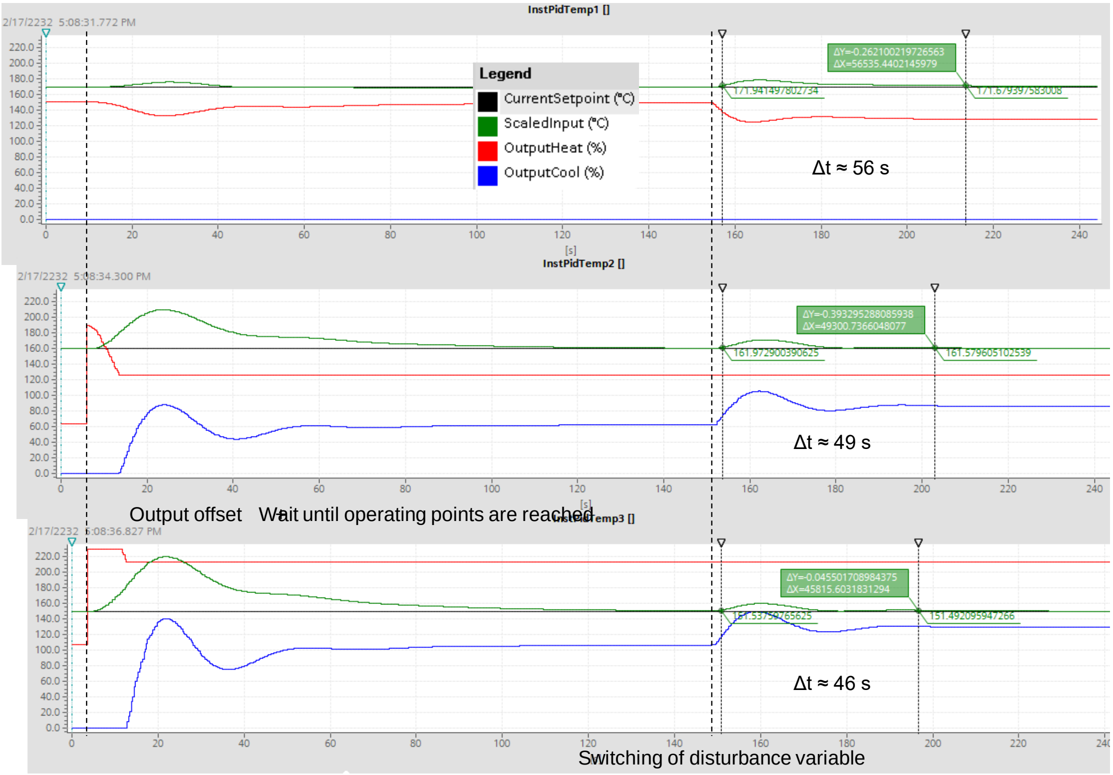
39.	Then remove the disturbance variables again from the control loop via *"Tags"zones[x].simMultizone.disturbance = 0.0* and the offsets *"InstPidTempX".PIDSelfTune.TIR.OutputOffsetHeat = 0.0*.
	Click on the "Modify now" button.
		
		i Name ... Display format Monitor value Modify value
		// temperature process
		"Tags".zones[1].simMultizone.disturbance Floating-poi... 10.0 0.0 ☑
		"Tags".zones[2].simMultizone.disturbance Floating-poi... 10.0 0.0 ☑
		"Tags".zones[3].simMultizone.disturbance Floating-poi... 10.0 0.0 ☑
		i Name ... Display format Monitor value Modify value
		// offsets
		"InstPidTemp2".OutputHeat Floating-poi... 55.0 ☐
		"InstPidTemp2".PIDSelfTune.TIR.OutputOffsetHeat Floating-poi... 55.0 0.0 ☑
		"InstPidTemp3".OutputHeat Floating-poi... 93.0 ☐
		"InstPidTemp3".PIDSelfTune.TIR.OutputOffsetHeat Floating-poi... 93.0 0.0 ☑

Figure 7-3

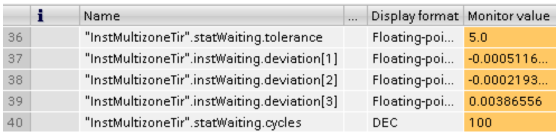
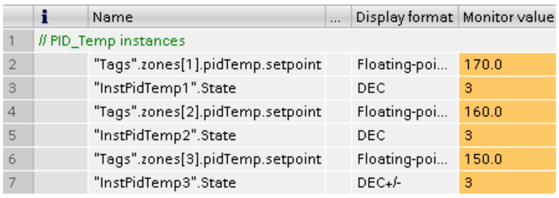


Since the controller for zone 1 is a pure heating controller, the previously determined disturbance variable recovery time is confirmed here.

7.3 Fine tuning

Perform the following steps to synchronize the fine tuning of multi-zone controllers.

Table 7-5

<table>
<tr><th>No.</th><th>Action</th><th>Remark</th></tr>
<tr><td>40.</td><td>Before fine tuning is started, it is waited with FB "Waiting" until all zones are located in the operating point.
This procedure is important if one zone is to be fine-tuned in the opposite energy type.
In the present example, first, the fine tuning is done in the energy type of the operating point. This is why these settings must not to be observed.</td><td>
<table>
<tr><th></th><th>i</th><th>Name</th><th>...</th><th>Display format</th><th>Monitor value</th></tr>
<tr><td>36</td><td></td><td>"InstMultizoneTir".statWaiting.tolerance</td><td></td><td>Floating-poi...</td><td>5.0</td></tr>
<tr><td>37</td><td></td><td>"InstMultizoneTir".instWaiting.deviation[1]</td><td></td><td>Floating-poi...</td><td>-0.0005116...</td></tr>
<tr><td>38</td><td></td><td>"InstMultizoneTir".instWaiting.deviation[2]</td><td></td><td>Floating-poi...</td><td>-0.0002193...</td></tr>
<tr><td>39</td><td></td><td>"InstMultizoneTir".instWaiting.deviation[3]</td><td></td><td>Floating-poi...</td><td>0.00386556</td></tr>
<tr><td>40</td><td></td><td>"InstMultizoneTir".statWaiting.cycles</td><td></td><td>DEC</td><td>100</td></tr>
</table></td></tr>
<tr><td>41.</td><td>Check the setpoints "setpoint" of the individual zone controllers and make sure that all controllers are in "Automatic" (State = 3) mode.</td><td>
<table>
<tr><th></th><th>i</th><th>Name</th><th>...</th><th>Display format</th><th>Monitor value</th></tr>
<tr><td>1</td><td colspan="5">// PID_Temp instances</td></tr>
<tr><td>2</td><td></td><td>"Tags".zones[1].pidTemp.setpoint</td><td></td><td>Floating-poi...</td><td>170.0</td></tr>
<tr><td>3</td><td></td><td>"InstPidTemp1".State</td><td></td><td>DEC</td><td>3</td></tr>
<tr><td>4</td><td></td><td>"Tags".zones[2].pidTemp.setpoint</td><td></td><td>Floating-poi...</td><td>160.0</td></tr>
<tr><td>5</td><td></td><td>"InstPidTemp2".State</td><td></td><td>DEC</td><td>3</td></tr>
<tr><td>6</td><td></td><td>"Tags".zones[3].pidTemp.setpoint</td><td></td><td>Floating-poi...</td><td>150.0</td></tr>
<tr><td>7</td><td></td><td>"InstPidTemp3".State</td><td></td><td>DEC+/-</td><td>3</td></tr>
</table></td></tr>
</table>

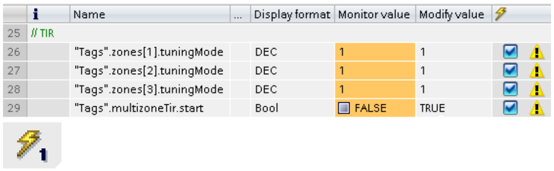
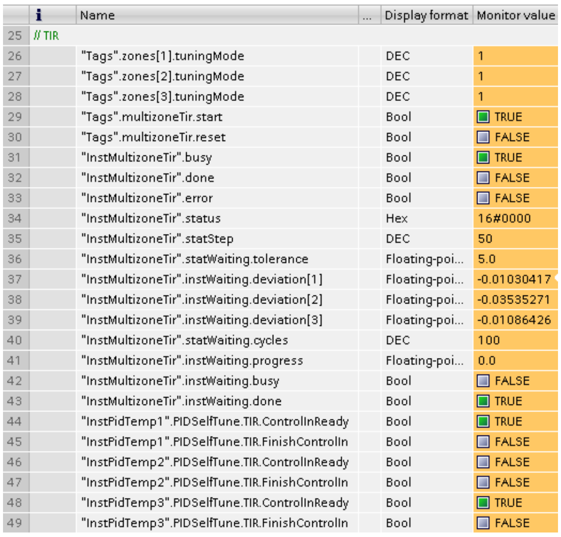
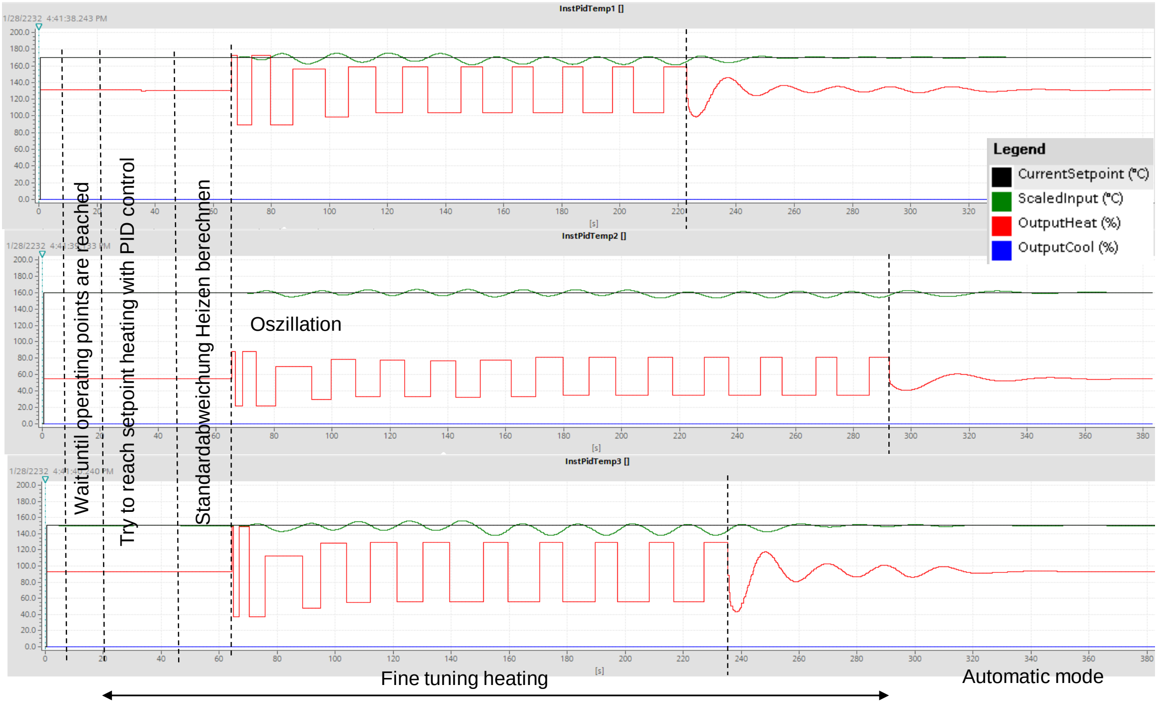
<table>
<tr><th>No.</th><th>Action</th><th>Remark</th></tr>
<tr>
<td>42.</td>
<td>You should do the fine tuning first of all in the energy type of the operating point.
After the pretuning, all controllers in the current example are in heating mode.
As a result, select the energy type of each zone "tuningMode = 1” for heating.
Start the synchronized fine tuning with these specifications by setting the bit ”<i>Tags".multizoneTir.start</i>.</td>
<td>

<table>
<tr><th></th><th>i</th><th>Name</th><th>...</th><th>Display format</th><th>Monitor value</th><th>Modify value</th><th>⚡</th><th></th></tr>
<tr><td>25</td><td colspan="8">// TIR</td></tr>
<tr><td>26</td><td></td><td>"Tags".zones[1].tuningMode</td><td></td><td>DEC</td><td>1</td><td>1</td><td>☑</td><td>⚠</td></tr>
<tr><td>27</td><td></td><td>"Tags".zones[2].tuningMode</td><td></td><td>DEC</td><td>1</td><td>1</td><td>☑</td><td>⚠</td></tr>
<tr><td>28</td><td></td><td>"Tags".zones[3].tuningMode</td><td></td><td>DEC</td><td>1</td><td>1</td><td>☑</td><td>⚠</td></tr>
<tr><td>29</td><td></td><td>"Tags".multizoneTir.start</td><td></td><td>Bool</td><td>☐ FALSE</td><td>TRUE</td><td>☑</td><td>⚠</td></tr>
</table>
⚡1
</td>
</tr>
<tr>
<td>43.</td>
<td>You can monitor the progress of the fine tuning through the courses of the curves in the commissioning windows (Figure 7-1) and in the watch table.
In the figure, FB <i>"MultizoneTir"</i> is in step "<i>statStep" = 50</i> and waits during the fine tuning that all zone controllers report that the setpoints have been reached via the <i>"ControlInReady"</i> bits. Then the <i>"FinishControlIn</i>” bits are set and the fine tuning for all zones is continued with the calculation of the standard deviation.</td>
<td>

<table>
<tr><th></th><th>i</th><th>Name</th><th>...</th><th>Display format</th><th>Monitor value</th></tr>
<tr><td>25</td><td colspan="5">// TIR</td></tr>
<tr><td>26</td><td></td><td>"Tags".zones[1].tuningMode</td><td></td><td>DEC</td><td>1</td></tr>
<tr><td>27</td><td></td><td>"Tags".zones[2].tuningMode</td><td></td><td>DEC</td><td>1</td></tr>
<tr><td>28</td><td></td><td>"Tags".zones[3].tuningMode</td><td></td><td>DEC</td><td>1</td></tr>
<tr><td>29</td><td></td><td>"Tags".multizoneTir.start</td><td></td><td>Bool</td><td>🟩 TRUE</td></tr>
<tr><td>30</td><td></td><td>"Tags".multizoneTir.reset</td><td></td><td>Bool</td><td>☐ FALSE</td></tr>
<tr><td>31</td><td></td><td>"InstMultizoneTir".busy</td><td></td><td>Bool</td><td>🟩 TRUE</td></tr>
<tr><td>32</td><td></td><td>"InstMultizoneTir".done</td><td></td><td>Bool</td><td>☐ FALSE</td></tr>
<tr><td>33</td><td></td><td>"InstMultizoneTir".error</td><td></td><td>Bool</td><td>☐ FALSE</td></tr>
<tr><td>34</td><td></td><td>"InstMultizoneTir".status</td><td></td><td>Hex</td><td>16#0000</td></tr>
<tr><td>35</td><td></td><td>"InstMultizoneTir".statStep</td><td></td><td>DEC</td><td>50</td></tr>
<tr><td>36</td><td></td><td>"InstMultizoneTir".statWaiting.tolerance</td><td></td><td>Floating-poi...</td><td>5.0</td></tr>
<tr><td>37</td><td></td><td>"InstMultizoneTir".instWaiting.deviation[1]</td><td></td><td>Floating-poi...</td><td>-0.01030417</td></tr>
<tr><td>38</td><td></td><td>"InstMultizoneTir".instWaiting.deviation[2]</td><td></td><td>Floating-poi...</td><td>-0.03535271</td></tr>
<tr><td>39</td><td></td><td>"InstMultizoneTir".instWaiting.deviation[3]</td><td></td><td>Floating-poi...</td><td>-0.01086426</td></tr>
<tr><td>40</td><td></td><td>"InstMultizoneTir".statWaiting.cycles</td><td></td><td>DEC</td><td>100</td></tr>
<tr><td>41</td><td></td><td>"InstMultizoneTir".instWaiting.progress</td><td></td><td>Floating-poi...</td><td>0.0</td></tr>
<tr><td>42</td><td></td><td>"InstMultizoneTir".instWaiting.busy</td><td></td><td>Bool</td><td>☐ FALSE</td></tr>
<tr><td>43</td><td></td><td>"InstMultizoneTir".instWaiting.done</td><td></td><td>Bool</td><td>🟩 TRUE</td></tr>
<tr><td>44</td><td></td><td>"InstPidTemp1".PIDSelfTune.TIR.ControlInReady</td><td></td><td>Bool</td><td>🟩 TRUE</td></tr>
<tr><td>45</td><td></td><td>"InstPidTemp1".PIDSelfTune.TIR.FinishControlIn</td><td></td><td>Bool</td><td>☐ FALSE</td></tr>
<tr><td>46</td><td></td><td>"InstPidTemp2".PIDSelfTune.TIR.ControlInReady</td><td></td><td>Bool</td><td>☐ FALSE</td></tr>
<tr><td>47</td><td></td><td>"InstPidTemp2".PIDSelfTune.TIR.FinishControlIn</td><td></td><td>Bool</td><td>☐ FALSE</td></tr>
<tr><td>48</td><td></td><td>"InstPidTemp3".PIDSelfTune.TIR.ControlInReady</td><td></td><td>Bool</td><td>🟩 TRUE</td></tr>
<tr><td>49</td><td></td><td>"InstPidTemp3".PIDSelfTune.TIR.FinishControlIn</td><td></td><td>Bool</td><td>☐ FALSE</td></tr>
</table>
</td>
</tr>
</table>

Figure 7-4



The FB "MultizoneTir" waits until all operating points have been reached (only required if the opposite energy type is to be fine-tuned).

Afterwards it is tried to reach the setpoints with the existing PID parameters as part of the fine tuning, in order to calculate the standard deviation synchronously and to initiate the oscillation of the actual values via variation of the manipulated variables.

The PID parameters for each zone are recalculated from amplitude and frequency of the actual value oscillations.

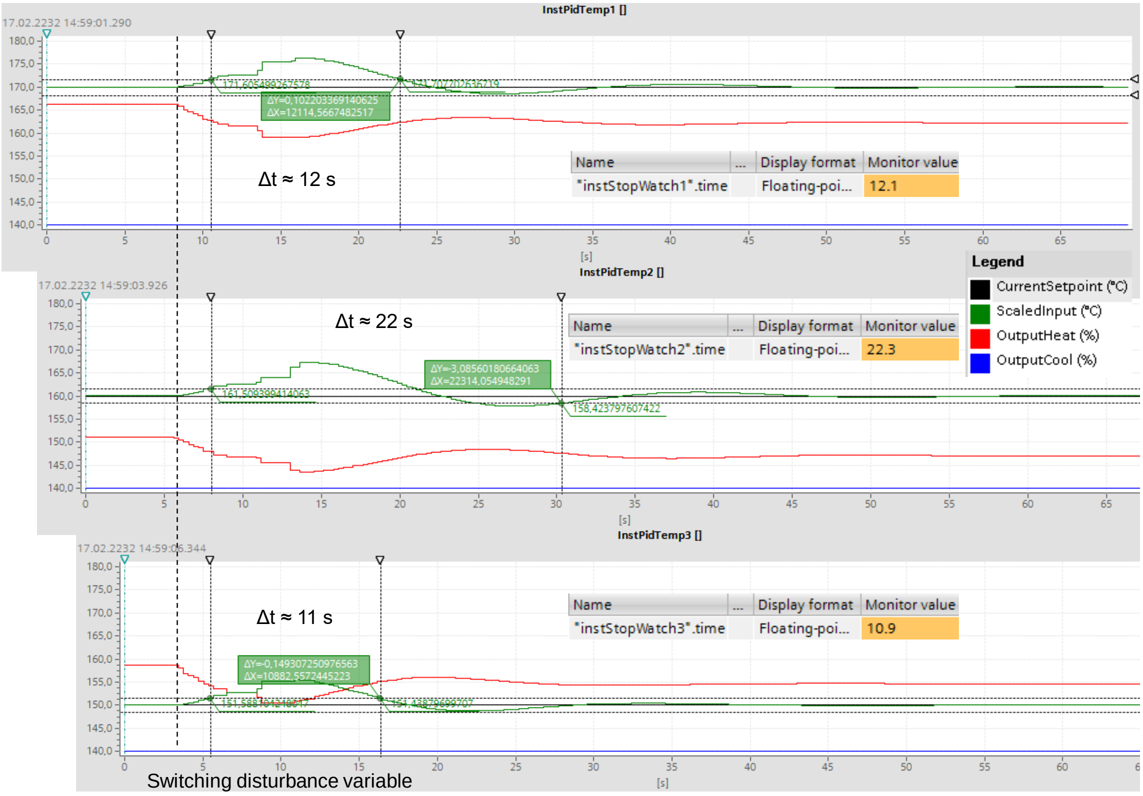
With the completion of the last fine tuning (zone 2) the FB "MultizoneTir" is successfully processed.

Disturbance variable compensation

Repeat the procedure from Table 7-3, to be able to evaluate the control quality of the parameters found.

In the process, keep the tolerance specification and the number of waited cycles to be able to compare the detected times for the disturbance variable compensation with each other.

Figure 7-5



Name	...	Display format	Monitor value
"instStopWatch1".time		Floating-poi...	12.1

Name	...	Display format	Monitor value
"instStopWatch2".time		Floating-poi...	22.3

Name	...	Display format	Monitor value
"instStopWatch3".time		Floating-poi...	10.9

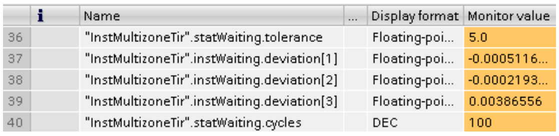
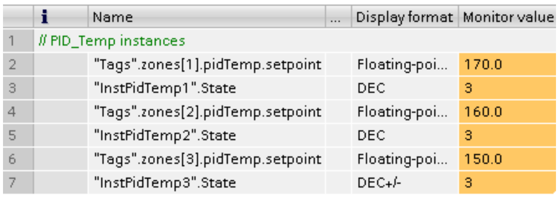
Legend

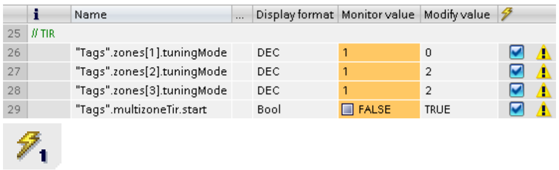
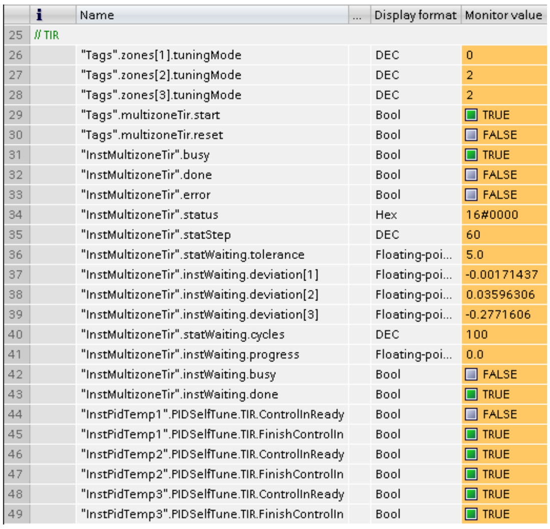
■	CurrentSetpoint (°C)
🟩	ScaledInput (°C)
🟥	OutputHeat (%)
🟦	OutputCool (%)

7.3.1 Tuning of the opposite energy type

With the FB "MultizoneTir" you can also fine tune the other energy type (other than the present one) for heating/cooling controllers. In the present example this is the energy type cooling for zone 2 and 3.

Table 7-6

No.	Action	Remark
44.	Before fine tuning is started, it is waited with FB "Waiting" until all zones are located in the operating point. This procedure guarantees that the calculation of the tuning offset happens in the steady-state control loop. If required, adjust the percentage of the "tolerance" and/or the number of waited "cycles".	 i | Name | ... | Display format | Monitor value 36 | "InstMultizoneTir".statWaiting.tolerance | Floating-poi... | 5.0 37 | "InstMultizoneTir".instWaiting.deviation[1] | Floating-poi... | -0.0005116... 38 | "InstMultizoneTir".instWaiting.deviation[2] | Floating-poi... | -0.0002193... 39 | "InstMultizoneTir".instWaiting.deviation[3] | Floating-poi... | 0.00386556 40 | "InstMultizoneTir".statWaiting.cycles | DEC | 100
45.	Check the setpoints "setpoint" of the individual zone controllers and make sure that all controllers are in "Automatic" (State = 3) mode.	 i | Name | ... | Display format | Monitor value 1 | // PID_Temp instances 2 | "Tags".zones[1].pidTemp.setpoint | Floating-poi... | 170.0 3 | "InstPidTemp1".State | DEC | 3 4 | "Tags".zones[2].pidTemp.setpoint | Floating-poi... | 160.0 5 | "InstPidTemp2".State | DEC | 3 6 | "Tags".zones[3].pidTemp.setpoint | Floating-poi... | 150.0 7 | "InstPidTemp3".State | DEC+/- | 3

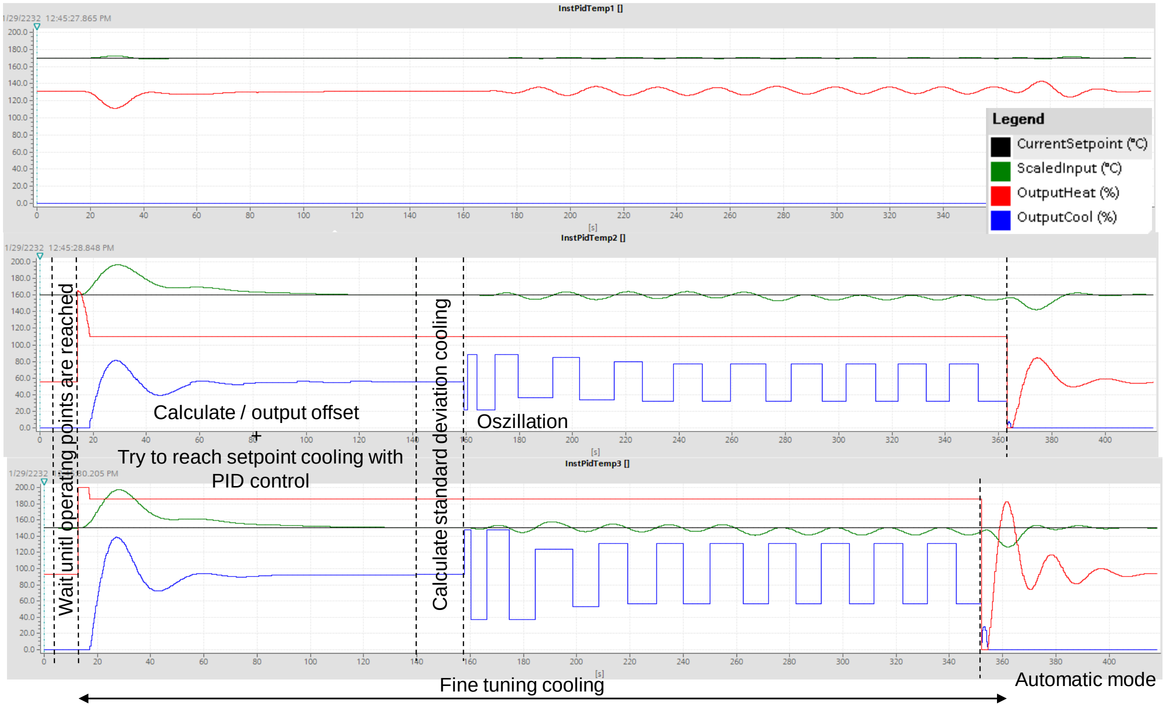
No.	Action	Remark
46.	Select "tuningMode = 2" for the energy type cooling in zones 2 and 3. Since the heating controller of zone 1 has already run through the fine tuning, select "tuningMode = 0" here for no tuning. Start the synchronized fine tuning with these specifications by setting the bit *"Tags".multizoneTir.start* .	
47.	You can monitor the progress of the fine tuning through the courses of the curves in the commissioning windows (Figure 7-1) and in the watch table. In the figure, FB *"MultizoneTir"* is in step *"statStep" = 60* and calculates the standard deviation during the fine tuning after the two zone controllers 2 and 3 have reported that the setpoints have been reached (*"ControlInReady")* and *"FB "MultizoneTir"* has been set.	

	i	Name	...	Display format	Monitor value	Modify value	⚡	
25	// TIR							
26		"Tags".zones[1].tuningMode		DEC	1	0	☑	⚠
27		"Tags".zones[2].tuningMode		DEC	1	2	☑	⚠
28		"Tags".zones[3].tuningMode		DEC	1	2	☑	⚠
29		"Tags".multizoneTir.start		Bool	□ FALSE	TRUE	☑	⚠

⚡1

	i	Name	...	Display format	Monitor value
25	// TIR				
26		"Tags".zones[1].tuningMode		DEC	0
27		"Tags".zones[2].tuningMode		DEC	2
28		"Tags".zones[3].tuningMode		DEC	2
29		"Tags".multizoneTir.start		Bool	🟩 TRUE
30		"Tags".multizoneTir.reset		Bool	□ FALSE
31		"InstMultizoneTir".busy		Bool	🟩 TRUE
32		"InstMultizoneTir".done		Bool	□ FALSE
33		"InstMultizoneTir".error		Bool	□ FALSE
34		"InstMultizoneTir".status		Hex	16#0000
35		"InstMultizoneTir".statStep		DEC	60
36		"InstMultizoneTir".statWaiting.tolerance		Floating-poi...	5.0
37		"InstMultizoneTir".instWaiting.deviation[1]		Floating-poi...	-0.00171437
38		"InstMultizoneTir".instWaiting.deviation[2]		Floating-poi...	0.03596306
39		"InstMultizoneTir".instWaiting.deviation[3]		Floating-poi...	-0.2771606
40		"InstMultizoneTir".statWaiting.cycles		DEC	100
41		"InstMultizoneTir".instWaiting.progress		Floating-poi...	0.0
42		"InstMultizoneTir".instWaiting.busy		Bool	□ FALSE
43		"InstMultizoneTir".instWaiting.done		Bool	🟩 TRUE
44		"InstPidTemp1".PIDSelfTune.TIR.ControlInReady		Bool	□ FALSE
45		"InstPidTemp1".PIDSelfTune.TIR.FinishControlIn		Bool	🟩 TRUE
46		"InstPidTemp2".PIDSelfTune.TIR.ControlInReady		Bool	🟩 TRUE
47		"InstPidTemp2".PIDSelfTune.TIR.FinishControlIn		Bool	🟩 TRUE
48		"InstPidTemp3".PIDSelfTune.TIR.ControlInReady		Bool	🟩 TRUE
49		"InstPidTemp3".PIDSelfTune.TIR.FinishControlIn		Bool	🟩 TRUE

Figure 7-6



The FB "MultizoneTir" waits until all operating points have been reached before the FCs "TirOffset" calculate and output the heating tuning offsets.

This increase of the heating manipulated variables forces the zone controllers 2 and 3 to energy type cooling, in order to maintain the setpoint.

Afterwards it is tried to reach the setpoints with the existing PID parameters as part of the fine tuning, in order to calculate the standard deviation synchronously and to initiate the oscillation of the actual values via variation of the manipulated variables.

The PID parameters for each zone are recalculated from amplitude and frequency of the actual value oscillations.

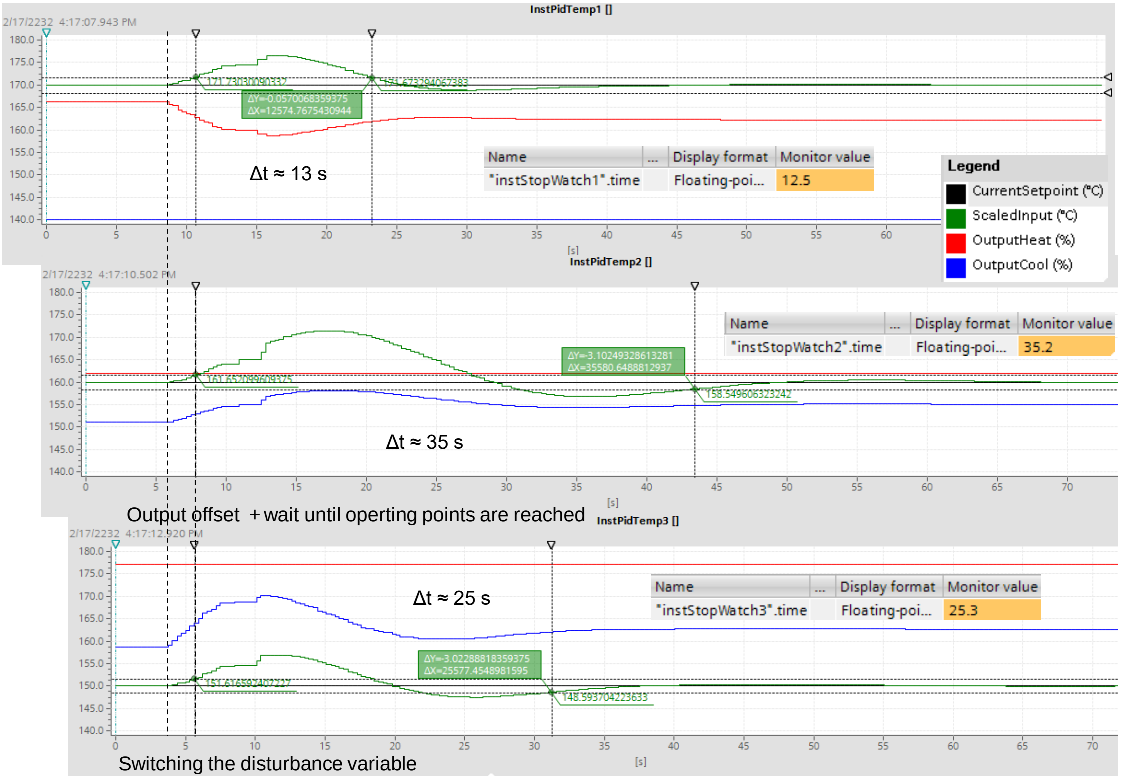
With the completion of the last fine tuning (zone 2) the FB "MultizoneTir" is successfully processed.

Disturbance variable compensation

Repeat the procedure from Table 7-4, to be able to evaluate the control quality of the parameters found.

In the process, keep the tolerance specification and the number of waited cycles to be able to compare the detected times for the disturbance variable compensation with each other.

Figure 7-7



Δt ≈ 13 s

Name	...	Display format	Monitor value
"instStopWatch1".time		Floating-poi...	12.5

Δt ≈ 35 s

Name	...	Display format	Monitor value
"instStopWatch2".time		Floating-poi...	35.2

Output offset + wait until operting points are reached

Δt ≈ 25 s

Name	...	Display format	Monitor value
"instStopWatch3".time		Floating-poi...	25.3

Switching the disturbance variable

Since the controller for zone 1 is a pure heating controller, the previously determined disturbance variable recovery time is confirmed here.

Table 7-7: Disturbance variable recovery time according to tuning types

Zones	Energy type	Pretuning	Fine tuning
1	Heating	58 s	12 s
2	Heating	75 s	22 s
	Cooling	49 s	35 s
3	Heating	68 s	11 s
	Cooling	46 s	25 s

Table 7-7 shows that the fine tuning leads to a faster compensation of manipulated variables then the pretuning.

8 Further Notes, Tips and Tricks

8.1 Adjusting simulation

During the handling processing of an extruder, at times so much heat is created through friction and shearing of the material (for example, plastic granulate) that cooling is necessary.

In order to recreate this behavior within the simulation block, a disturbance variable can be connected for the respective zone.

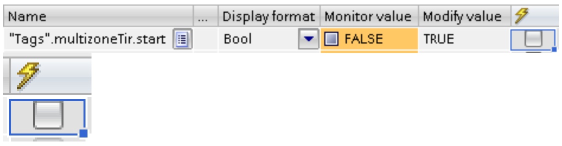
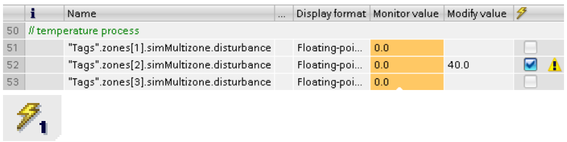
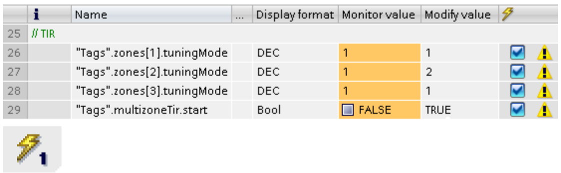
For example, zone 2 is to cool after commissioning through the pretuning in automatic mode to reach the setpoint.

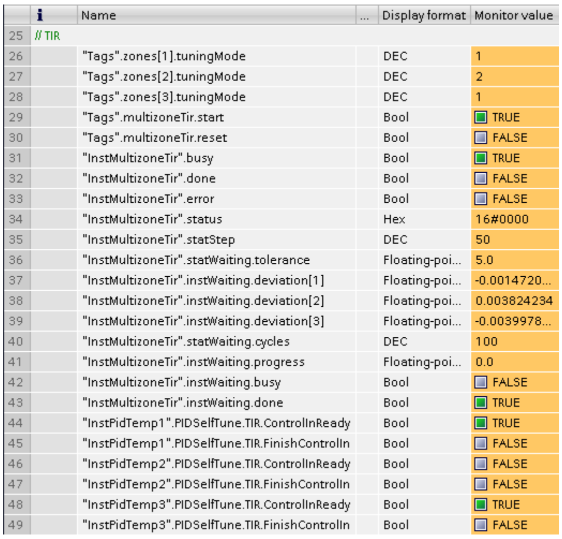
To do this, empirically determine the value for the *"Tags".zones[2].simMultizone.disturbance* disturbance value (for example, *40.0*).

First of all, the synchronous fine tuning of the zones takes place in the respective energy type of the controller:

- Zone 1: Heating (*"Tags".zones[1].tuningMode = 1*)
- Zone 2: Cooling (*"Tags".zones[1].tuningMode = 2*)
- Zone 3: Heating (*"Tags".zones[1].tuningMode = 1*)

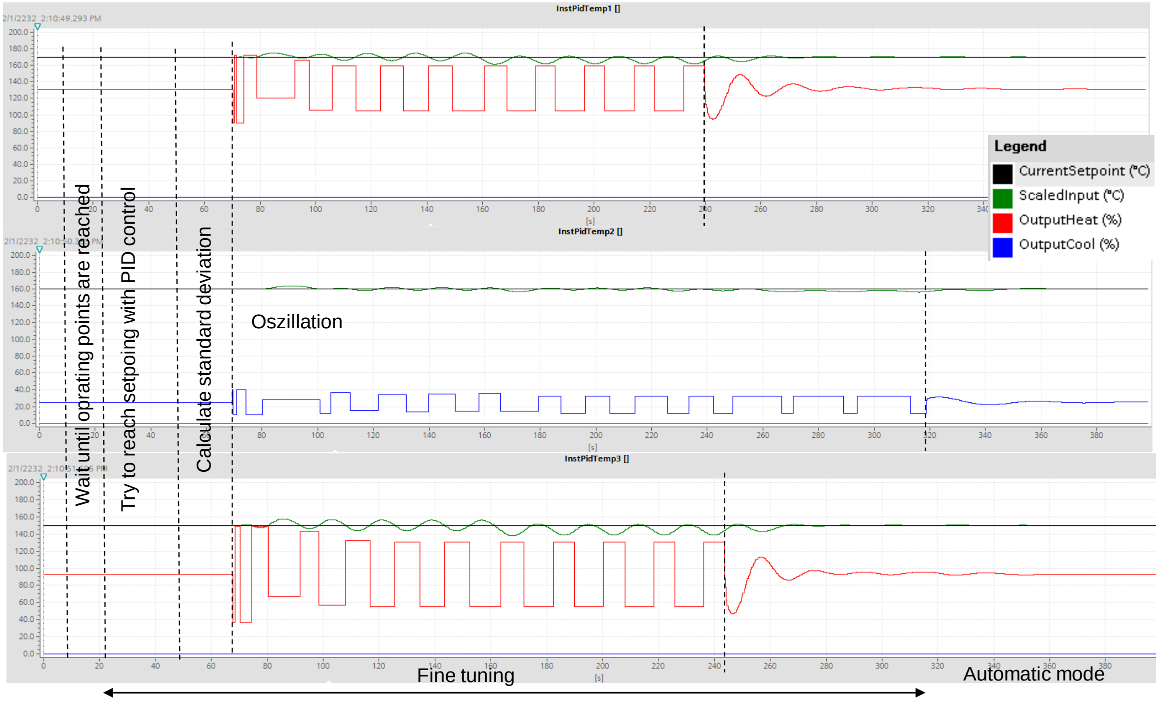
Table 8-1

No.	Action	Remark
48.	Remove all ticks in the option boxes for the selection of the tags to be controlled in the "WatchTable".	 Name | ... | Display format | Monitor value | Modify value "Tags".multizoneTir.start | Bool | FALSE | TRUE
49.	Enter "*Tags"zones[2].simMultizone.disturbance = 40.0* as control value of the disturbance value for zone 2. Click on the "Modify now" button.	 i | Name | ... | Display format | Monitor value | Modify value 50 // temperature process 51 "Tags".zones[1].simMultizone.disturbance | Floating-poi... | 0.0 | ☐ 52 "Tags".zones[2].simMultizone.disturbance | Floating-poi... | 0.0 | 40.0 | ☑ ⚠ 53 "Tags".zones[3].simMultizone.disturbance | Floating-poi... | 0.0 | ☐
50.	Follow step 1 and 2 from Table 7-5.	-
51.	Select the following energy types for the fine tuning: • Zone 1: Heating ("Tags".zones[1].tuningMode = 1) • Zone 2: Cooling ("Tags".zones[1].tuningMode = 2) • Zone 3: Heating ("Tags".zones[1].tuningMode = 1) Start the synchronized fine tuning with these specifications by setting the bit "*Tags".multizoneTir.start*.	 i | Name | ... | Display format | Monitor value | Modify value 25 // TIR 26 "Tags".zones[1].tuningMode | DEC | 1 | 1 | ☑ ⚠ 27 "Tags".zones[2].tuningMode | DEC | 1 | 2 | ☑ ⚠ 28 "Tags".zones[3].tuningMode | DEC | 1 | 1 | ☑ ⚠ 29 "Tags".multizoneTir.start | Bool | FALSE | TRUE | ☑ ⚠

No.	Action	Remark
52.	You can monitor the progress of the fine tuning through the courses of the curves in the commissioning windows (Figure 7-1) and in the watch table.	

	i	Name	...	Display format	Monitor value
25	// TIR				
26		"Tags".zones[1].tuningMode		DEC	1
27		"Tags".zones[2].tuningMode		DEC	2
28		"Tags".zones[3].tuningMode		DEC	1
29		"Tags".multizoneTir.start		Bool	🟩 TRUE
30		"Tags".multizoneTir.reset		Bool	□ FALSE
31		"InstMultizoneTir".busy		Bool	🟩 TRUE
32		"InstMultizoneTir".done		Bool	□ FALSE
33		"InstMultizoneTir".error		Bool	□ FALSE
34		"InstMultizoneTir".status		Hex	16#0000
35		"InstMultizoneTir".statStep		DEC	50
36		"InstMultizoneTir".statWaiting.tolerance		Floating-poi...	5.0
37		"InstMultizoneTir".instWaiting.deviation[1]		Floating-poi...	-0.0014720...
38		"InstMultizoneTir".instWaiting.deviation[2]		Floating-poi...	0.003824234
39		"InstMultizoneTir".instWaiting.deviation[3]		Floating-poi...	-0.0039978...
40		"InstMultizoneTir".statWaiting.cycles		DEC	100
41		"InstMultizoneTir".instWaiting.progress		Floating-poi...	0.0
42		"InstMultizoneTir".instWaiting.busy		Bool	□ FALSE
43		"InstMultizoneTir".instWaiting.done		Bool	🟩 TRUE
44		"InstPidTemp1".PIDSelfTune.TIR.ControlInReady		Bool	🟩 TRUE
45		"InstPidTemp1".PIDSelfTune.TIR.FinishControlIn		Bool	□ FALSE
46		"InstPidTemp2".PIDSelfTune.TIR.ControlInReady		Bool	□ FALSE
47		"InstPidTemp2".PIDSelfTune.TIR.FinishControlIn		Bool	□ FALSE
48		"InstPidTemp3".PIDSelfTune.TIR.ControlInReady		Bool	🟩 TRUE
49		"InstPidTemp3".PIDSelfTune.TIR.FinishControlIn		Bool	□ FALSE

Figure 8-1



8.2 Fine tuning in the event of strong coupling

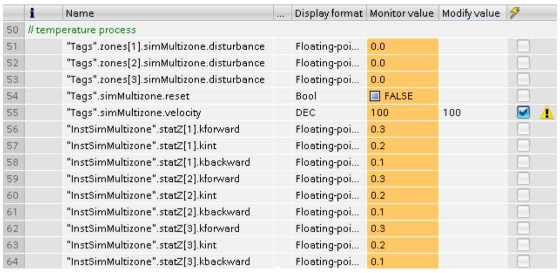
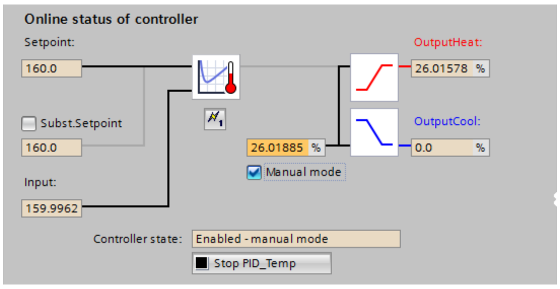
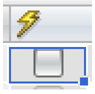
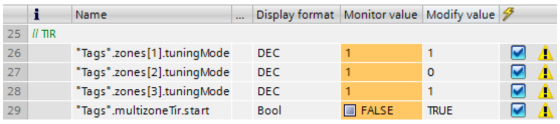
In the event of strong thermal coupling of the zones, the neighboring zones should not be fine-tuned simultaneously.
Here you should put the neighboring zones during the fine tuning to manual mode by maintaining the manipulated variable.

If the system tends to oscillate during the disturbance variable compensation after the fine tuning, this points towards a negative influence of the neighboring zones to finding parameters of the fine-tuned zones.

In chapter 7.3.1 zone 1 was not fine-tuned but left in automatic mode during the fine tuning of zones 2 and 3.
Analog to this, the neighboring zones are put into manual mode by the respective commissioning editor and are assigned with *"Tags".zones[x].tuningMode = 0* (do not carry out fine tuning) for the fine tuning energy type specification.

For the complete fine tuning of all zones, this approach is carried out once for even-numbered zones and is this then repeated with the odd-numbered zones.

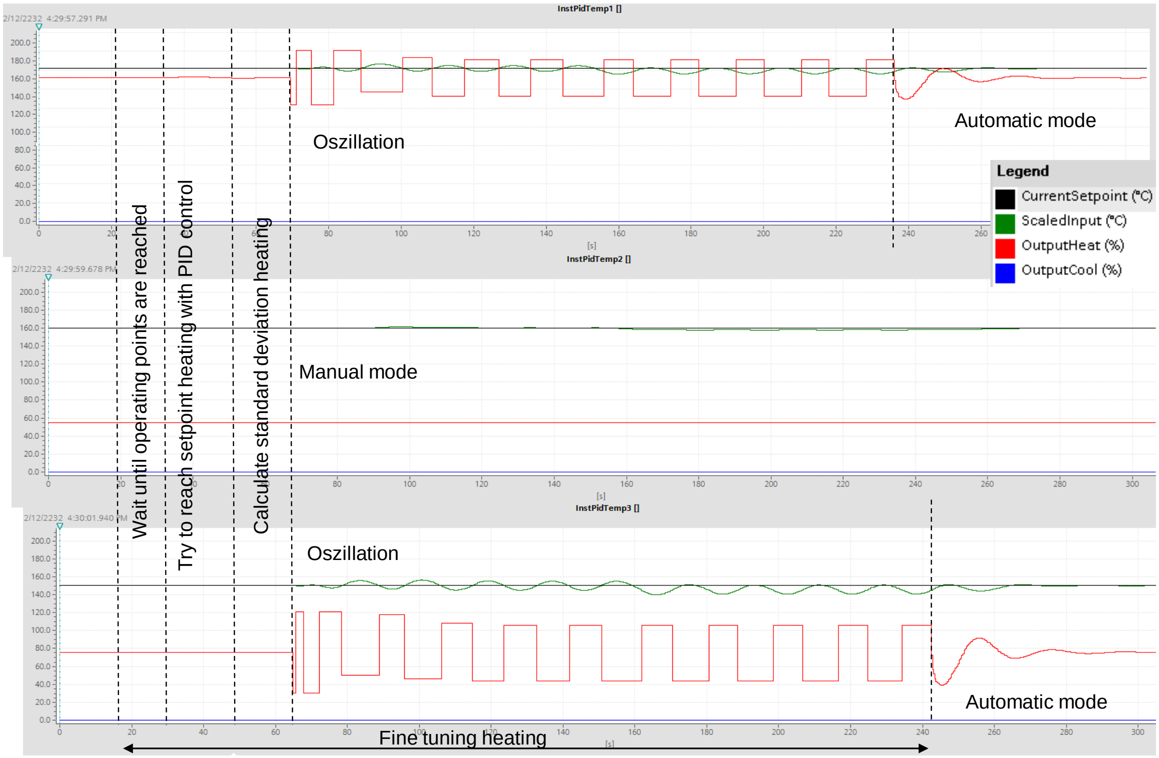
Table 8-2

<table>
<tr><th>No.</th><th>Action</th><th>Remark</th></tr>
<tr><td>53.</td><td>To increase the forwards coupling factors, set the medium transport velocity to 100 %:
<i>"Tags".simMultizone.velocity = 100.0</i></td><td>
<table>
<tr><th></th><th>i</th><th>Name</th><th>...</th><th>Display format</th><th>Monitor value</th><th>Modify value</th><th>⚡</th><th></th></tr>
<tr><td>50</td><td colspan="8">// temperature process</td></tr>
<tr><td>51</td><td></td><td>"Tags".zones[1].simMultizone.disturbance</td><td></td><td>Floating-poi...</td><td>0.0</td><td></td><td>□</td><td></td></tr>
<tr><td>52</td><td></td><td>"Tags".zones[2].simMultizone.disturbance</td><td></td><td>Floating-poi...</td><td>0.0</td><td></td><td>□</td><td></td></tr>
<tr><td>53</td><td></td><td>"Tags".zones[3].simMultizone.disturbance</td><td></td><td>Floating-poi...</td><td>0.0</td><td></td><td>□</td><td></td></tr>
<tr><td>54</td><td></td><td>"Tags".simMultizone.reset</td><td></td><td>Bool</td><td>□ FALSE</td><td></td><td>□</td><td></td></tr>
<tr><td>55</td><td></td><td>"Tags".simMultizone.velocity</td><td></td><td>DEC</td><td>100</td><td>100</td><td>☑</td><td>⚠</td></tr>
<tr><td>56</td><td></td><td>"InstSimMultizone".statZ[1].kforward</td><td></td><td>Floating-poi...</td><td>0.3</td><td></td><td>□</td><td></td></tr>
<tr><td>57</td><td></td><td>"InstSimMultizone".statZ[1].kint</td><td></td><td>Floating-poi...</td><td>0.2</td><td></td><td>□</td><td></td></tr>
<tr><td>58</td><td></td><td>"InstSimMultizone".statZ[1].kbackward</td><td></td><td>Floating-poi...</td><td>0.1</td><td></td><td>□</td><td></td></tr>
<tr><td>59</td><td></td><td>"InstSimMultizone".statZ[2].kforward</td><td></td><td>Floating-poi...</td><td>0.3</td><td></td><td>□</td><td></td></tr>
<tr><td>60</td><td></td><td>"InstSimMultizone".statZ[2].kint</td><td></td><td>Floating-poi...</td><td>0.2</td><td></td><td>□</td><td></td></tr>
<tr><td>61</td><td></td><td>"InstSimMultizone".statZ[2].kbackward</td><td></td><td>Floating-poi...</td><td>0.1</td><td></td><td>□</td><td></td></tr>
<tr><td>62</td><td></td><td>"InstSimMultizone".statZ[3].kforward</td><td></td><td>Floating-poi...</td><td>0.3</td><td></td><td>□</td><td></td></tr>
<tr><td>63</td><td></td><td>"InstSimMultizone".statZ[3].kint</td><td></td><td>Floating-poi...</td><td>0.2</td><td></td><td>□</td><td></td></tr>
<tr><td>64</td><td></td><td>"InstSimMultizone".statZ[3].kbackward</td><td></td><td>Floating-poi...</td><td>0.1</td><td></td><td>□</td><td></td></tr>
</table></td></tr>
<tr><td>54.</td><td>Wait until all zones have reached their operating point.</td><td>-</td></tr>
<tr><td>55.</td><td>Put the controller in the commissioning editor of zone 2 to manual mode.
The "OutputHeat” heat output maintains the last value and the status of the controller displays manual mode.</td><td>
Online status of controller
Setpoint: 160.0
□ Subst.Setpoint 160.0
Input: 159.9962
26.01885 %
☑ Manual mode
OutputHeat: 26.01578 %
OutputCool: 0.0 %
Controller state: Enabled - manual mode
■ Stop PID_Temp</td></tr>
<tr><td>56.</td><td>Remove all ticks in the option boxes for the selection of the tags to be controlled in the "WatchTable”.</td><td></td></tr>
<tr><td>57.</td><td>Select the predominant tuning energy types for the odd zones of the multi-zone controller
<ul>
<li>Zone 1: Heating
("Tags".zones[1].tuningMode = 1)</li>
<li>Zone 2: none
("Tags".zones[1].tuningMode = 0)</li>
<li>Zone 3: Heating
("Tags".zones[1].tuningMode = 1)</li>
</ul>
Start the synchronized fine tuning with these specifications by setting the bit ”<i>Tags".multizoneTir.start</i>.</td><td>
<table>
<tr><th></th><th>i</th><th>Name</th><th>...</th><th>Display format</th><th>Monitor value</th><th>Modify value</th><th>⚡</th><th></th></tr>
<tr><td>25</td><td colspan="8">// TIR</td></tr>
<tr><td>26</td><td></td><td>"Tags".zones[1].tuningMode</td><td></td><td>DEC</td><td>1</td><td>1</td><td>☑</td><td>⚠</td></tr>
<tr><td>27</td><td></td><td>"Tags".zones[2].tuningMode</td><td></td><td>DEC</td><td>1</td><td>0</td><td>☑</td><td>⚠</td></tr>
<tr><td>28</td><td></td><td>"Tags".zones[3].tuningMode</td><td></td><td>DEC</td><td>1</td><td>1</td><td>☑</td><td>⚠</td></tr>
<tr><td>29</td><td></td><td>"Tags".multizoneTir.start</td><td></td><td>Bool</td><td>□ FALSE</td><td>TRUE</td><td>☑</td><td>⚠</td></tr>
</table>

</td></tr>
</table>

No.	Action	Remark
58.	The synchronous fine tuning for the zones 1 and 3 is carried out.	See Figure 8-2
59.	Then repeat this procedure for synchronous fine tuning for the even zones. To do this, the odd zones (1 and 3) are put in steady-state control loop in manual mode and zone 2 is fine tuned.	-

Figure 8-2



9 Links & Literature

Table 9-1

	Topic
\1\	Siemens Industry Online Support https://support.industry.siemens.com
\2\	Download page of the entry https://support.industry.siemens.com/cs/ww/en/view/109740463
\3\	Function Manual: SIMATIC S7-1200, S7-1500 PID Control https://support.industry.siemens.com/cs/ww/en/view/108210036
\4\	SIMATIC S7 S7-1200 Programmable controller manual https://support.industry.siemens.com/cs/ww/en/view/109741593
\5\	TIA Selection Tool http://w3.siemens.com/mcms/topics/en/simatic/tia-selection-tool/Pages/tab.aspx
\6\	Application example: "Closed-Loop Control of Simulated Controlled Systems in the S7-1500 with PID_Compact V2.2" https://support.industry.siemens.com/cs/ww/en/view/79047707
\7\	Günther Schmidt: Simulationstechnik Published by: Oldenbourg Wissenschaftsverlag (1980) ISBN-13: 978-3486251814
\8\	Manual SIMATIC S7-1500, ET 200MP Automation system https://support.industry.siemens.com/cs/ww/en/view/59191792
\9\	SIMATIC STEP 7 Basic/Professional V15.1 and SIMATIC WinCC V15.1 https://support.industry.siemens.com/cs/ww/en/view/109755202
\10\	Manual SIMATIC S7-1500/ET 200MP Automation system In a nutshell https://support.industry.siemens.com/cs/ww/en/view/109481357

10 History

Table 10-1

Version	Date	Modifications
V1.0	12/2016	First version
V1.1	07/2019	Update TIA Portal V15.1