BECKHOFF New Automation Technology

Manual | EN

TS4110

TwinCAT 2 | PLC Temperature Controller

Supplement | Control

2022-10-25 | Version: 1.1

Table of contents

1 Foreword

1.1 Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with applicable national standards.
It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.
It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without prior announcement.
No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.
Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
with corresponding applications or registrations in various other countries.

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

Copyright

1.2 Safety instructions

Safety regulations

Please note the following safety instructions and explanations!
Product-specific safety instructions can be found on following pages or in the areas mounting, wiring, commissioning etc.

Exclusion of liability

All the components are supplied in particular hardware and software configurations appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

Personnel qualification

This description is only intended for trained specialists in control, automation and drive engineering who are familiar with the applicable national standards.

Description of symbols

In this documentation the following symbols are used with an accompanying safety instruction or note. The safety instructions must be read carefully and followed without fail!

⚠ DANGER

Serious risk of injury!

Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.

⚠ WARNING

Risk of injury!

Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.

⚠ CAUTION

Personal injuries!

Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.

NOTE

Damage to the environment or devices

Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.

Tip or pointer

This symbol indicates information that contributes to better understanding.

1.3 Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our https://www.beckhoff.com/secguide.

Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

To stay informed about information security for Beckhoff products, subscribe to the RSS feed at https://www.beckhoff.com/secinfo.

2 Overview

The TwinCAT temperature controller is a universally applicable PLC block for monitoring and controlling a wide variety of temperature-dependent processes. The controller can be operated in

- automatic (closed loop) and
- manual (open loop)

modes.

The control value can be accessed in digital or analogue form. The digital control value is pulse width modulated (PWM). A two-point or three-point output is also available. The control value is limited to the permitted maximum and minimum values.

The set value is also limited to permitted minimum and maximum values, and can also be ramped. A bit is available in the interface to the block that provides easy switching from the set value to a standby set value. A soft start can be parameterised to support "heater baking". This involves the set value (optionally ramped) being initially set to a low value, remaining there for a certain time, then being changed to the true set value (again optionally ramped).

The actual value can be digitally filtered.

The control algorithm is PID-based. An additional pre-regulator can be inserted in order to minimise overshoot.

The controller has a variety of parameterisable monitoring functions. There is

- tolerance band monitoring (two different tolerance bands),
- absolute value monitoring,
- sensor monitoring (open, back voltage, reverse) and
- monitoring of the heating current (open, short circuit, leakage current).

There is an algorithm for determination of optimal controller parameters that greatly simplifies the process of commissioning the controller. This algorithm evaluates a step, and uses a method of inflectional tangents to determine the maximum speed and delay time of the loop. This data allows a controller to be specified according to the rules of Chien, Hrones and Reswick. The parameters for the pre-controller are also determined here. If the controller parameters are already known, then the controller can also be operated using these externally supplied parameters.

Commissioning the controller in stages [▶ 16]

Documentation of the Function Block [▶ 20] and the structures [▶ 22].

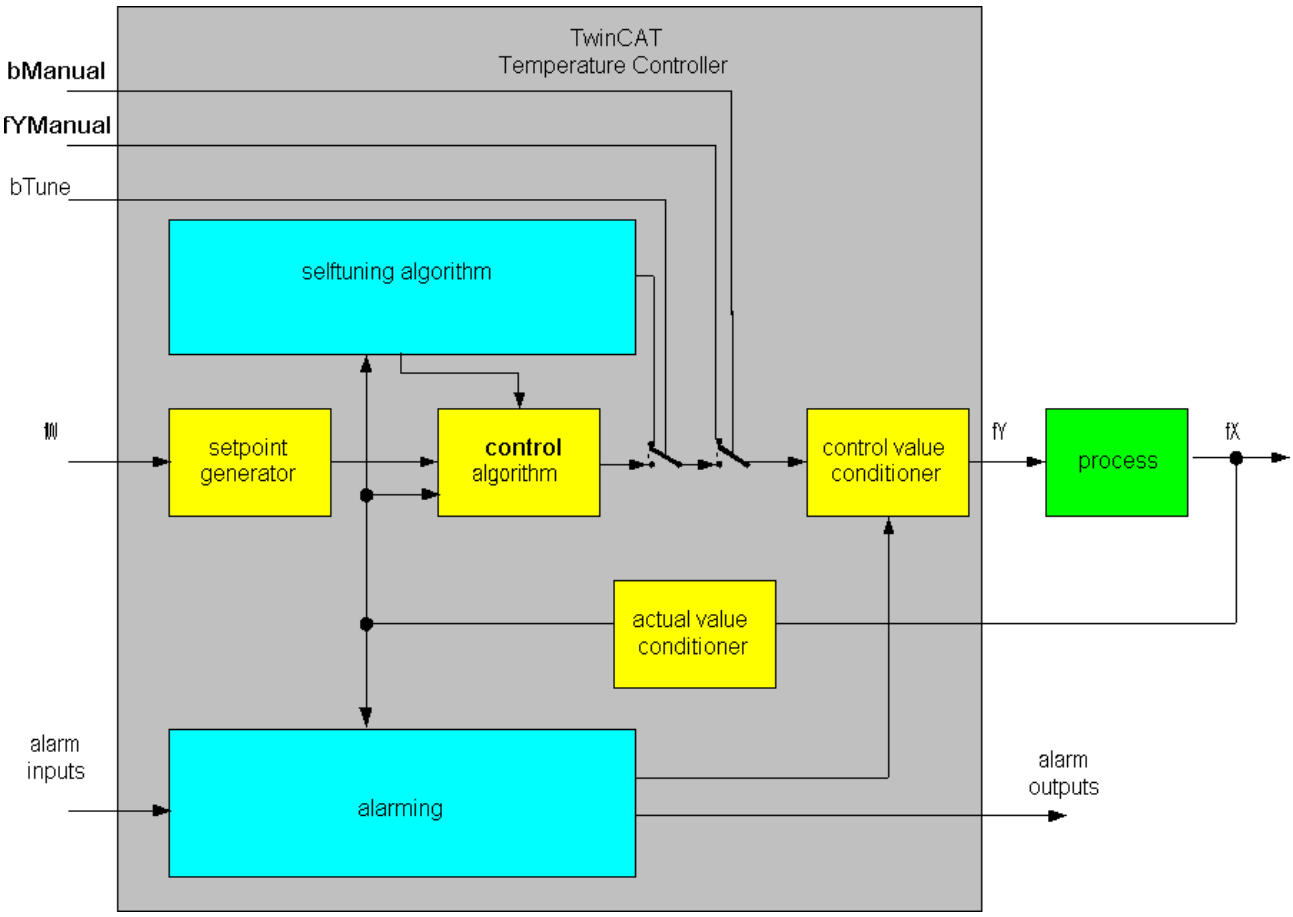
For compatibility reasons the documentation of the old Temperature Controller [▶ 30] Function Block and structure is also included in this document.

3 Block Diagram

The TwinCAT Temperature Controller consists of a number of function blocks. The following function blocks are involved:

- Self-tuning algorithm (FB_Selftuner)
- Control algorithm (FB_ControlAlgorithm)
- Setpoint generator (FB_SetpointConditioner)
- Control value generator (FB_ControlValueConditioner)
- Alarming (FB_Alarming)

These function blocks in turn call a number of other subsidiary function blocks.

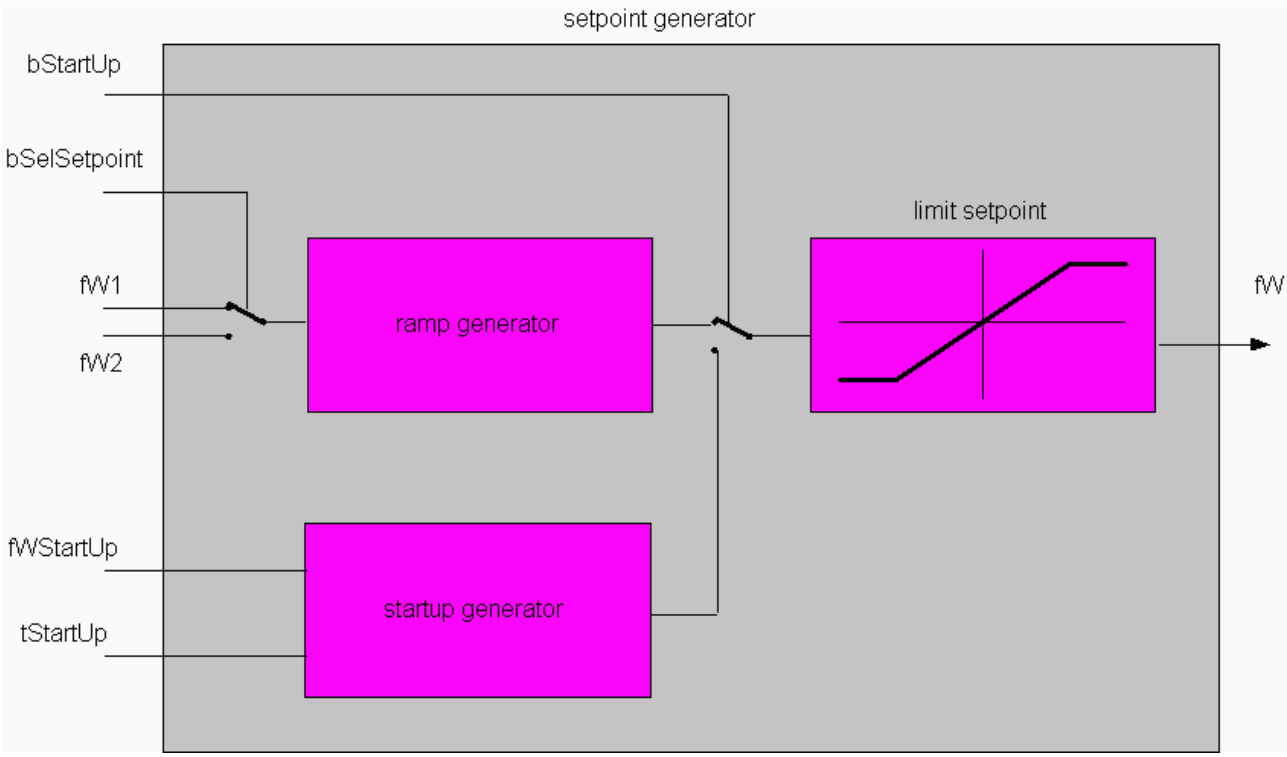


The diagram illustrates the individual function blocks.

4 Generating the Set Value

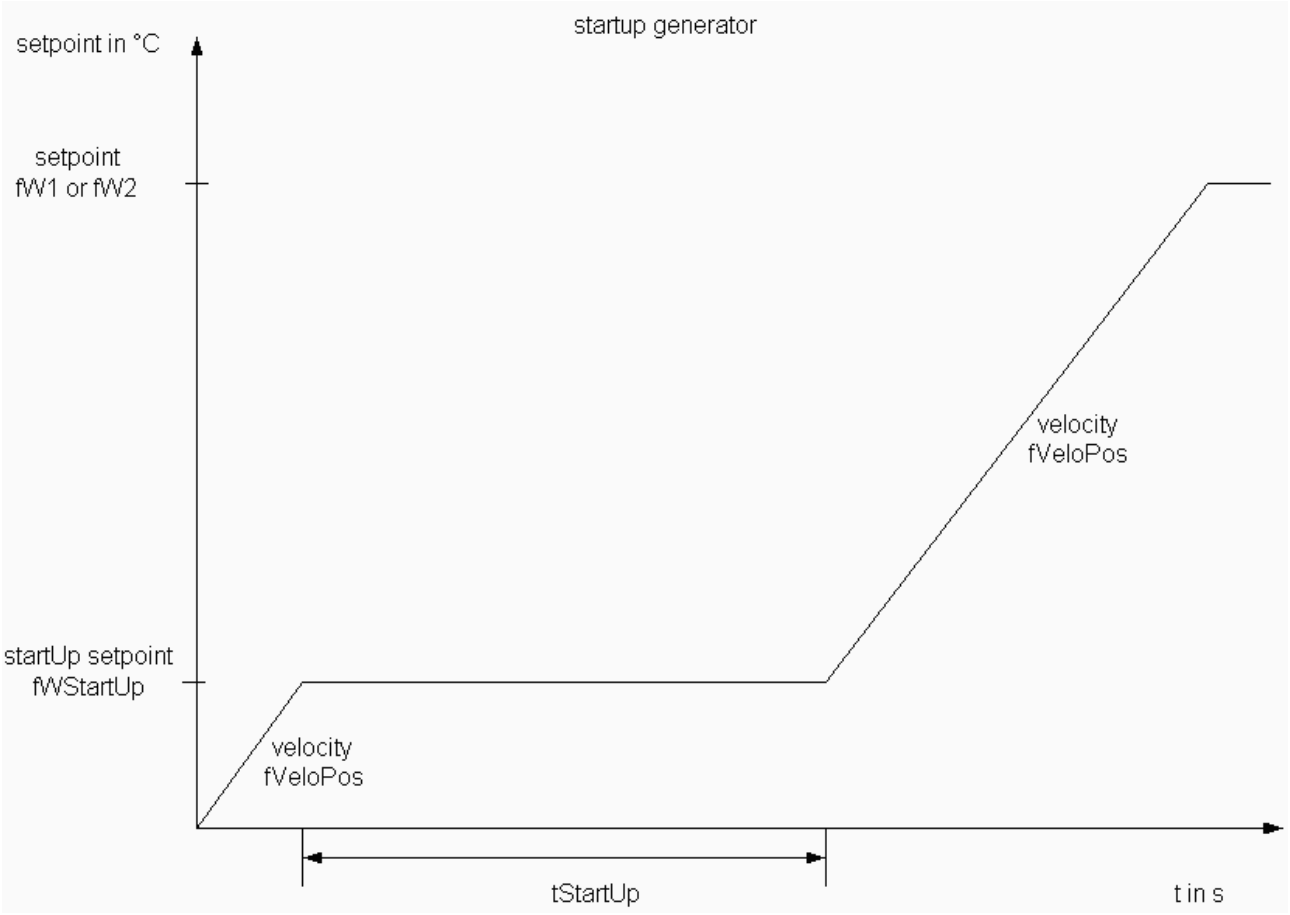
One bit switches between the setpoints. In addition to the actual setpoint, there is also a standby setpoint. The standby setpoint can be used to reduce the temperature during operating pauses to a lower value in order to save power. If necessary the steps in the setpoint can be ramped. The parameter set for the setpoints includes a rate of rise and a rate of fall.

The setpoints are restricted to their limits.



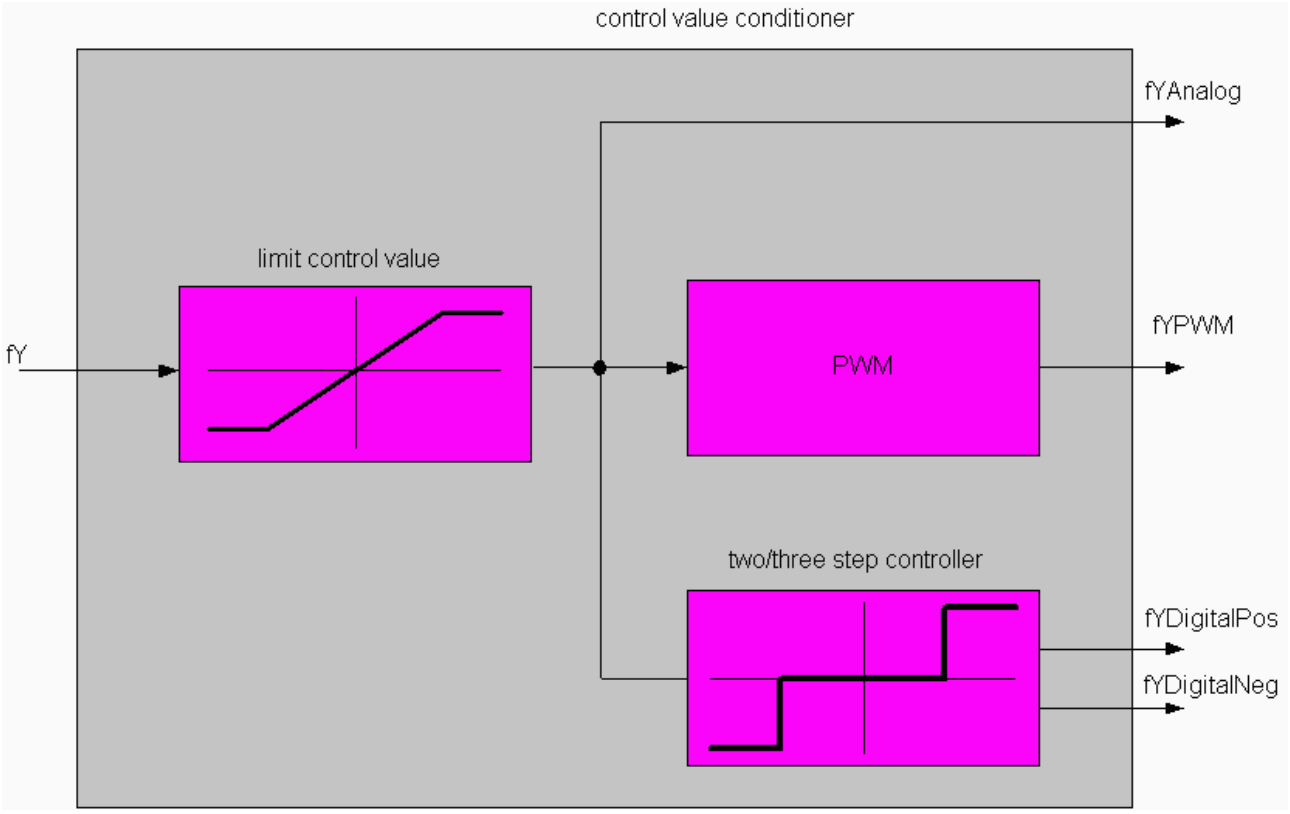
In order to permit "heater baking", a soft start can be parameterized. In this case, the temperature is first ramped up from ambient to a low setpoint (`fWStartUp`). This temperature is then maintained for a period of time (`tStartUp`), and only after that has elapsed does the ramp up to the actual setpoint begin.

Start-up

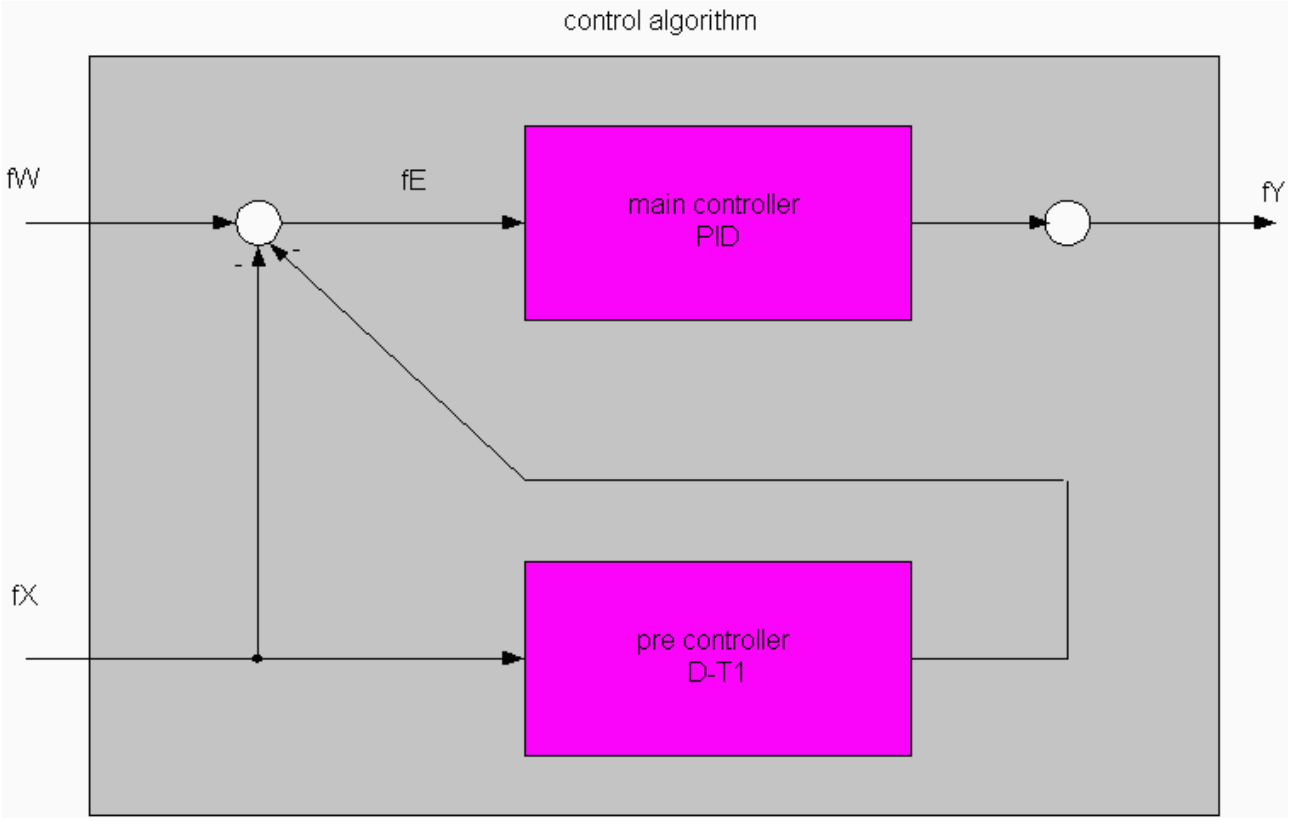

startup generator
setpoint in °C
setpoint
fW1 or fW2
velocity
fVeloPos
startUp setpoint
fWStartUp
velocity
fVeloPos
tStartUp
t in s

5 Generating the Control Value

The **C**ontrol **V**alue calculated by the controller is initially limited to valid values. The values of the limits are passed to the controller function block via the control value structure. The control value is made available in three different ways. The control value can be picked up in analog form. However it is more usual for the digital output to take the form of a pulse width modulated signal. The cycle time required for the pulse width modulation is supplied to the controller in the control value structure. Additionally, a two-point output (for heating or cooling) and a three-point output (for heating and cooling) can be connected.

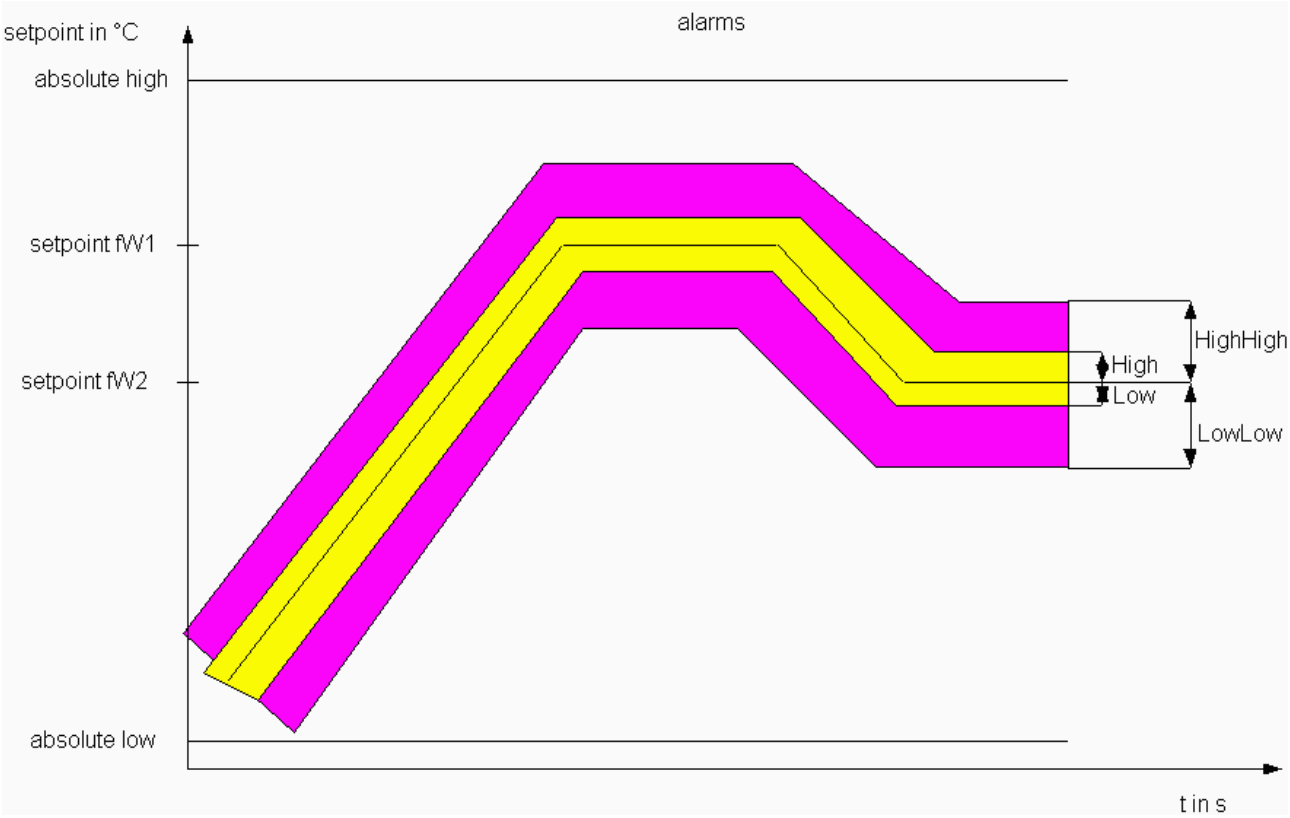


6 Control Algorithm



The heart of the TwinCAT Temperature Controller is a standard PID controller. This controller kernel also supports anti-reset windup measures to limit the I-component if the control value is subjected to limiting. Since the controller has been designed to minimise disturbances using the adjusting procedure according to Chien, Hrones and Reswick, overshoot is possible when the set point is changed. In order to reduce such overshoot, a pre-controller can be inserted to handle changes in the set point. The pre-regulator has a D-T1 characteristic, and reduces ringing in the controller as a whole. Since the D component of the pre-controller has the effect of "roughening" the control value, the use of a pre-controller must be considered very carefully. The pre-controller is switched off when the actual value enters within a certain range of the set value and remains there for some length of time. The pre-controller is switched off by ramping it down over a considerable period of time. To minimise oscillation of the control value, it is optionally possible to follow the main controller with a filter. P-T1 and moving average filters are available for this purpose.

7 Alarming



The following alarm conditions are continuously monitored by the temperature controller:

- Absolute temperatures (high and low)
- Relative temperatures (in two bands around the setpoint)

The following hardware conditions related to the sensor can also be linked to the temperature controller:

- Open Thermocouple: broken wire to the temperature sensor.
- Back Voltage: a voltage outside the permitted range is present at the temperature sensor.
- Reverse Thermocouple: temperature sensor is connected with the wrong polarity.

If a current sensor is connected, then the following signals can be linked to the temperature controller:

- Short circuit
- open circuit
- Leakage current

8 Self-tuning

The self-tuning algorithm is based on the classic inflectional tangents method. This method was first developed by Ziegler and Nichols. It is assumed that a linear P-T1 loop with a delay time is being examined. The maximum rate of change is determined following an experimental step. This is achieved through examining the differences over a number of samples. A tangent is constructed to the point where the rate of change is a maximum, and its intersection with the time axis is found. The delay time, Tu, is the time from the start of the measurement up to the intersection of the inflection tangent and the time axis. Knowing Tu and Vmax, the Chien, Hrones and Reswick formula yields the controller parameters for suppression of disturbances with 20% overshoot. The parameters for the pre-controller can easily be derived from the parameters for the main controller with the aid of heuristic formulae. After completion of the self-tuning these parameters are used in an automatic switch to closed loop operation.

9 Commisioning the Controller in Stages

The following steps must be taken:

- **The controller library must be added to the project using the Library Manager.**

TcTempCtrl.lib is to be added in the Library Manager.

- **At least one instance of the controller must be programmed.**

An instance of the `FB_TempController` controller function block is to be created. Also create an instance of the structure `ST_ControllerParameter`.

- **Perform the required external connection.**

Name		Description
`eCtrlMode`	Connection necessary	Switches the controller to an operation mode (active, passive, tuning)
`bSelSetpoint`	Connection optional	Selects one of two possible setpoints. FALSE selects the normal setpoint, while TRUE selects the standby setpoint.
`fW1`	Connection necessary	Setpoint.
`fW2`	Connection optional	Standby setpoint, generally lower than fW1. fSelSetpoint can be used to switch between fW1 and fW2.
`fX`	Connection necessary	Actual value. This value must be converted to LREAL.
`fYManual`	Connection optional	Control value in manual operation.
`bOpenThermocouple`	Connection optional	The thermocouple is open if TRUE. Must be reported by the hardware (e.g. KLxxxx).
`bReverseThermocouple`	Connection optional	TRUE reports that the thermocouple has been connected with the wrong polarity. Must be reported by the hardware.
`bBackVoltage`	Connection optional	TRUE indicates that the input voltage at the thermocouple is too high. Must be reported by the hardware.
`bLeakage`	Connection optional	TRUE indicates that leakage current has been detected at the heating element. Must be reported by the hardware.
`bShortCircuit`	Connection optional	TRUE indicates that a short circuit has been detected at the heating element. Must be reported by the hardware.
`bOpenCircuit`	Connection optional	TRUE indicates that an open circuit has been detected at the heating element. Must be reported by the hardware.
`sControllerParameter`	Connection necessary	General parameters (sampling time etc.) are passed to the function block in this structure.
`sParaControllerExternal`	Connection optional	An external controller parameter set is passed to the function block in this structure.

- **Perform the necessary parameterization of the controller via the structure.**

The parameters can be specified through initial values, or by assignment. If the parameters are assigned by initial values, it could look like this, for example:

```
(* parameters *)
sControllerParameter : ST_CTRL_TempCtrlParameter :=
(
(* base *)
tCtrlCycleTime := t#1000ms,
tTaskCycleTime := t#10ms,
```

```
fYMin := -100,
fYMax := 100,
tPWMCycleTime := t#100ms ,
fYManual := 20,
bFilter := FALSE,
tFilter := t#100ms,
bDeadband := FALSE,
fEDeadband := 1.0, (* deadband *)
fWMin := 15,
fWMax := 60,
fWStartUp := 20.0,
tStartUp := t#160s,
fWVeloPos := 0.01,
fWVeloNeg := 0.01,
bStartUpRamping := FALSE,
fWStartUpVeloPos := 0.1,
fWStartUpVeloNeg := 0.1,
iMode := eCTRL_ControlMode_HEATING,
dwAlarmSupp := 16#FF_FF_FF_FF,
bSelCtrlParameterSet:= FALSE,

(* tuninig *)
iTuningMode := eCTRL_TuneMode_heating,
fYTuneHeating := 100.0,
fYTuneCooling := -100.0,
fEndTunePercentHeating := 80.0, (* switch to closed loop control when X > 0.8*W *)
fEndTunePercentCooling := -70.0, (* switch to closed loop control when X < 0.2*W *)

iReactionOnFailure := eCTRL_ReactionOnFailure_StopController,
TempLow := -50.0,
TempLowLow := -100.0,
TempHigh := 100.0,
TempHighHigh := 155.0,
TempAbsoluteHigh := 150.0,
TempAbsoluteLow := -95.0,
bEnablePreController := FALSE,
bEnableZones := FALSE,
bEnableCVFilter := FALSE,
iFilterType := eCTRL_FilterType_AVERAGE,
iControllerType := eCTRL_ControllerType_PID
);
```

Assignment in the code may look as follows in ST:

```
sControllerParameter.tPWMCycleTime :=
t#100ms;
```

- **Specification of the controller sampling time, the task cycle time and the PWM cycle time**

The controller's sampling time must be adapted to the section. It should be selected to be equal to or less than one tenth of the section's dominant time constants. The task cycle time is specified by the PLC task from which the controller function block has been called. This value can be read from the task configuration (PLC Control: Resources Task Configuration). The PWM cycle time is usually equal to the controller cycle time. If the task cycle time is 10 ms and the chosen PWM cycle time (=controller sampling time) is 100 ms, then a total of 10 levels (PWM cycle time / task cycle time) are available.

- **Parameterization of TwinCAT Scope**

To check the results, a scope recording should always be made of the tuning process and the closed loop control behavior. To do this, TwinCAT Scope View should be started and parameterized. The following channels should be recorded: setpoint (fW1 or fW2), actual value (fX) and the analog control value (fYAnalog).

- **Switching off the alarms during the commissioning phase**

The alarms can be temporarily switched off during the commissioning phase. An appropriate bit mask must be written into the dwAlarmSupp Dword. If a bit is set in this Dword, the corresponding alarm is disabled. The assignment of the individual alarms is described here [▶ 30].

All required alarms should be switched on again after initial commissioning!

- **Starting the controller with tuning**

If the controller parameters are to be determined by tuning, the control mode must be set to `eCTRL_MODE_TUNE`. Firstly, a fixed waiting time of 20 s elapses. During this waiting time, the temperature of the section is monitored to ensure that it remains within a +-1°C band. If the temperature goes outside this band, the waiting time starts again. The section is then subjected to a step excitation with a control value of `fYTune`. The section then reacts with the step response. As long as 80% of the setpoint has not been reached, the section parameters are determined using the inflectional tangent method. For safety reasons, control is switched over to closed control loop once 80% of the setpoint has been reached. If the temperature reaches the 80% mark too quickly (with no clear inflection) then the value of `fYTune` is to be reduced. The determined parameters are used for the PID controller, and are provided in a structure at the output of the controller.

NOTE

The section must perform a step of at least 40°C for the purposes of tuning. Smaller steps can result in incorrect parameter determination!

NOTE

Once the tuning has been completed successfully, the `eCtrlState` is set to `eCTRL_STATE_TUNED`. The controller enters standby mode. Closed-loop operation with the estimated parameters can only be activated by setting the control mode to `eCTRL_MODE_ACTIVE`.

- **Linking the internal control parameters with the external connections**

The controller parameters determined in the tuning process can be supplied to the controller´again as external parameters. This may be necessary if the tuning is only to be carried out once (e.g. only during the initial commissioning). To do this, the `sParaControllerInternal` structure is fed back to the controller's `sParaControllerExternal` input, and the `bSelCtrlParameterSet` flag set to TRUE.

- **Fine tuning**

The control parameters determined in the tuning process are designed to produce fast settling, with about 10% overshoot. If only very little overshoot is permitted, or even none at all, then the following parameters from the `ST_ControllerParameter` structure can be used to perform fine tuning. These values should be considered only as a guide.

Behavior	fTuneKp	fTuneTn	fTuneTv	fTuneTd
Fast settling with an overshoot of 10% – 20%	1.2	2.0	0.42	0.25
Slower settling with less overshoot	1.0	2.5	0.42	0.25
Almost asymptotic settling with extremely low overshoot	0.5	3.0	1.0	0.25

10 Sample

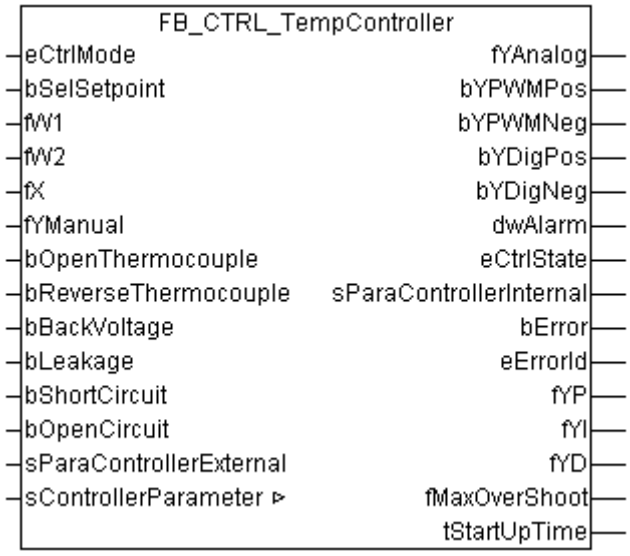
The Sample program https://infosys.beckhoff.com/content/1033/tcplctempcontrol/Resources/11292577291/.zip has one `MAIN_Simu` program. The process is a simulated second order process with optional noise on the output. Use TwinCAT ScopeView to sample the data to a TwinCAT ScopeView.

11 PLC-API

11.1 FB_CTRL_TempController

The temperature controller function block FB_CTRL_TempController has various inputs and outputs, which are described below. All the controller's parameters are passed to it via structures. The structures and enums are defined here [▶ 37].

Function block



Interface

```
VAR_INPUT
    eCtrlMode : E_CTRL_MODE; (* controller mode: passive, activ,check *)
    bSelSetpoint : BOOL; (* FALSE = setpoint 1, TRUE = setpoint 2*)
    fW1                     : FLOAT; (* first setpoint *)
    fW2                     : FLOAT; (* second setpoint *)
    fX                      : FLOAT; (* actual value *)
    fYManual                : FLOAT; (* manual control value *)
(* alarming inputs *)
    bOpenThermocouple       : BOOL; (* thermocouple *)
    bReverseThermocouple    : BOOL;
    bBackVoltage            : BOOL;
    bLeakage                : BOOL; (* heating system *)
    bShortCircuit           : BOOL;
    bOpenCircuit            : BOOL;
    sParaControllerExternal : ST_CTRL_ParaController; (* external controller parameter set *)
END_VAR
VAR_IN_OUT
    sControllerParameter : ST_CTRL_TempCtrlParameter; (* controller
parameter set *)
END_VAR
VAR_OUTPUT
    (* control output *)
    fYAnalog : FLOAT; (* control value analog *)
    bYPWMPos : BOOL; (* control value PWM *)
    bYPWMNeg : BOOL; (* control value PWM *)
    bYDigPos : BOOL; (* 3-Point: control value digital positive*)
    bYDigNeg : BOOL; (* 3-Point: control value digital negative*)
    (* alarming *)
    dwAlarm  : DWORD; (* max. 32 alarms *)
    (*quality of control*)
    fMaxOverShoot : FLOAT;
    tStartUpTime  : TIME;
    (* state *)
    eCtrlState    : E_CTRL_STATE := eCTRL_STATE_IDLE;
    (* controller parameter *)
    sParaControllerInternal : ST_CTRL_ParaController;
```

```
    (* general errors *)
    bError : BOOL;
    eErrorId : E_CTRL_ErrorCodes;
END_VAR
```

Inputs

Name	Unit	Value range	Description
eControlMode	1	E_CTRL_MODE	Mode switching.
bSelSetpoint	1	[TRUE,FALSE]	Selects one of two possible setpoints. FALSE selects the normal setpoint, while TRUE selects the standby setpoint.
fW1	°C	LREAL	Setpoint.
fW2	°C	LREAL	Standby setpoint, generally smaller than fW1. fSelSetpoint can be used to switch between fW1 and fW2.
fX	°C	LREAL	Actual value. This value must be converted to LREAL.
fYManual	-100% - +100%	LREAL	Control value in manual operation.
bOpenThermocouple	1	[TRUE,FALSE]	The thermocouple is open if TRUE. Must be indicated by the hardware (e.g. KLxxxx).
bReverseThermocouple	1	[TRUE,FALSE]	TRUE indicates that the thermocouple has been connected with the wrong polarity. Must be indicated by the hardware.
bBackVoltage	1	[TRUE,FALSE]	TRUE indicates that the input voltage at the thermocouple is too high. Must be indicated by the hardware.
bLeakage	1	[TRUE,FALSE]	TRUE indicates that leakage current has been detected at the heating element. Must be indicated by the hardware.
bShortCircuit	1	[TRUE,FALSE]	TRUE indicates that a short circuit has been detected at the heating element. Must be indicated by the hardware.
bOpenCircuit	1	[TRUE,FALSE]	TRUE indicates that an open circuit has been detected at the heating element. Must be indicated by the hardware.
sControllerParameter	none	Structure	General parameters (sampling time etc.) are passed to the function block in this structure.
sParaControllerExternal	none	Structure	An external controller parameter set is passed to the function block in this structure.

Outputs

Name	Unit	Value range	Description
fYAnalog	none	LREAL	Analog control value.
bYPWMPos	none	[TRUE,FALSE]	Boolean output, pulse width modulated. Positive/heating mode
bYPWMNeg	none	[TRUE,FALSE]	Boolean output, pulse width modulated. Negative/cooling mode

Name	Unit	Value range	Description
bYDigPos	none	[TRUE,FALSE]	Boolean output of a three-step controller (TRUE control value 100%, FALSE control value off)
bYDigNeg	none	[TRUE,FALSE]	Boolean output of a three-step controller (TRUE control value -100%, FALSE control value off)
dwAlarm	none	DWORD	Alarm messages (see ENUM ...)
fMaxOverShoot	°C	LREAL	max. overshoot in °C above/below setpoint.
tStartUpTime	TIME	-	Startup time until the setpoint is reached for the first time.
eCtrlState	none	E_CTRL_STATE	Current controller state (see ENUM ...)
sParaControllerInternal	none	Structure	In this structure the internal controller parameter set (determined by the tuning) is made available.
bError	none	[TRUE,FALSE]	If an error is present, then bError is TRUE.
iErrorId	none	INT	If bError is TRUE, then iErrorId provides an error code (see ENUM ...)

11.2 Structure definitions

ST_ControllerParameter

```
TYPE ST_CTRL_TempCtrlParameter:
STRUCT

(*************************************************************************)

    (* general parameters *)
    iMode : E_CTRL_ControlMode; (* 1=heating, 2=cooling,3=heating&cooling *)
    iReactionOnFailure : E_CTRL_ReactionOnFailure; (* 0=controller
off, 1=manual op, 2=yMin, 3=yMax *)
    bSelCtrlParameterSet : BOOL; (* FALSE = internal set, TRUE = external set *)
    dwAlarmSupp    : DWORD; (* alarm suppression *)
    tCtrlCycleTime : TIME; (* controller cycle time *)
    tTaskCycleTime : TIME; (* plc task cycle time *)

(*************************************************************************)

    (* tuning parameteer *)
    iTuningMode : E_CTRL_TuneMode; (* only heating, only cooling,
first heating then cooling or vice versa *)
    tTuneStabilisation : TIME := t#20s; (* wait for a stable system*)
    fEndTunePercentHeating : FLOAT := 80.0; (* switch to closed loop control when X > 0.8*W *)
    fYTuneHeating : FLOAT; (* step change while tuning operation*)
    fYStableHeating : FLOAT; (* tuning operation *)
    fEndTunePercentCooling : FLOAT := 20.0; (* switch to closed loop control when X < 0.2*W *)
    fYTuneCooling : FLOAT; (* step change while tuning operation*)
    fYStableCooling : FLOAT; (* tuning operation *)
    fScalingFactor : FLOAT := 1.0; (* Scaling factor for KP heating/cooling *)

(*************************************************************************)

    (* setpoint parameters  *)
    fWMin : FLOAT; (* lower limit *)
    fWMax : FLOAT; (* upper limit *)
    (* start up *)
    bEnableSoftStart : BOOL; (* FALSE = no soft start, TRUE = soft start *)
    bEnableRamping : BOOL; (* FALSE = no ramping, TRUE = ramping*)
    fWStartUp : FLOAT; (* soft start plateau setpoint *)
    tStartUp : TIME; (* soft start waiting time*)
    bStartUpRamping : BOOL; (* enable ramping while start up phase*)
    fWStartUpVeloPos : FLOAT; (* max gradient for increasing setpoint in start up phase*)
    fWStartUpVeloNeg : FLOAT; (* max gradient for decreasing setpoint in start up phase *)
    fWVeloPos : FLOAT; (* max gradient for increasing setpoint*)
    fWVeloNeg : FLOAT; (* max gradient for decreasing setpoint*)
```

```
(*************************************************************************)

    (* actual value parameters      *)
    bFilter : BOOL;
    tFilter : TIME;

(*************************************************************************)

    (* deadband parameters      *)
    bDeadband : BOOL;
    fEDeadband : FLOAT; (* deadband *)

(*************************************************************************)

    (* control value parameters      *)
    fYMin : FLOAT; (* lower limit *)
    fYMax : FLOAT; (* upper limit *)
    fYManual : FLOAT;  (* manual operation*)
    fYOnFailure : FLOAT; (* control value on failure *)
    tPWMCycleTime : TIME; (* PWM: period *)
    tPWMMinOffTime : TIME; (* PWM: min off time *)
    tPWMMinOnTime : TIME; (* PWM: min on time *)
    tPWMWaitingTime : TIME; (* PWM: min waiting time *) (* not yet implemented !!!!*)
    fYThresholdOff : FLOAT; (* 3-Point: Off threshold *)
    fYThresholdOn : FLOAT; (* 3-Point: On threshold *)
    nCyclesForSwitchOver : INT := 100;

(*************************************************************************)

    (* controller settings       *)
    bEnablePreController: BOOL; (* enable precontroller *)
    bEnableZones : BOOL; (* enable zone around setpoint with open loop control *)
    bEnableCVFilter : BOOL; (* enable filter for CV (type see iFilterType) *)
    iFilterType : E_CTRL_FilterType; (* filtertype of CV filter*)
    iControllerType : E_CTRL_ControllerType; (* used controller normally PID *)

(*************************************************************************)

    (* min max temperatures       *)
    TempLow      : FLOAT;
    TempLowLow : FLOAT;
    TempHigh : FLOAT;
    TempHighHigh : FLOAT;
    TempAbsoluteHigh : FLOAT;
    TempAbsoluteLow : FLOAT;

(*************************************************************************)

    (* internal tuning parameters *)
    fTuneKp : FLOAT := 1.2;
    fTuneTn : FLOAT := 2.0;
    fTuneTv : FLOAT := 0.42;
    fTuneTd : FLOAT := 0.25;
END_STRUCT
END_TYPE
```

General parameters

Name	Unit	Value range	Description
iMode	none	INT	Controller operation mode (1 = heating, 2 = cooling, 3 = heating & cooling) (see below) [▶ 29]
iReactionOnFailure	none	INT	Parameterizable reaction to errors (see below) [▶ 28]
bSelCtrlParameterSet	none	BOOL	TRUE = external parameter set, FALSE = internal parameter set (determined by tuning)
dwAlarmSupp	none	DWORD	Masks out the alarms (see below) [▶ 30]

Name	Unit	Value range	Description
tCtrlCycleTime	s	TIME	Controller's sampling time. In the course of the sampling time the controller re-calculates the control value.
tTaskCycleTime	s	TIME	Task cycle time. The FB is called with this time interval.

Tuning parameters

Name	Unit	Value range	Description
iTuningMode	K	E_CTRL_TuneMode	Determination of the tuning sequence (see below.)
tTune stabilization	s	TIME	Waiting time until the section is stable for tuning.
fEndTunePercentHeating	%	(L)REAL	Percentage value of setpoint, from which the system switches to Closed Loop Control.
fYTuneCooling	K	(L)REAL	Step change in control value during tuning.
fYStableCooling	K	(L)REAL	Control value when switching to tuning during cooling.
fScalingFactor	none	(L)REAL	Scaling factor for parameter switching if no tuning is performed for cooling.

Setpoint parameters

Name	Unit	Value range	Description
fWMin	K	(L)REAL	Minimum setpoint.
fWMax	K	(L)REAL	Maximum setpoint.
bEnableSoftStart	none	BOOL	FALSE = no soft start, TRUE = soft start
bEnableRamping	none	BOOL	FALSE = no ramping, TRUE = ramping
fWStartUp	K	(L)REAL	Setpoint at start-up.
tStartUp	s	TIME	Time with the fWStartUp setpoint.
bStartUpRamping	none	[TRUE,FALSE]	Switches on ramping during the start-up phase.
fWStartUpVeloPos	K/s	(L)REAL	Rate of rise (of ramp) during the start-up phase.
fWStartUpVeloNeg	K/s	(L)REAL	Rate of fall (of ramp) during the start-up phase
fWVeloPos	K/s	(L)REAL	Rate of rise (of ramp).
fWVeloNeg	K/s	(L)REAL	Rate of fall (of ramp).

Actual value parameters

Name	Unit	Value range	Description
tFilter	s	TIME	Time constant of the actual value filter (first order P-T1 filter)
bFilter	none	[TRUE,FALSE]	The actual value filter is actuated if TRUE.

Deadband parameters

Name	Unit	Value range	Description
bDeadband	none	[TRUE,FALSE]	TRUE = deadband on, FALSE = deadband off
fEDeadband	K	(L)REAL	Size of the deadband in degrees.

Control value parameters

Name	Unit	Value range	Description
fYMin	none	(L)REAL	Minimum value of the control value.
fYMax	none	(L)REAL	Maximum value of the control value.
fYManual	none	(L)REAL	Control value in manual operation.
fYOnFailure	none	(L)REAL	Control value in case of error (parameterizable).
tPWMCycleTime	s	TIME	Cycle time of the PWM signal.
tPWMMinOffTime	s	TIME	PWM: minimum switch-off time
tPWMMinOnTime	s	TIME	PWM: minimum switch-on time
tPWMWaitingTime	s	TIME	PWM: waiting time when switching from heating to cooling
fYThresholdOff	%	(L)REAL	3-point: switch-off threshold
fYThresholdOn	%	(L)REAL	3-point: switch-on threshold
nCyclesForSwitchOver	none	INT	Number of cycles for transition from one parameter set to another

Controller parameters

Name	Unit	Value range	Description
bEnablePreController	none	[TRUE,FALSE]	Switches pre-controller on.
bEnableZones	none	[TRUE,FALSE]	Switches open loop characteristic on until close to setpoint.
bEnableCVFilter	none	[TRUE,FALSE]	Switches on control value filter following the main controller.
iFilterType	none	ENUM	Selection of a filter type for the control value filter following the main controller (see below) [▶ 29].
iControllerType	none	ENUM	Selection of a control algorithm (see below) [▶ 30]

Alarming parameters

Name	Unit	Value range	Description
TempLow	K	(L)REAL	Relative lower temperature limit in the first band.
TempLowLow	K	(L)REAL	Relative lower temperature limit in the second band.
TempHigh	K	(L)REAL	Relative upper temperature limit in the first band.
TempHighHigh	K	(L)REAL	Relative upper temperature limit in the second band.
TempAbsoluteHigh	K	(L)REAL	Absolute upper temperature limit.
TempAbsoluteLow	K	(L)REAL	Absolute lower temperature limit.

Expert parameters

Name	Unit	Value range	Description
fTuneKp	none	(L)REAL	FineTuning parameters for the PID controller (only for advanced users)
fTuneTn	none	(L)REAL	FineTuning parameters for the PID controller (only for advanced users)
fTuneTv	none	(L)REAL	FineTuning parameters for the PID controller (only for advanced users)
fTuneTd	none	(L)REAL	FineTuning parameters for the PID controller (only for advanced users)

Description

ST_CTRL_ParaController

```
TYPE ST_CTRL_ParaController :
STRUCT
    (* Controller parameter set - heating *)
    KpHeat : FLOAT;
    TnHeat : TIME;
    TvHeat : TIME;
    TdHeat : TIME;
    (* Controller parameter set - cooling *)
    KpCool : FLOAT;
    TnCool : TIME;
    TvCool : TIME;
    TdCool : TIME;
END_STRUCT
END_TYPE
```

Description

Name	Unit	Value range	Description
KpHeat	none	(L)REAL	Amplification factor for the main controller.
TnHeat	s	TIME	Integral action time for main controller (I component).
TvHeat	s	TIME	Derivative action time for main controller (D component).
TdHeat	s	TIME	Damping time for the main controller.
KpCool	none	(L)REAL	Amplification factor for the main controller.
TnCool	s	TIME	Integral action time for main controller (I component).
TvCool	s	TIME	Derivative action time for main controller (D component).
TdCool	s	TIME	Damping time for the main controller.

E_CTRL_ERRORCODES:

```
TYPE E_CTRL_ERRORCODES :
(
    eCTRL_ERROR_NOERROR                     := 0, (* no error *)
    eCTRL_ERROR_INVALIDTASKCYCLETIME          := 1, (* invalid task cycle time *)
    eCTRL_ERROR_INVALIDCTRLCYCLETIME          := 2, (* invalid ctrl cycle time *)
    eCTRL_ERROR_INVALIDPARAM                := 3, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Tv             := 4, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Td             := 5, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Tn             := 6, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Ti             := 7, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fHystereisisRange          := 8, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fPosOutOn_Off      := 9, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fNegOutOn_Off      := 10, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_TableDescription          := 11, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_TableData              := 12, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_DataTableADR       := 13, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_T0                 := 14, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_T1                 := 15, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_T2               := 16, (* invalid parameter *)
```

```
    eCTRL_ERROR_INVALIDPARAM_T3                 := 17, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Theta              := 18, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nOrder                 := 19, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Tt                 := 20, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Tu                 := 21, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Tg                 := 22, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_infinite_slope         := 23, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fMaxIsLessThanfMin             := 24, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fOutMaxLimitIsLessThanfOutMinLimit     := 25, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fOuterWindow           := 26, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fInnerWindow           := 27, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fOuterWindowIsLessThanfInnerWindow     := 28, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fDeadBandInput             := 29, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fDeadBandOutput            := 30, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_PWM_Cycletime          := 31, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_no_Parameterset            := 32, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fOutOn                 := 33, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fOutOff                := 34, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fGain              := 35, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fOffset                := 36, (* invalid parameter *)
    eCTRL_ERROR_MODE_NOT_SUPPORTED              := 37, (* invalid mode: mode not supported *)
    eCTRL_ERROR_INVALIDPARAM_Tv_heating                 := 38, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Td_heating                 := 39, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Tn_heating                 := 40, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Tv_cooling                 := 41, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Td_cooling                 := 42, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_Tn_cooling                 := 43, (* invalid parameter *)
    eCTRL_ERROR_RANGE_NOT_SUPPORTED             := 44, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nParameterChangeCycleTicks         := 45, (* invalid parameter *)
    eCTRL_ERROR_ParameterEstimationFailed           := 46, (* invalid parameter *)
    eCTRL_ERROR_NoiseLevelToHigh                := 47, (* invalid parameter *)
    eCTRL_ERROR_INTERNAL_ERROR_0                := 48, (* internal error *)
    eCTRL_ERROR_INTERNAL_ERROR_1                := 49, (* internal error *)
    eCTRL_ERROR_INTERNAL_ERROR_2                := 50, (* internal error *)
    eCTRL_ERROR_INTERNAL_ERROR_3                := 51, (* internal error *)
    eCTRL_ERROR_INTERNAL_ERROR_4                := 52, (* internal error *)
    eCTRL_ERROR_INTERNAL_ERROR_5                := 53, (* internal error *)
    eCTRL_ERROR_INTERNAL_ERROR_6                := 54, (* internal error *)
    eCTRL_ERROR_INTERNAL_ERROR_7                := 55, (* internal error *)
    eCTRL_ERROR_INTERNAL_ERROR_8                := 56, (* internal error *)
    eCTRL_ERROR_INTERNAL_ERROR_9                := 57, (* internal error *)
    eCTRL_ERROR_INVALIDPARAM_WorkArrayADR           := 58, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_tOnTiime                   := 59, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_tOffTiime                  := 60, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nMaxMovingPulses               := 61, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nAdditionalPulsesAtLimits          := 62, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fCtrlOutMax_Min            := 63, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fDeltaMax                  := 64, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_tMovingTime            := 65, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_tDeadTime                  := 66, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_tAdditionalMoveTimeAtLimits        := 67, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fThreshold                 := 68, (* invalid parameter *)
    eCTRL_ERROR_MEMCPY                      := 69, (* MEMCPY failed *)
    eCTRL_ERROR_MEMSET                      := 70, (* MEMSET failed *)
    eCTRL_ERROR_INVALIDPARAM_nNumberOfColumns               := 71, (* invalid parameter *)
    eCTRL_ERROR_FileClose                   := 72, (* File Close failed *)
    eCTRL_ERROR_FileOpen                    := 73, (* File Open failed *)
    eCTRL_ERROR_FileWrite                   := 74, (* File Write failed *)
    eCTRL_ERROR_INVALIDPARAM_fVeloNeg                   := 75, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fVeloPos                   := 76, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_DeadBandInput          := 77, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_DeadBandOutput         := 78, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_CycleDuration          := 79, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_tStart                 := 80, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_StepHeigthTuningToLow      := 81, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fMinLimitIsLessThanZero        := 82, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fMaxLimitIsGreaterThan100          := 83, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fStepSize                  := 84, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fOkRangeIsLessOrEqualZero          := 85, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fForceRangeIsLessOrEqualfOkRange       := 86, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPWMPeriod                := 87, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_tMinimumPulseTime              := 88, (* invalid parameter *)
    eCTRL_ERROR_FileDelete                  := 89, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nNumberOfPwmOutputs            := 90, (* File Delete failed *)
    eCTRL_ERROR_INVALIDPARAM_nPwmInputArray_SIZEOF      := 91, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nPwmOutputArray_SIZEOF         := 92, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nPwmWaitTimesConfig_SIZEOF         := 93, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nPwmInternalData_SIZEOF        := 94, (* invalid parameter *)
    eCTRL_ERROR_SIZEOF                      := 95, (* SIZEOF failed *)
    eCTRL_ERROR_INVALIDPARAM_nOrderOfTheTransferfunction        := 96, (* invalid parameter *)
```

```
    eCTRL_ERROR_INVALIDPARAM_nNumeratorArray_SIZEOF         := 97, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nDenominatorArray_SIZEOF           := 98, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_a_n_IsZero                 := 99, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_WorkArraySIZEOF            := 100, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_MOVINGRANGE_MIN_MAX            := 101, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_MOVINGTIME                 := 102, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_DEADTIME                   := 103, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fMinLimitIsGreaterThanfMaxLimit    := 104, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_DataTableSIZEOF            := 105, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_OutputVectorDescription        := 106, (* invalid parameter *)
    eCTRL_ERROR_TaskCycleTimeChanged                := 107, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nMinMovingPulses               := 108, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fAcceleration          := 109, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fDeceleration          := 110, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_StartAndTargetPos              := 111, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fVelocity                  := 112, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fTargetPos                 := 113, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fStartPos                  := 114, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fMovingLength          := 115, (* invalid parameter *)
    eCTRL_ERROR_NT_GetTime                  := 116, (* internal error NT_GetTime *)
    eCTRL_ERROR_INVALIDPARAM_No3PhaseSolutionPossible           := 117, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fStartVelo                 := 118, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fTargetVelo            := 119, (* invalid parameter *)
    eCTRL_ERROR_INVALID_NEW_PARAMETER_TYPE          := 120  (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fBaseTime                  := 121, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nOrderOfTheTransferfunction_SIZEOF     := 122, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nFilterOrder           := 124, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nCoefficientsArray_a_SIZEOF        := 125, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nCoefficientsArray_b_SIZEOF        := 126, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nDigitalFiterData_SIZEOF           := 127, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nLogBuffer_SIZEOF              := 128, (* invalid parameter *)
    eCTRL_ERROR_LogBufferOverflow               := 129, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nLogBuffer_ADR             := 130, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nCoefficientsArray_a_ADR           := 131, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nCoefficientsArray_b_ADR           := 132, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nPwmInputArray_ADR             := 133, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nPwmOutputArray_ADR            := 134, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nPwmWaitTimesConfig_ADR        := 135, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nPwmInternalData_ADR       := 136, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nDigitalFiterData_ADR      := 137, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nNumeratorArray_ADR            := 138, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nDenominatorArray_ADR      := 139, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nTransferfunction1Data_ADR         := 140, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_nTransferfunction2Data_ADR         := 141, (* invalid parameter *)
    eCTRL_ERROR_FileSeek                    := 142, (* internal error FB_FileSeek *)
    eCTRL_ERROR_INVALIDPARAM_AmbientTempMaxIsLessThanAmbientTempMin := 143, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_ForerunTempMaxIsLessThanForerunTempMin := 144, (* invalid parameter *)
    eCTRL_ERROR_INVALIDLOGCYCLETIME                 := 145, (* invalid parameter *)
    eCTRL_ERROR_INVALIDVERSION_TcControllerToolbox      := 146,
    eCTRL_ERROR_INVALIDPARAM_Bandwidth                  := 147, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_NotchFrequency             := 148, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_DampingCoefficient             := 149, (* invalid parameter *)
    eCTRL_ERROR_INVALIDPARAM_fKpIsLessThanZero              := 150  (* invalid parameter *)
);
END_TYPE
```

ENUM: E_CTRL_ReactionOnFailure

ENUM: E_CTRL_ReactionOnFailure

Name	Description
eCTRL_ReactionOnFailure_NoFailure	No error..
eCTRL_ReactionOnFailure_StopController	If there is an error (an alarm) the controller will stop.
eCTRL_ReactionOnFailure_SetManMode	If there is an error (an alarm) the controller will switch to manual operation.
eCTRL_ReactionOnFailure_SetYMax	If there is an error (an alarm) set the control value to its maximum.
eCTRL_ReactionOnFailure_SetYMin	If there is an error (an alarm) set the control value to its minimum.
eCTRL_ReactionOnFailure_SetYMean	If there is an error (an alarm) set the control value to the average control value (not yet implemented)

ENUM: E_CTRL_ControllerStateInternal

ENUM: E_CTRL_ControllerStateInternal

Name	Description
E_CTRL_ControllerStateInternalHeating	internal
E_CTRL_ControllerStateInternalCooling	internal

ENUM: E_CTRL_ControlMode

Name	Description
eCTRL_ControlMode_HEATING	Heating only.
eCTRL_ControlMode_COOLING	Cooling only.
eCTRL_ControlMode_HEATING_COOLING	Heating and cooling.

E_CTRL_STATE :

```
TYPE E_CTRL_STATE :
(
    eCTRL_STATE_IDLE        := 0, (* state idle *)
    eCTRL_STATE_PASSIVE     := 1, (* state passive *)
    eCTRL_STATE_ACTIVE      := 2, (* state active *)
    eCTRL_STATE_RESET           := 3, (* state reset *)
    eCTRL_STATE_MANUAL      := 4, (* state manual *)
    eCTRL_STATE_TUNING      := 5, (* state tuning *)
    eCTRL_STATE_TUNED           := 6, (* state tuning ready - tuned *)
    eCTRL_STATE_SELFTEST    := 7, (* state selftest *)
    eCTRL_STATE_ERROR           := 8, (* state error *)
    eCTRL_STATE_SYNC_MOVEMENT       := 9  (* state synchronizing movement *)
);
END_TYPE
```

ENUM: E_CTRL_STATE_TUNING

ENUM: E_CTRL_STATE_TUNING

Name	Description
eCTRL_STATE_TUNING_INIT	Tuning: Initialisation
eCTRL_STATE_TUNING_IDLE	Tuning: wainting for a stable actual value
eCTRL_STATE_TUNING_PULSE	Tuning: not yet realized
eCTRL_STATE_TUNING_STEP	Tuning: Tuning with step reponse
eCTRL_STATE_TUNING_READY	Tuning: Calculation of parameters
eCTRL_STATE_TUNING_ERROR	Tuning: Error while tuning.

ENUM: E_CTRL_TuneMode

ENUM: E_CTRL_TuneMode

Name	Description
eCTRL_TuneMode_HEATING	Tuning: only heating
eCTRL_TuneMode_COOLING	Tuning: only cooling
eCTRL_TuneMode_HEATING_COOLING	Tuning: first heating, then cooling
eCTRL_TuneMode_COOLING_HEATING	Tuning: first cooling, then heating
eCTRL_TuneMode_OSCILLATION	Tuning: on-the-fly tuning with a defined oscillation (in planning)

ENUM: E_CTRL_FilterType

ENUM: E_CTRL_FilterType

Name	Description
eCTRL_FilterType_FIRSTORDER	first order filter
eCTRL_FilterType_AVERAGE	moving average filter

ENUM: E_CTRL_ControllerType

Name	Description
eCTRL_ControllerType_PID	Standard PID control algorithm.
eCTRL_ControllerType_PI	Standard PI control algorithm.
eCTRL_ControllerType_PID_Pre	Standard PID control algorithm with pre-controller (in preparation).
eCTRL_ControllerType_PIDD2	Serial PID control algorithm (in preparation)

Bit-masks for alarms

Name	Mask	Description
nAlarmOpenThermocouple	2#0000_0000_0000_0000_0000_0000_0000_0001	Hardware: open temperature sensor
nAlarmReverseThermocouple	2#0000_0000_0000_0000_0000_0000_0000_0010	Hardware: reverse connected temperature sensor
nAlarmBackVoltage	2#0000_0000_0000_0000_0000_0000_0000_0100	Hardware: excessive voltage at temperature sensor
nAlarmLeakageCurrent	2#0000_0000_0000_0000_0000_0000_0000_1000	Hardware: leakage current measured
nAlarmShortCircuit	2#0000_0000_0000_0000_0000_0000_0001_0000	Hardware: short circuit
nAlarmOpenCircuit	2#0000_0000_0000_0000_0000_0000_0010_0000	Hardware: no current
nAlarmLimitLow	2#0000_0000_0000_0000_0000_0001_0000_0000	Software: fallen below first lower relative temperature
nAlarmLimitLowLow	2#0000_0000_0000_0000_0000_0010_0000_0000	Software: fallen below second lower relative temperature
nAlarmLimitHigh	2#0000_0000_0000_0000_0000_0100_0000_0000	Software: first upper relative temperature exceeded
nAlarmLimitHighHigh	2#0000_0000_0000_0000_0000_1000_0000_0000	Software: second upper relative temperature exceeded
nAlarmAbsoluteHigh	2#0000_0000_0000_0000_0001_0000_0000_0000	Software: upper absolute temperature exceeded
nAlarmAbsoluteLow	2#0000_0000_0000_0000_0010_0000_0000_0000	Software: fallen below lower absolute temperature

11.3 Previous implementation

The following steps must be taken:

- **The controller library must be added to the project using the Library Manager.**

TcTempCtrl.lib is to be added in the Library Manager.

- **At least one instance of the controller must be programmed.**

An instance of the FB_TempController controller function block is to be created. It is also necessary for an instance of the ST_ControllerParameter structure to be created.

- **Perform the required external connection.**

Name		Description
bOn	Connection necessary	TRUE switches the controller on. Can be permanently set to TRUE if the controller is always to be switched on.
bInit	Connection necessary	Initialization flag, which must be active (TRUE) for precisely the first cycle in which the controller is called.
bTune	Connection necessary if tuning is to be used	A rising edge switches the self-tuning on. If it is switched to FALSE during the self-tuning process then the self-tuning is aborted and the controller continues operation using the old parameters (if they are still present).
bManual	Connection optional	TRUE switches manual operation on. If the signal goes to FALSE again, the controller returns to automatic mode.
bSelSetpoint	Connection optional	Selects one of two possible setpoints. FALSE selects the normal setpoint, while TRUE selects the standby setpoint.
bSelCtrlParameterSet	Connection optional	Selects one of two parameter sets. FALSE causes the internal (determined) parameter set to be used, while TRUE switches to one provided externally.
bEnableSoftStart	Connection optional	The soft start-up process is used if TRUE.
bEnableRamping	Connection optional	TRUE causes each setpoint step-change to be converted to a ramp.
fW1	Connection necessary	Setpoint.
fW2	Connection optional	Standby setpoint, generally lower than fW1. fSelSetpoint can be used to switch between fW1 and fW2.
fX	Connection necessary	Actual value. This value must be converted to LREAL.
bOpenThermocouple	Connection optional	The thermocouple is open if TRUE. Must be reported by the hardware (e.g. KLxxxx).
bReverseThermocouple	Connection optional	TRUE reports that the thermocouple has been connected with the wrong polarity. Must be reported by the hardware.
bBackVoltage	Connection optional	TRUE indicates that the input voltage at the thermocouple is too high. Must be reported by the hardware.
bLeakage	Connection optional	TRUE indicates that leakage current has been detected at the heating element. Must be reported by the hardware.
bShortCircuit	Connection optional	TRUE indicates that a short circuit has been detected at the heating element. Must be reported by the hardware.
bOpenCircuit	Connection optional	TRUE indicates that an open circuit has been detected at the heating element. Must be reported by the hardware.
sControllerParameter	Connection necessary	General parameters (sampling time etc.) are passed to the function block in this structure.
sParaControllerExternal	Connection optional	An external controller parameter set is passed to the function block in this structure.
sLogData	Connection optional	This structure passes parameters for logging to the function block (filenames etc.).

- **Perform the necessary parameterization of the controller via the structure.**

The parameters can be specified through initial values, or by assignment. If initial values are used, then the instance of the structure with initial values looks like this:

```
sControllerParameter : ST_ControllerParameter :=
(
(*********************************************************************)
(* general parameters *)
iMode := CTRLMODE_HEATING, (* 1=heating, 2=cooling, 3=heating&cooling *)
iReactionOnFailure := TC_OnFailureStopController, (* controller off or manual op or yMin or yMax *)
fYTune := 100, (* step change while tuning operation *)
fYStable := 0.0, (* tuning operation *)
dwAlarmSupp := 16#ff_ff_ff_ff, (* alarm suppression *)
tCtrlCycleTime := t#100ms, (* controller cycle time *)
tTaskCycleTime := t#10ms, (* plc task cycle time *)
```

```
(*********************************************************************)
(* setpoint parameters *)
fWMin := 0.0, (* lower limit *)
fWMax := 200, (* upper limit *)

(* start up optional *)
fWStartUp := 0.0, (* soft start plateau setpoint *)
tStartUp := t#0s, (* soft start waiting time*)
bStartUpRamping := FALSE, (* enable ramping while start up phase *)
fWStartUpVeloPos := 0.0, (* max gradient for increasing setpoint in start up phase*)
fWStartUpVeloNeg := 0.0, (* max gradient for decreasing setpoint in start up phase *)

fWVeloPos := 0.0, (* max gradient for increasing setpoint *)
fWVeloNeg := 0.0, (* max gradient for decreasing setpoint *)
(*********************************************************************)
(* actual value parameters *)
bFilter := FALSE,
tFilter := t#0s,
(*********************************************************************)
(* control value parameters *)
fYMin := 0, (* lower limit *)
fYMax := 100, (* upper limit *)
fYManual := 0.0, (* manual operation*)
fYOnFailure := 0.0, (* control value on failure *)
tPWMCycleTime := t#100ms, (* PWM *)
(*********************************************************************)
(* controller settings *)
bEnablePreController := FALSE, (* enable precontroller *)
bEnableZones := FALSE, (* enable zone around setpoint with open loop control *)
bEnableCVFilter := FALSE, (* enable filter for CV (type see iFilterType) *)
iFilterType := E_FilterType_FIRSTORDER, (* filtertype of CV filter *)
iControllerType := E_ControllerType_PID, (* used controller normally PID *)
(*********************************************************************)
(* min max temperatures *)
TempLow := 10,
TempLowLow := 20,
TempHigh := 10,
TempHighHigh := 20,
TempAbsoluteHigh := 200,
TempAbsoluteLow := 0,
(*********************************************************************)
(* internal tuning parameters *)
fTuneKp := 1.2,
fTuneTn := 2.0,
fTuneTv := 0.42,
fTuneTd := 0.25
);
```

The marked parameters are optional, and only have to be initialized if they are needed.

Assignment in the code may look as follows in ST:

```
sControllerParameter.tPWMCycleTime :=
t#100ms;
```

- **Specification of the controller sampling time, the task cycle time and the PWM cycle time**

The controller's sampling time must be adapted to the section. It should be selected to be equal to or less than one tenth of the section's dominant time constants. The task cycle time is specified by the PLC task from which the controller function block has been called. This value can be read from the task configuration (PLC Control: Resources Task Configuration). The PWM cycle time is usually equal to the controller cycle time. If the task cycle time is 10 ms and the chosen PWM cycle time (=controller sampling time) is 100 ms, then a total of 10 levels (PWM cycle time / task cycle time) are available.

- **Parameterization of TwinCAT Scope**

To check the results, a scope recording should always be made of the tuning process and the closed loop control behavior. To do this, TwinCAT Scope View should be started and parameterized. The following channels should be recorded: setpoint (fW1 or fW2), actual value (fX) and the analog control value (fYAnalog).

- **Switching off the alarms during the commissioning phase**

The alarms can be temporarily switched off during the commissioning phase. An appropriate bit mask must be written into the dwAlarmSupp Dword. If a bit is set in this Dword, the corresponding alarm is disabled. The assignment of the individual alarms is described here [▶ 41].

All required alarms should be switched on again after initial commissioning!

- **Starting the controller with tuning**

If the controller parameters are to be determined with the aid of the tuning system, the bOn and bTune inputs must both be TRUE. Firstly, a fixed waiting time of 20 s elapses. During this waiting time, the temperature of the section is monitored to ensure that it remains within a +-1°C band. If the temperature goes outside this band, the waiting time starts again. The section is then subjected to a step excitation with a control value of fYTune. The section then reacts with the step response. As long as 80% of the setpoint has not been reached, the section parameters are determined using the inflectional tangent method. For safety reasons, control is switched over to closed control loop once 80% of the setpoint has been reached. If the temperature reaches the 80% mark too quickly (with no clear inflection) then the value of fYTune is to be reduced. The determined parameters are used for the PID controller, and are provided in a structure at the output of the controller.

The section must perform a step of at least 40°C for the purposes of tuning. Smaller steps can result in incorrect parameter determination!

- **Linking the internal control parameters with the external connections**

The controller parameters determined in the tuning process can be supplied to the controller again as external parameters. This may be necessary if the tuning is only to be carried out once (e.g. only during the initial commissioning). To do this, the sParaControllerInternal structure is fed back to the controller's sParaControllerExternal input, and the bSelCtrlParameterSet flag set to TRUE.

- **Fine tuning**

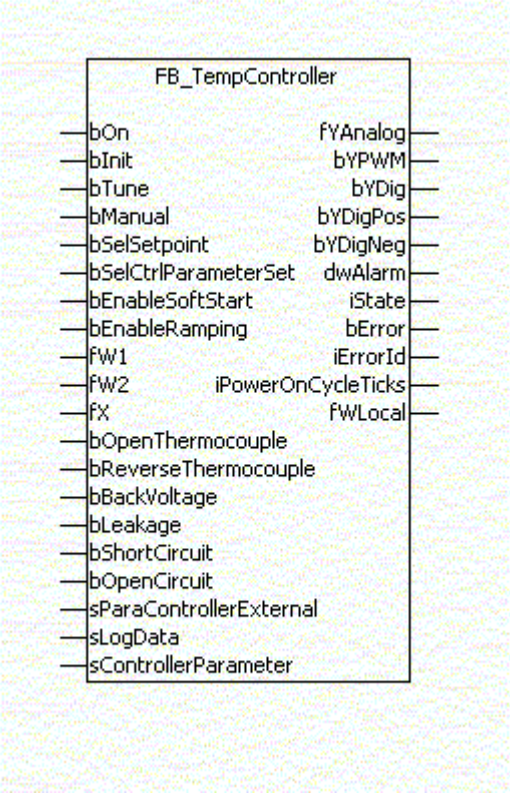
The control parameters determined in the tuning process are designed to produce fast settling, with about 10% overshoot. If only very little overshoot is permitted, or even none at all, then the following parameters from the ST_ControllerParameter structure can be used to perform fine tuning. These values should be considered only as a guide.

Behavior	fTuneKp	fTuneTn	fTuneTv	fTuneTd
Fast settling with an overshoot of 10% – 20%	1.2	2.0	0.42	0.25
Slower settling with less overshoot	1.0	2.5	0.42	0.25
Almost asymptotic settling with extremely low overshoot	0.5	3.0	1.0	0.25

11.3.1 FB_TempController

The temperature controller function block FB_TempController has a variety of inputs and outputs that are described below. All the controller's parameters are passed to it via structures. The structures and enums are defined here [▶ 37].

Function block



Interface

```
VAR_INPUT
    bOn : BOOL; (* start closed loop operating *)
    bInit : BOOL; (* init controller *)
    bTune : BOOL; (* start self tuning *)
    bManual : BOOL; (* manual operation *)
    bSelSetpoint : BOOL; (* FALSE = setpoint 1, TRUE = setpoint 2
*)
    bSelCtrlParameterSet : BOOL; (* FALSE = internal set, TRUE =
external set *)
    bEnableSoftStart : BOOL; (* FALSE = no soft start, TRUE = soft
start *)
    bEnableRamping : BOOL; (* FALSE = no ramping, TRUE = ramping
*)
    fW1 : LREAL; (* first setpoint *)
    fW2 : LREAL; (* second setpoint *)
    fX : LREAL; (* actual value *)
    (* alarming inputs *)
    bOpenThermocouple : BOOL; (* thermocouple *)
    bReverseThermocouple : BOOL;
    bBackVoltage : BOOL;
    bLeakage : BOOL; (* heating system *)
    bShortCircuit : BOOL;
    bOpenCircuit : BOOL;
    sParaControllerExternal : ST_ParaController; (* external
controller parameter set *)
    (* logging *)
    sLogData : ST_LogData := (bLog := FALSE, strLogFileName :=
'', strLogString := '' );
END_VAR
VAR_IN_OUT
    sControllerParameter : ST_ControllerParameter; (* parameters
*)
END_VAR
VAR_OUTPUT
    fYAnalog : LREAL; (* control value analog *)
    bYPWM : BOOL; (* control value PWM *)
    bYDig : BOOL; (* 2-Point *)
    bYDigPos : BOOL; (* 3-Point: control value digital positive
*)
    bYDigNeg : BOOL; (* 3-Point: control value digital
negative*)
    (* alarming *)
```

```
    dwAlarm : DWORD; (* max. 32 alarms *)
    (* state *)
    iState : States := TC_STATE_IDLE;
    (* controller parameter *)
    sParaControllerInternal : ST_ParaController;
    (* general errors *)
    bError : BOOL;
    iErrorId : ErrorCodes;
END_VAR
```

Inputs

Name	Unit	Value range	Description
bOn	1	[TRUE,FALSE]	TRUE switches the controller on.
bInit	1	[TRUE,FALSE]	Initialization flag, which must be active (TRUE) for precisely the first cycle in which the controller is called.
bTune	1	[TRUE,FALSE]	A rising edge switches the self-tuning on. If it is switched to FALSE during the self-tuning process then the self-tuning is aborted and the controller continues operation using the old parameters (if they are still present).
bManual	1	[TRUE,FALSE]	TRUE switches manual operation on. If the signal goes FALSE again, the controller returns to automatic mode.
bSelSetpoint	1	[TRUE,FALSE]	Selects one of two possible setpoints. FALSE selects the normal setpoint, while TRUE selects the standby setpoint.
bSelCtrlParameterSet	1	[TRUE,FALSE]	Selects one of two parameter sets. FALSE causes the internal (determined) parameter set to be used, while TRUE switches to one provided externally.
bEnableSoftStart	1	[TRUE,FALSE]	The soft start-up process is used if TRUE.
bEnableRamping	1	[TRUE,FALSE]	TRUE causes each setpoint step-change to be converted to a ramp.
fW1	°C	LREAL	Setpoint.
fW2	°C	LREAL	Standby setpoint, generally smaller than fW1. fSelSetpoint can be used to switch between fW1 and fW2.
fX	°C	LREAL	Actual value. This value must be converted to LREAL.

Name	Unit	Value range	Description
bOpenThermocouple	1	[TRUE,FALSE]	The thermocouple is open if TRUE. Must be indicated by the hardware (e.g. KLxxxx).
bReverseThermocouple	1	[TRUE,FALSE]	TRUE indicates that the thermocouple has been connected with the wrong polarity. Must be indicated by the hardware.
bBackVoltage	1	[TRUE,FALSE]	TRUE indicates that the input voltage at the thermocouple is too high. Must be indicated by the hardware.
bLeakage	1	[TRUE,FALSE]	TRUE indicates that leakage current has been detected at the heating element. Must be indicated by the hardware.
bShortCircuit	1	[TRUE,FALSE]	TRUE indicates that a short circuit has been detected at the heating element. Must be indicated by the hardware.
bOpenCircuit	1	[TRUE,FALSE]	TRUE indicates that an open circuit has been detected at the heating element. Must be indicated by the hardware.
sControllerParameter	none	Structure	General parameters (sampling time etc.) are passed to the function block in this structure.
sParaControllerExternal	none	Structure	An external controller parameter set is passed to the function block in this structure.
sLogData	none	Structure	This structure passes parameters for logging to the function block (filenames etc.).

Outputs

Name	Unit	Value range	Description
fYAnalog	none	LREAL	Analog control value.
bYPWM	none	[TRUE,FALSE]	Boolean output, pulse width modulated.
bYDig	none	[TRUE,FALSE]	Boolean output of an on-off controller (TRUE control value 100%, FALSE control value off)
bYDigPos	none	[TRUE,FALSE]	Boolean output of a three-step controller (TRUE control value 100%, FALSE control value off)

Name	Unit	Value range	Description
bYDigNeg	none	[TRUE,FALSE]	Boolean output of a three-step controller (TRUE control value -100%, FALSE control value off)
dwAlarm	none	DWORD	Alarm messages (see ENUM ...)
iState	none	INT	Current controller state (see ENUM ...)
sParaControllerInternal	none	Structure	In this structure the internal controller parameter set (determined by the tuning) is made available.
bError	none	[TRUE,FALSE]	If an error is present, then bError is TRUE.
iErrorId	none	INT	If bError is TRUE, then iErrorId provides an error code (see ENUM ...)

11.3.2 Structure Definitions

ST_ControllerParameter

```
TYPE ST_ParaControlValue :
STRUCT

(*************************************************************************)

    (* general parameters *)
    iMode : E_ControlMode; (* 1=heating, 2=cooling,
3=heating&cooling *)
    iReactionOnFailure : E_ReactionOnFailure; (* 0=controller off,
1=manual op, 2=yMin, 3=yMax *)
    fYTune : LREAL; (* step change while tuning operation *)
    fYStable : LREAL; (* tuning operation *)
    dwAlarmSupp : DWORD; (* alarm suppression *)
    tCtrlCycleTime : TIME; (* controller cycle time *)
    tTaskCycleTime : TIME; (* plc task cycle time *)

(*************************************************************************)

    (* setpoint parameters *)
    fWMin : LREAL; (* lower limit *)
    fWMax : LREAL; (* upper limit *)
    (* start up *)
    fWStartUp : LREAL; (* soft start plateau setpoint *)
    tStartUp : TIME; (* soft start waiting time*)
    bStartUpRamping : BOOL; (* enable ramping while start up phase
*)
    fWStartUpVeloPos : LREAL; (* max gradient for increasing
setpoint in start up phase*)
    fWStartUpVeloNeg : LREAL; (* max gradient for decreasing
setpoint in start up phase *)
    fWVeloPos : LREAL; (* max gradient for increasing setpoint
*)
    fWVeloNeg : LREAL; (* max gradient for decreasing setpoint
*)

(*************************************************************************)

    (* actual value parameters *)
    bFilter : BOOL;
    tFilter : TIME;

(*************************************************************************)

    (* control value parameters *)
    fYMin : LREAL; (* lower limit *)
```

```
    fYMax : LREAL; (* upper limit *)
    fYManual : LREAL; (* manual operation*)
    fYOnFailure : LREAL; (* control value on failure *)
    tPWMCycleTime : TIME; (* PWM *)

(*************************************************************************)

    (* controller settings *)
    bEnablePreController : BOOL; (* enable precontroller *)
    bEnableZones : BOOL; (* enable zone around setpoint with open
loop control *)
    bEnableCVFilter : BOOL; (* enable filter for CV (type see
iFilterType) *)
    iFilterType : E_FilterType; (* filtertype of CV filter *)
    iControllerType : E_ControllerType; (* used controller normally
PID *)

(*************************************************************************)

    (* min max temperatures *)
    TempLow : LREAL;
    TempLowLow : LREAL;
    TempHigh : LREAL;
    TempHighHigh : LREAL;
    TempAbsoluteHigh : LREAL;
    TempAbsoluteLow : LREAL;

(*************************************************************************)

    (* internal tuning parameters *)
    fTuneKp : LREAL := 1.2;
    fTuneTn : LREAL := 2.0;
    fTuneTv : LREAL := 0.42;
    fTuneTd : LREAL := 0.25;
END_STRUCT
END_TYPE
```

Description

Name	Unit	Value range	Description
iMode	none	INT	Controller operation mode (1 = heating, 2 = cooling, 3 = heating & cooling) (see below) [▶ 40]
iReactionOnFailure	none	INT	Parameterizable reaction to errors (see below) [▶ 40]
fYTune	none	LREAL	Control value during the self-tuning (normally 100%)
fYStable	none	LREAL	Control value during the settling phase (normally 0%)
dwAlarmSupp	none	DWORD	Masks out the alarms (see below) [▶ 41]
tCtrlCycleTime	s	TIME	Controller's sampling time. In the course of the sampling time the controller re-calculates the control value.
tTaskCycleTime	s	TIME	Task cycle time. The FB is called with this time interval.
fWMin	K	LREAL	Minimum setpoint.
fWMax	K	LREAL	Maximum setpoint.
fWVeloPos	K/s	LREAL	Rate of rise (of ramp).
fWVeloNeg	K/s	LREAL	Rate of fall (of ramp).
fWStartUp	K	LREAL	Setpoint at start-up.
tStartUp	s	TIME	Time with the fWStartUp setpoint.
bStartUpRamping	none	[TRUE,FALSE]	Switches on ramping during the start-up phase.
fWStartUpVeloPos	K/s	LREAL	Rate of rise (of ramp) during the start-up phase.
fWStartUpVeloNeg	K/s	LREAL	Rate of fall (of ramp) during the start-up phase
fYMin	none	LREAL	Minimum value of the control value.
fYMax	none	LREAL	Maximum value of the control value.
fYManual	none	LREAL	Control value in manual operation.

Name	Unit	Value range	Description
fYOnFailure	none	LREAL	Control value in case of error (parameterizable).
tPWMCycleTime	s	TIME	Cycle time of the PWM signal.
tFilter	s	TIME	Time constant of the actual value filter (first order P-T1 filter)
bFilter	none	[TRUE,FALSE]	The actual value filter is actuated if TRUE.
bEnablePreController	none	[TRUE,FALSE]	Switches pre-controller on.
bEnableZones	none	[TRUE,FALSE]	Switches open loop characteristic on until close to setpoint.
bEnableCVFilter	none	[TRUE,FALSE]	Switches on control value filter following the main controller.
iFilterType	none	ENUM	Selection of a filter type for the control value filter following the main controller (see below) [▶ 41].
iControllerType	none	ENUM	Selection of a control algorithm (see below) [▶ 41]
TempLow	K	LREAL	Relative lower temperature limit in the first band.
TempLowLow	K	LREAL	Relative lower temperature limit in the second band.
TempHigh	K	LREAL	Relative upper temperature limit in the first band.
TempHighHigh	K	LREAL	Relative upper temperature limit in the second band.
TempAbsoluteHigh	K	LREAL	Absolute upper temperature limit.
TempAbsoluteLow	K	LREAL	Absolute lower temperature limit.
fTuneKp	none	LREAL	FineTuning parameters for the PID controller (only for advanced users)
fTuneTn	none	LREAL	FineTuning parameters for the PID controller (only for advanced users)
fTuneTv	none	LREAL	FineTuning parameters for the PID controller (only for advanced users)
fTuneTd	none	LREAL	FineTuning parameters for the PID controller (only for advanced users)

Description

ST_ParaController

```
TYPE ST_ParaController :
STRUCT
    (* Main Controller parameter set *)
    KpMain : LREAL;
    TnMain : LREAL;
    TvMain : LREAL;
    TdMain : LREAL;
    (* Pre Controller parameter set *)
    KpPre : LREAL;
    TvPre : LREAL;
    TdPre : LREAL;
END_STRUCT
END_TYPE
```

Description

Name	Unit	Value range	Description
KpMain	none	LREAL	Amplification factor for the main controller.
TnMain	s	TIME	Integral action time for main controller (I component).
TvMain	s	TIME	Derivative action time for main controller (D component).
TdMain	s	TIME	Damping time for the main controller.

Name	Unit	Value range	Description
KpPre	none	LREAL	Amplification factor for the pre-controller.
TvPre	s	TIME	Derivative action time for pre-controller (D component).
TdPre	s	TIME	Damping time for the pre-controller.

ENUM: error codes

Name	Description
TC_ERR_NOERROR	No error.
TC_ERR_INVALIDPARAM	Invalid parameter.
TC_ERR_NO_INIT	Missing function block initialization.
TC_ERR_NO_INFLECTION_POINT	No inflection was found during self-tuning. No parameters could be determined.
TC_ERR_INVALID_PARAM	Invalid parameter.
TC_ERR_INVALID_CYCLETIME	Invalid combination of cycle times (sampling times and PWM cycle times).
TC_ERR_WRONG_TU	A valid value for the Tu parameter could not be found due to faulty or aborted self-tuning.

ENUM: ReactionOnFailure

Name	Description
TC_OnFailureNoFailure	No error.
TC_OnFailureStopController	If there is an error (an alarm) the controller will stop.
TC_OnFailureSetManMode	If there is an error (an alarm) the controller will switch to manual operation.
TC_OnFailureSetYMax	If there is an error (an alarm) set the control value to its maximum.
TC_OnFailureSetYMin	If there is an error (an alarm) set the control value to its minimum.

ENUM: ST_ControlMode

Name	Description
CTRLMODE_HEATING	Heating only.
CTRLMODE_COOLING	Cooling only.
CTRLMODE_HEATING_COOLING	Heating and cooling.

ENUM: states

Name	Description
TC_STATE_IDLE	Controller switched off.
TC_STATE_INIT	Controller is being initialized.
TC_STATE_OFF	Controller switched off, was previously switched on.
TC_STATE_TUNE	Controller in tuning / self-tuning state.
TC_STATE_MANUAL_OPERATION	Controller in manual operation.
TC_STATE_CLOSED_LOOP	Controller in automatic operation.
TC_STATE_TUNE_IDLE	Tuning started but not yet running. Waiting for idle.
TC_STATE_TUNE_PULSE	Pulse for determination of dead time.
TC_STATE_TUNE_STEP	Step for determination of dead time and maximum velocity.

Name	Description
TC_STATE_TUNE_READY	Self-tuning complete.
TC_STATE_ERROR	Error (logical error).

ENUM: E_FilterType

Name	Description
E_FilterType_FIRSTORDER	First order filter.
E_FilterType_AVERAGE	Mean value filter.

ENUM: E_ControllerType

Name	Description
E_ControllerType_PID	Standard PID control algorithm.
E_ControllerType_PIDD2	Planned serial PID control algorithm.

Bit-masks for alarms

Name	Mask	Description
nAlarmOpenThermocouple	2#0000_0000_0000_0000_0000_0000_0000_0001	Hardware: open temperature sensor
nAlarmReverseThermocouple	2#0000_0000_0000_0000_0000_0000_0000_0010	Hardware: reverse connected temperature sensor
nAlarmBackVoltage	2#0000_0000_0000_0000_0000_0000_0000_0100	Hardware: excessive voltage at temperature sensor
nAlarmLeakageCurrent	2#0000_0000_0000_0000_0000_0000_0000_1000	Hardware: leakage current measured
nAlarmShortCircuit	2#0000_0000_0000_0000_0000_0000_0001_0000	Hardware: short circuit
nAlarmOpenCircuit	2#0000_0000_0000_0000_0000_0000_0010_0000	Hardware: no current
nAlarmLimitLow	2#0000_0000_0000_0000_0000_0001_0000_0000	Software: fallen below first lower relative temperature
nAlarmLimitLowLow	2#0000_0000_0000_0000_0000_0010_0000_0000	Software: fallen below second lower relative temperature
nAlarmLimitHigh	2#0000_0000_0000_0000_0000_0100_0000_0000	Software: first upper relative temperature exceeded
nAlarmLimitHighHigh	2#0000_0000_0000_0000_0000_1000_0000_0000	Software: second upper relative temperature exceeded
nAlarmAbsoluteHigh	2#0000_0000_0000_0000_0001_0000_0000_0000	Software: upper absolute temperature exceeded
nAlarmAbsoluteLow	2#0000_0000_0000_0000_0010_0000_0000_0000	Software: fallen below lower absolute temperature

More Information:
www.beckhoff.com/ts4110

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com