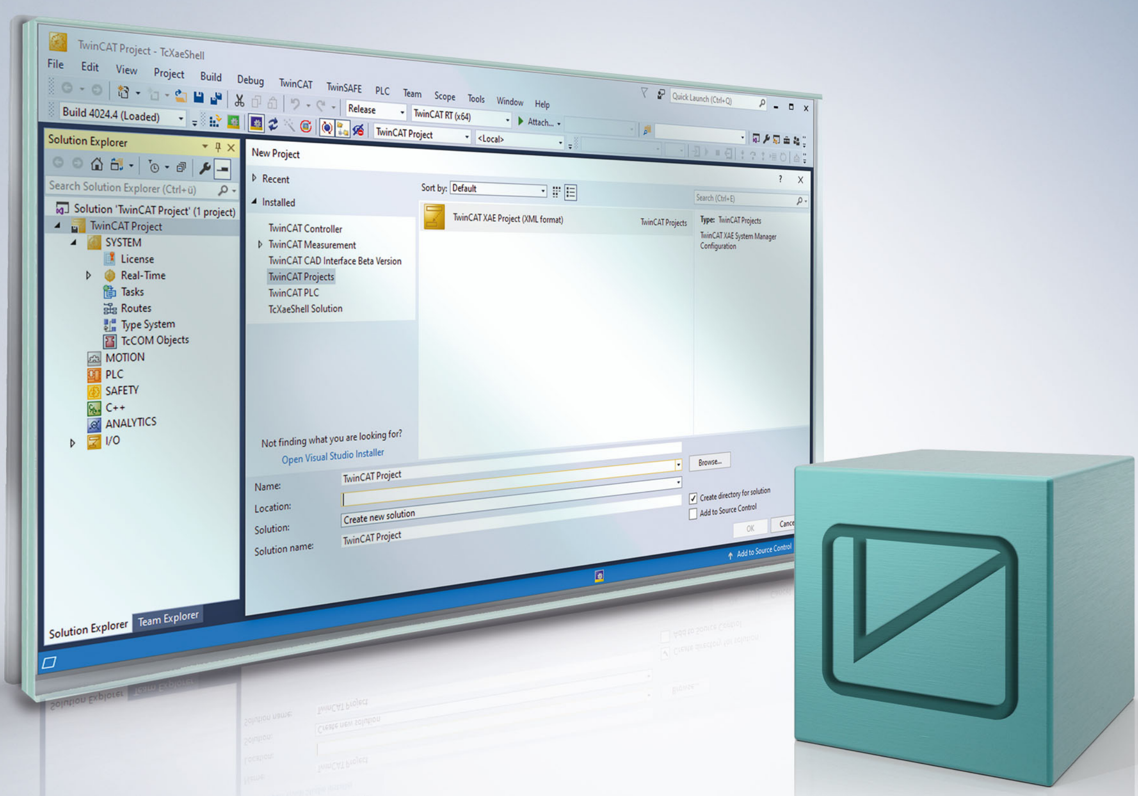
BECKHOFF New Automation Technology

Manual | EN

TF4100

TwinCAT 3 | Controller Toolbox



2025-02-03 | Version: 2.2.4

Table of contents

1 Foreword

1.1 Notes on the documentation

This description is intended exclusively for trained specialists in control and automation technology who are familiar with the applicable national standards.
The documentation and the following notes and explanations must be complied with when installing and commissioning the components.
The trained specialists must always use the current valid documentation.

The trained specialists must ensure that the application and use of the products described is in line with all safety requirements, including all relevant laws, regulations, guidelines, and standards.

Disclaimer

The documentation has been compiled 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 notice.
Claims to modify products that have already been supplied may not 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 and licensed trademarks of Beckhoff Automation GmbH.
If third parties make use of the designations or trademarks contained in this publication for their own purposes, this could infringe upon the rights of the owners of the said designations.

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

Copyright

Third-party trademarks

Trademarks of third parties may be used in this documentation. You can find the trademark notices here: https://www.beckhoff.com/trademarks.

1.2 For your safety

Safety regulations

Read the following explanations for your safety.
Always observe and follow product-specific safety instructions, which you may find at the appropriate places in this document.

Exclusion of liability

All the components are supplied in particular hardware and software configurations which are 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 technology who are familiar with the applicable national standards.

Signal words

The signal words used in the documentation are classified below. In order to prevent injury and damage to persons and property, read and follow the safety and warning notices.

Personal injury warnings

⚠ DANGER
Hazard with high risk of death or serious injury.

⚠ WARNING
Hazard with medium risk of death or serious injury.

⚠ CAUTION
There is a low-risk hazard that could result in medium or minor injury.

Warning of damage to property or environment

NOTICE
The environment, equipment, or data may be damaged.

Information on handling the product

This information includes, for example:
recommendations for action, assistance or further information on the product.

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.

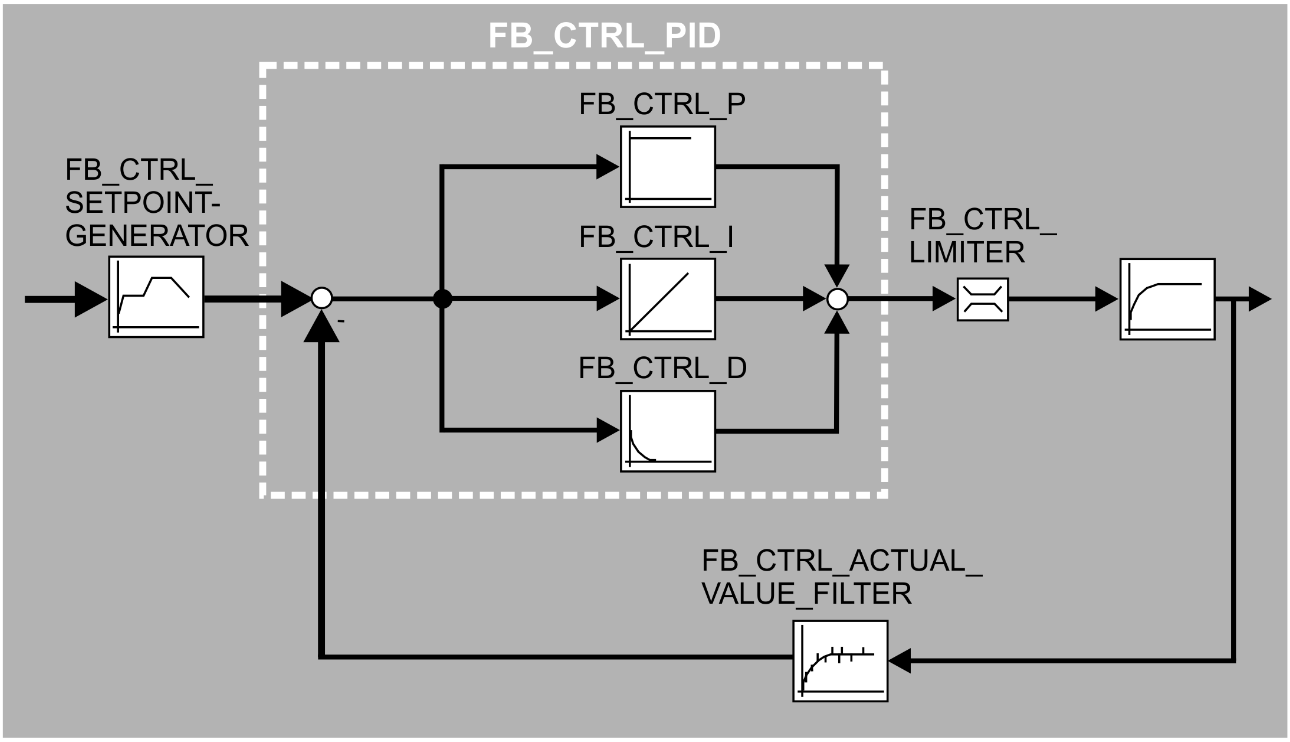
In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our https://www.beckhoff.com/secguide.

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

This library contains function blocks that represent various control engineering transfer elements in a functional diagram. Complex controllers that can be used for a large number of applications are included, as well as basic function blocks with which unique controller structures can be implemented for special applications.



Function blocks

Name	Description
FB_CTRL_2POINT [▸ 41]	2-position controller
FB_CTRL_2POINT_PWM_ADAPTIVE [▸ 43]	Adaptive 2-position controller with PWM output
FB_CTRL_3PHASE_SETPOINT_GENERATOR [▸ 153]	3 phase setpoint generator
FB_CTRL_3POINT [▸ 45]	3-position controller
FB_CTRL_3POINT_EXT [▸ 47]	Extended 3-position controller
FB_CTRL_ACTUAL_VALUE_FILTER [▸ 85]	Actual value filter
FB_CTRL_ARITHMETIC_MEAN [▸ 87]	Arithmetic mean value filter
FB_CTRL_CHECK_IF_IN_BAND [▸ 123]	Area monitoring
FB_CTRL_D [▸ 33]	D element
FB_CTRL_DEADBAND [▸ 132]	Dead band
FB_CTRL_DIGITAL_FILTER [▸ 89]	Digital filter
FB_CTRL_FLOW_TEMP_SETPOINT_GEN [▸ 162]	Specification of the flow temperature depending on the outdoor temperature
FB_CTRL_GET_SYSTEM_TIME [▸ 20]	Output of the Windows system time
FB_CTRL_GET_TASK_CYCLETIME [▸ 22]	Determination of the task cycle time
FB_CTRL_HYSTERESIS [▸ 26]	Hysteresis element
FB_CTRL_I [▸ 29]	I element
FB_CTRL_I_WITH_DRIFTCOMPENSATION [▸ 31]	I element with drift compensation
FB_CTRL_LEAD_LAG [▸ 94]	Lead/lag element
FB_CTRL_LIMITER [▸ 133]	Control value limiter
FB_CTRL_LIN_INTERPOLATION [▸ 118]	Linear interpolation element
FB_CTRL_LOG_DATA [▸ 124]	Data logger in *.csv ASCII format
FB_CTRL_LOG_MAT_FILE [▸ 127]	Data logger in Matlab 5 format
FB_CTRL_LOOP_SCHEDULER [▸ 23]	Distribution of computing power in situations with several control loops
FB_CTRL_MOVING_AVERAGE [▸ 93]	Moving average filter
FB_CTRL_MULTIPLE_PWM_OUT [▸ 135]	PWM element with multiple outputs
FB_CTRL_NORMALIZE [▸ 120]	Characteristic curve linearization
FB_CTRL_NOISE_GENERATOR [▸ 97]	Noise generator
FB_CTRL_NOTCH_FILTER [▸ 98]	Notch filter
FB_CTRL_nPOINT [▸ 50]	n-position controller
FB_CTRL_P [▸ 28]	P element
FB_CTRL_PARAMETER_SWITCH [▸ 52]	Parameter switching algorithm for a split range controller
FB_CTRL_PI [▸ 55]	PI controller
FB_CTRL_PI_PID [▸ 57]	Cascaded PI-PID controller
FB_CTRL_PID [▸ 60]	PID controller
FB_CTRL_PID_EXT [▸ 72]	Extended PID controller
FB_CTRL_PID_EXT_SPLITRANGE [▸ 65]	Extended PID regulator with parameter set switchover
FB_CTRL_PID_SPLITRANGE [▸ 77]	PID regulator with parameter set switchover
FB_CTRL_PT1 [▸ 100]	PT_1 element
FB_CTRL_PT2 [▸ 102]	PT_2 element
FB_CTRL_PT2oscillation [▸ 104]	Oscillating PT_2 element
FB_CTRL_PT3 [▸ 106]	PT_3 element

Name	Description
FB_CTRL_PTn [▸ 108]	PT_n element
FB_CTRL_PTt [▸ 110]	PT_t element
FB_CTRL_PWM_OUT [▸ 140]	PWM element
FB_CTRL_PWM_OUT_EXT [▸ 141]	Extended PWM element
FB_CTRL_RAMP_GENERATOR [▸ 164]	Ramp generator
FB_CTRL_RAMP_GENERATOR_EXT [▸ 166]	Extended ramp generator
FB_CTRL_SCALE [▸ 144]	Range adjustment
FB_CTRL_SERVO_MOTOR_OUT [▸ 145]	Actuator control
FB_CTRL_SERVO_MOTOR_SIMULATION [▸ 112]	Actuator simulation
FB_CTRL_SETPOINT_GENERATOR [▸ 168]	Setpoint generator
FB_CTRL_SIGNAL_GENERATOR [▸ 171]	Signal generator
FB_CTRL_SPLITRANGE [▸ 148]	Signal decomposition into a positive and negative part.
FB_CTRL_STEPPING_MOTOR_OUT [▸ 150]	Stepper motor control
FB_CTRL_TRANSFERFUNCTION_1 [▸ 35]	Transfer function according to the first standard form
FB_CTRL_TRANSFERFUNCTION_2 [▸ 38]	Transfer function according to the second standard form
FB_CTRL_TuTg [▸ 114]	TuTg element
FB_CTRL_ZERO_ZONE_DAMPING [▸ 116]	Zero damping

Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT 3.1.4016	PC or CX	Tc2_ControllerToolbox

3 Installation

3.1 System requirements

Description of minimum requirements needed for engineering and/or runtime systems.

Development environment

A pure development environment describes a computer on which PLC programs are developed but not executed. The following components must be installed on a development computer:

- TwinCAT 3 XAE (Engineering) build 4012 or higher
- TwinCAT 3 function TF4100 controller toolbox version 3.4.0.0 or higher
- Please note: A 7-day trial license can be used for the development environment, see Licensing [▸ 15].

Runtime environment

A runtime environment describes a computer on which PLC programs are executed. The following components must be installed on a runtime computer:

- TwinCAT3 XAR build 4012 or higher
- Licenses for TC1200 PLC and TF4100 controller toolbox
- Please note: A 7-day trial license key can be used for testing purposes, see Licensing [▸ 15].

Developer environment and runtime on one computer

If runtime and development environments are to run on the same computer (e.g. to test a PLC program before it is loaded on the target computer), the following requirements must be met:

- TwinCAT3 XAE (engineering installation) build 4012 or higher
- TwinCAT 3 function TF4100 controller toolbox version 3.4.0.0 or higher
- Please note: A 7-day trial license key can be used for testing purposes, see Licensing [▸ 15].

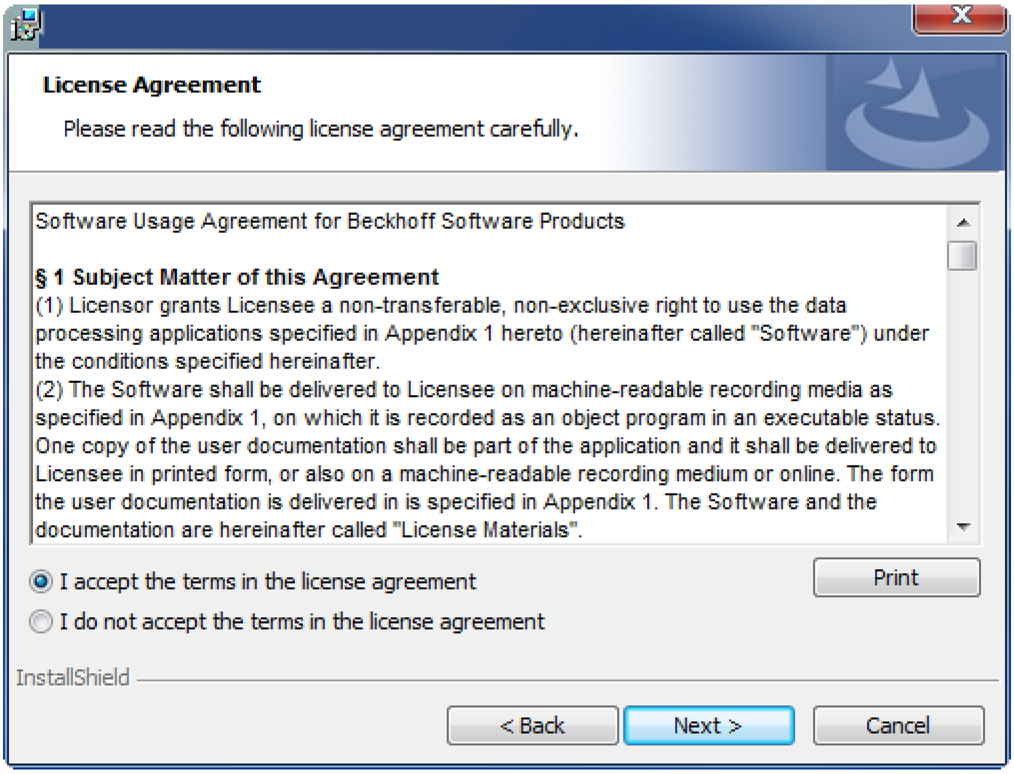
3.2 Installation

The following section describes how to install the TwinCAT 3 Function for Windows-based operating systems.

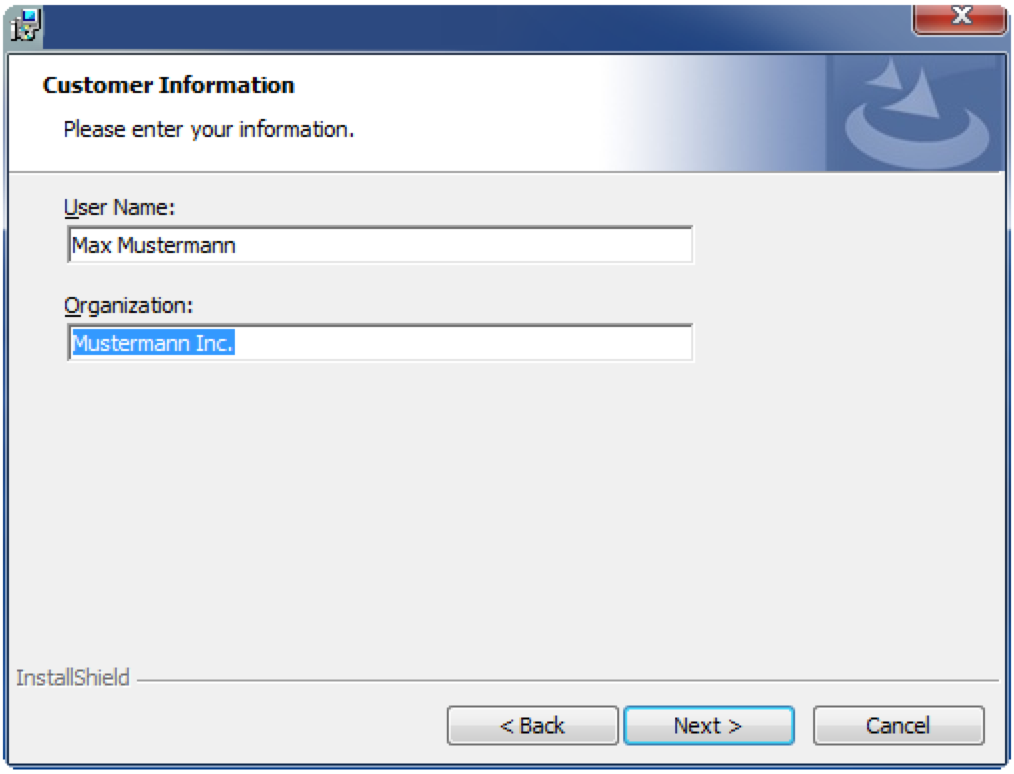
✓ The TwinCAT 3 Function setup file was downloaded from the Beckhoff website.

1. Run the setup file as administrator. To do this, select the command **Run as administrator** in the context menu of the file.
   ⇨ The installation dialog opens.

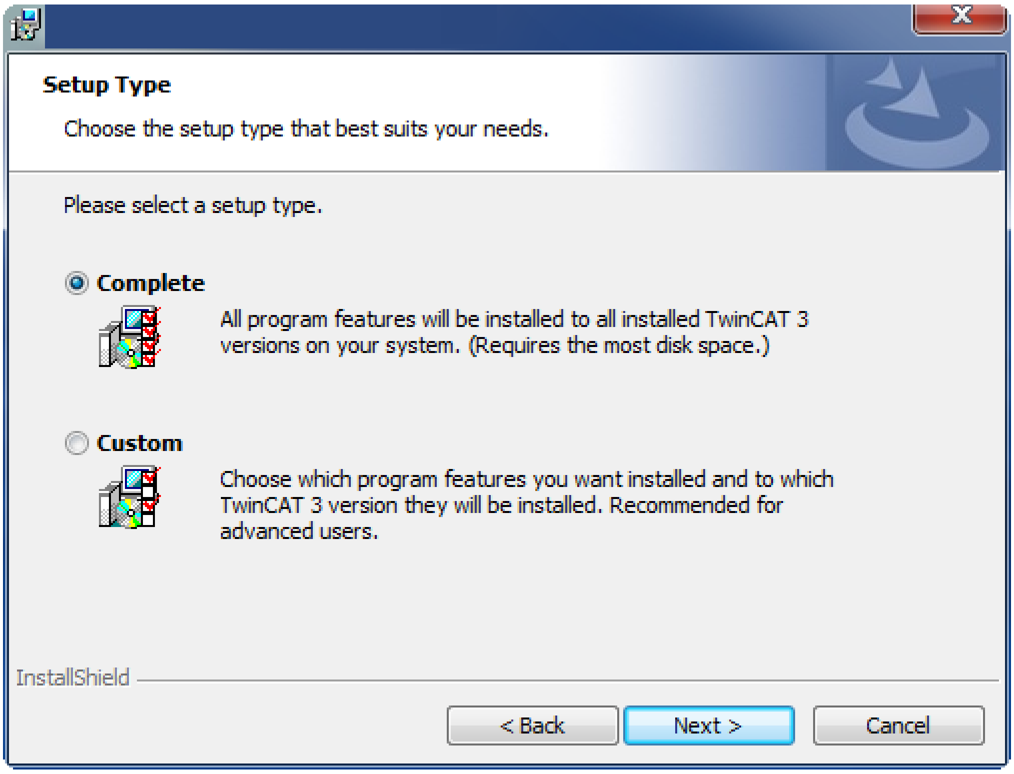
2. Accept the end user licensing agreement and click **Next**.



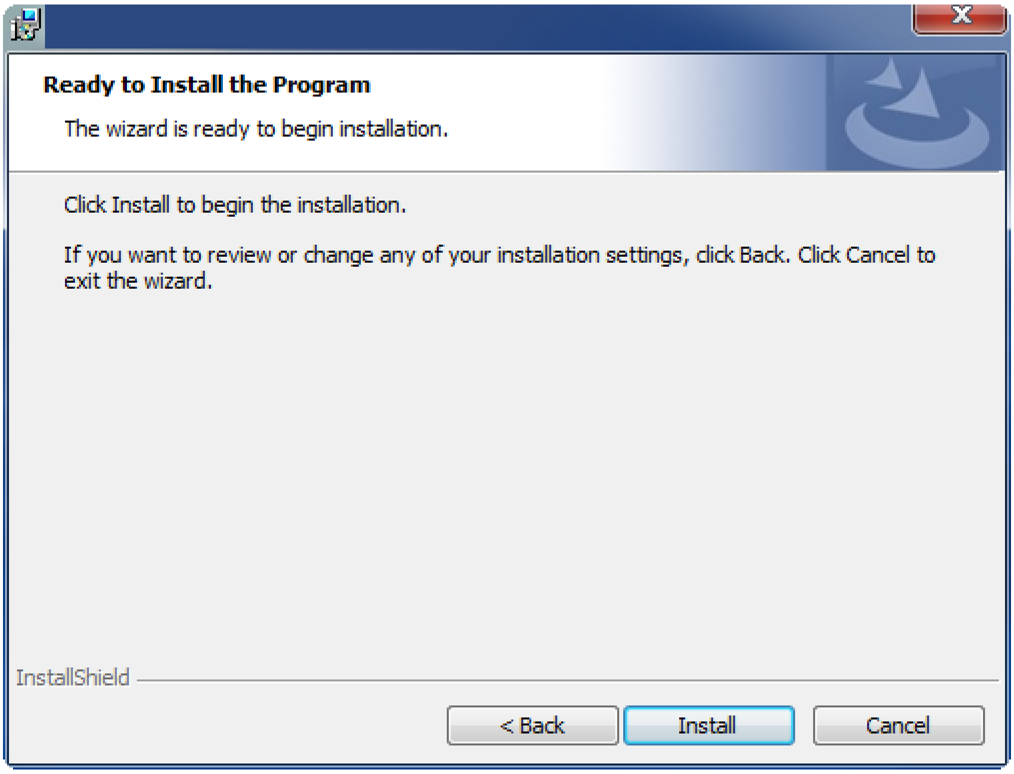
3. Enter your user data.



4. If you want to install the full version of the TwinCAT 3 Function, select **Complete** as installation type. If you want to install the TwinCAT 3 Function components separately, select **Custom**.

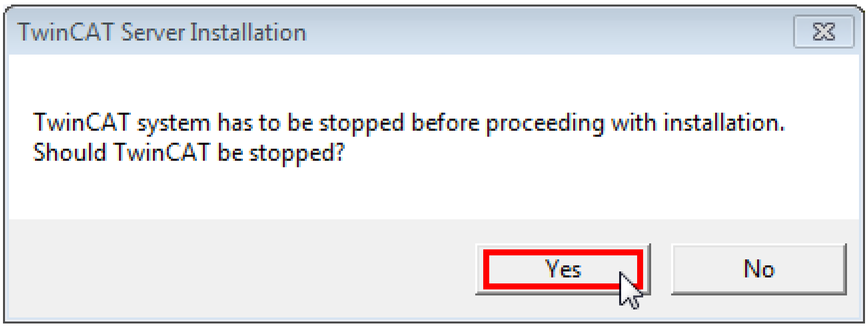


5. Select **Next**, then **Install** to start the installation.

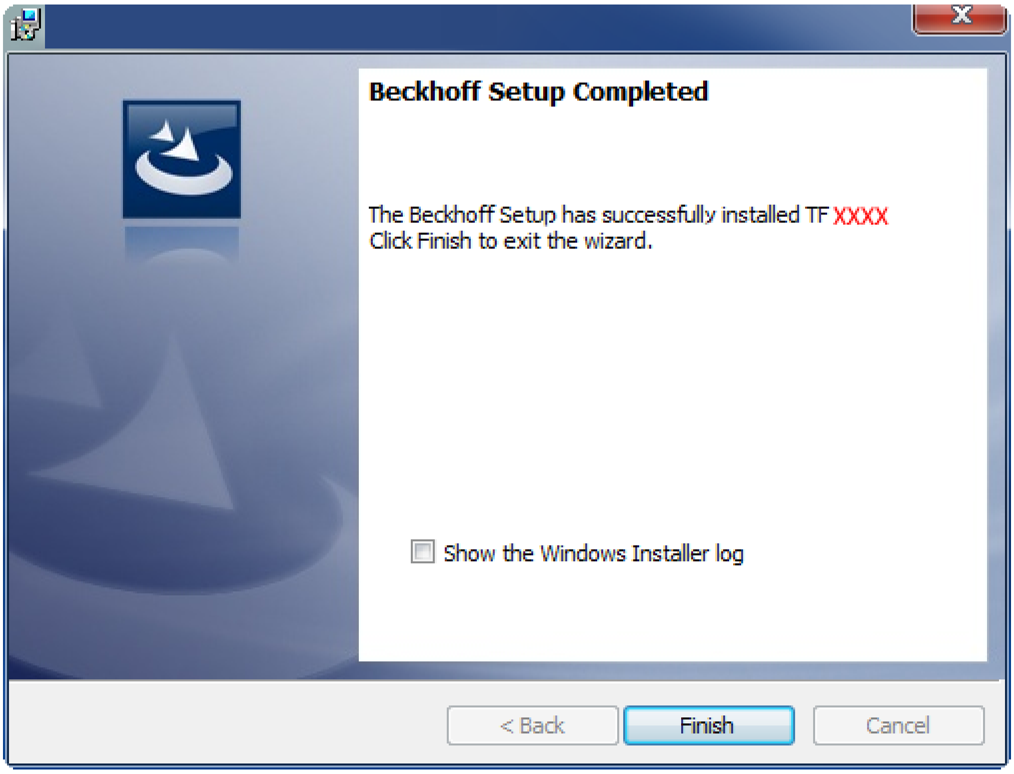


⇨ A dialog box informs you that the TwinCAT system must be stopped to proceed with the installation.

6. Confirm the dialog with **Yes**.



7. Select **Finish** to exit the setup.



⇨ The TwinCAT 3 Function has been successfully installed.

3.3 Licensing

The TwinCAT 3 function can be activated as a full version or as a 7-day test version. Both license types can be activated via the TwinCAT 3 development environment (XAE).

Licensing the full version of a TwinCAT 3 Function

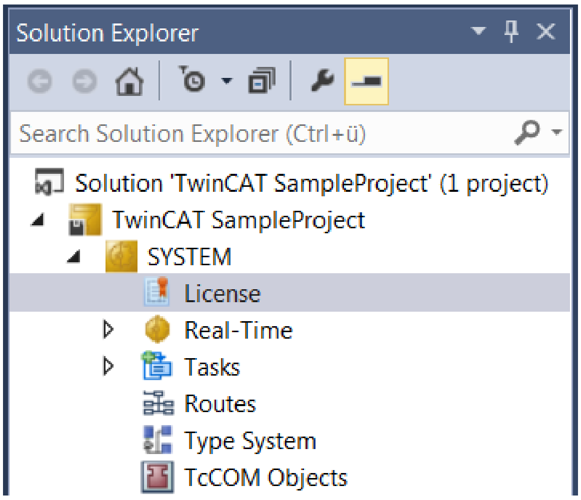
A description of the procedure to license a full version can be found in the Beckhoff Information System in the documentation "TwinCAT 3 Licensing".

Licensing the 7-day test version of a TwinCAT 3 Function

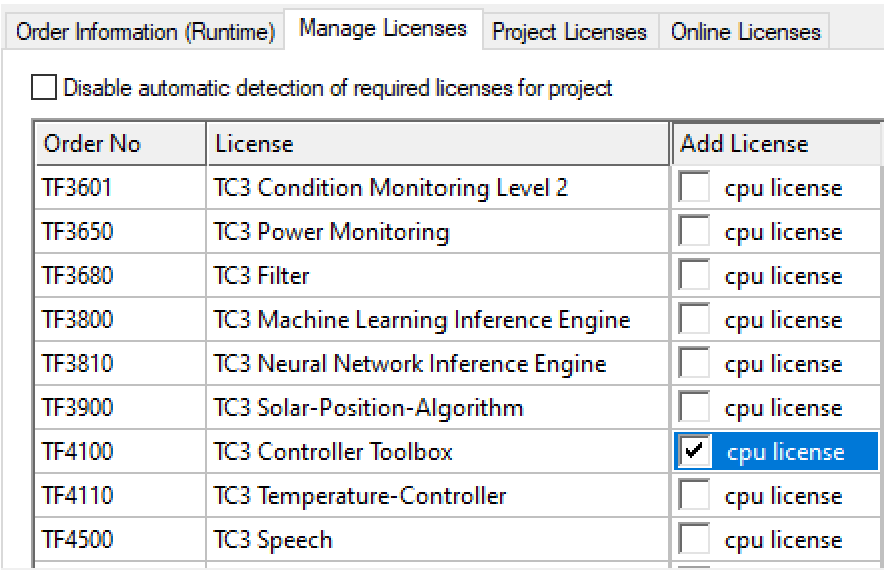
A 7-day test version cannot be enabled for a TwinCAT 3 license dongle.

1. Start the TwinCAT 3 development environment (XAE).
2. Open an existing TwinCAT 3 project or create a new project.

3. If you want to activate the license for a remote device, set the desired target system. To do this, select the target system from the **Choose Target System** drop-down list in the toolbar.
   ⇨ The licensing settings always refer to the selected target system. When the project is activated on the target system, the corresponding TwinCAT 3 licenses are automatically copied to this system.
4. In the **Solution Explorer**, double-click **License** in the **SYSTEM** subtree.

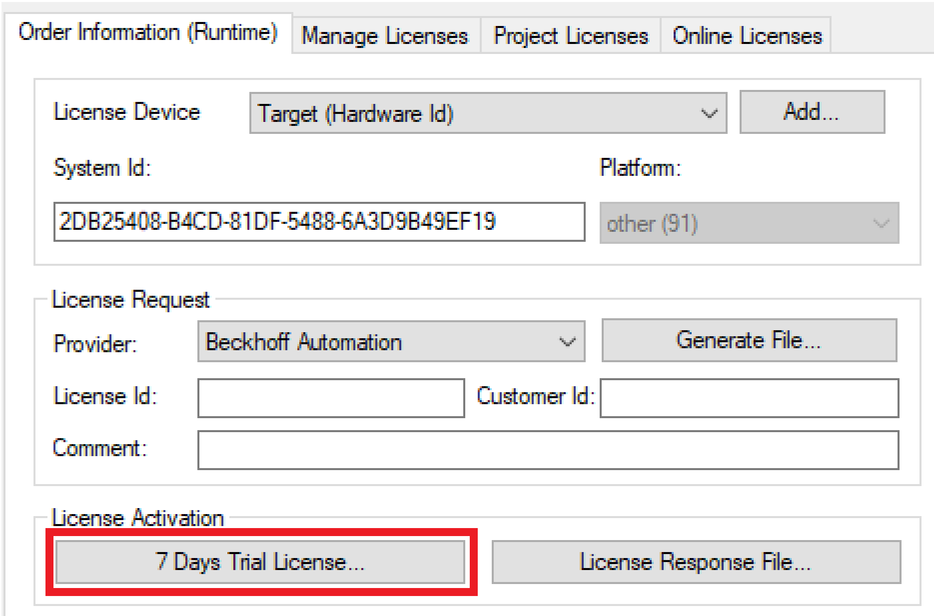


   ⇨ The TwinCAT 3 license manager opens.
5. Open the **Manage Licenses** tab. In the **Add License** column, check the check box for the license you want to add to your project (e.g. "TF4100 TC3 Controller Toolbox").

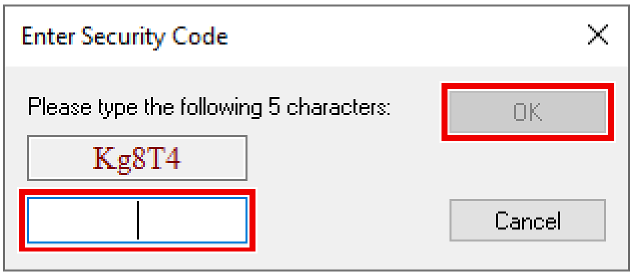


6. Open the **Order Information (Runtime)** tab.
   ⇨ In the tabular overview of licenses, the previously selected license is displayed with the status "missing"**.**

7. Click **7-Day Trial License...** to activate the 7-day trial license.



⇨ A dialog box opens, prompting you to enter the security code displayed in the dialog.



8. Enter the code exactly as it is displayed and confirm the entry.
9. Confirm the subsequent dialog, which indicates the successful activation.
   ⇨ In the tabular overview of licenses, the license status now indicates the expiry date of the license.
10. Restart the TwinCAT system.

⇨ The 7-day trial version is enabled.

4 PLC API

4.1 General operating principle

The following sections provide a generation description of the function blocks in the TwinCAT controller toolbox.

Discretization

The continuous transfer functions of the transfer elements assembled in this library are transformed to discrete values using the trapezoidal rule (Tustin formula).

The Tustin formula:

$$\frac{1}{s} = \frac{T}{2}\frac{z+1}{z-1}$$

Function block inputs

eMode

The operation mode of the majority of function blocks can be selected with this input. This makes it possible to select one of the following operation modes:

eCTRL_MODE_PASSIVE	The output or outputs of the function block are set to zero, but the internal states are retained.
eCTRL_MODE_ACTIVE	The function block is executed in accordance with its description, and appropriate output values are calculated (normal operation).
eCTRL_MODE_RESET	In this operation mode all internal states are reset and the error bit is cleared.
eCTRL_MODE_MANUAL	The value of the input value `fManSyncValue` is provided at the output (manual operation).

stParams

The necessary parameters are passed to the function block with this structure. The variables `tTaskCycleTime` and `tCtrlCycleTime` are contained in all the parameter structures. These parameters function in the following way:

The parameter `tTaskCycleTime` specifies the cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the cycle time of the calling task. If it is only called in every second cycle, the time must be doubled accordingly. The parameter `tCtrlCycleTime` indicates the control loop's sampling time. This time must be greater than or equal to the parameter `tTaskCycleTime`. If the sampling time is set equal to `tTaskCycleTime` then the function block is executed with every call. If the selected value is larger by a factor of 5, the function block is only processed in every fifth call. This makes it possible to implement slow control loops even in a fast task.

The parameters `tTaskCycleTime` and `tCtrlCycleTime` are of type TIME and therefore do not permit inputs of less than 1ms. In order to use the controller in a fast PLC task with a cycle time of less than 1ms, a global base time can be specified as reference for the specified cycle times. The use of the base time is explained in the following examples.

It is assumed that the function block is called in every task cycle.

Task: Configuration	Parameter: tTaskCycleTime	Parameter: tCtrlCycleTime	Action
T#10ms	T#10ms	T#10ms	The control loop is processed using a 10 ms sampling time.
T#10ms	T#10ms	T#50ms	The control loop is processed using a 50 ms sampling time.
T#100ms	T#100ms	T#100ms	The control loop is processed using a 100 ms sampling time.
T#100ms	T#100ms	T#50ms	**ERROR**, execution is not possible!
T#100ms	T#50ms	T#50ms	**ERROR**, although the function block has been executed, incorrect output values have been calculated!

The outputs of the function blocks

eState

This output indicates the current internal state of the function block.

eCTRL_STATE_IDLE	The function block has successfully been reset, and is now waiting for selection of the operation mode.
eCTRL_STATE_PASSIVE	The function block is in the passive state in which no calculations are carried out.
eCTRL_STATE_ACTIVE	The function block is in the active state, which is the normal operating state.
eCTRL_STATE_RESET	A reset request is being processed, but the reset has not yet been completed.
eCTRL_STATE_MANUAL	The function block is in the manual state, and the output value can be manually specified at the appropriate input.
eCTRL_STATE_...	If there are any other internal states, they are described together with the corresponding function blocks.
eCTRL_STATE_ERROR	An error has occurred; the function block is not executed when in this state. See **eErrorId** for further information.

bError

An error in the function block is indicated by a TRUE at this boolean output.

eErrorId

The error number [▶ 173] is provided at this output if the `bError` output is TRUE.

Using the global base time

In order to be able to use the function blocks of a PLC task with a cycle time of less than 1ms, it is possible to interpret the specified cycle times as ticks of a base time. In this special parameterization, the time unit of 1ms is interpreted as 1 tick. This approach is equivalent to setting a PLC cycle time of less than 1ms in the TwinCAT System Manager.

The switchover and declaration of the base time is done with the global structure `stCtrl_GLOBAL_CycleTimeInterpretation` for all function blocks of the toolbox.

```
VAR_GLOBAL
    stCtrl_GLOBAL_CycleTimeInterpretation :
    ST_CTRL_CYCLE_TIME_INTERPRETATION;
END_VAR

TYPE ST_CTRL_CYCLE_TIME_INTERPRETATION :
STRUCT
bInterpretCycleTimeAsTicks  : BOOL; (* e.g. 2ms -> 2ticks *)
fBaseTime                   : FLOAT; (* Base time in seconds, e.g. 200µs -> 200E-6s *)
END_STRUCT
END_TYPE
```

In order to interpret the specified cycle times as ticks, the variable `bInterpretCycleTimeAsTicks` in the global structure `stCtrl_GLOBAL_CycleTimeInterpretation` is set to TRUE. Within this structure, the base time unit has to be set in variable `fBaseTime`.

By setting the flag `bInterpretCycleTimeAsTicks`, the interpretation of the parameters with the names

- tTaskCycleTime
- tCtrlCycleTime

is changed. The interpretation and effect of all other parameters of type **TIME** remains unaffected.

Sample

The base time unit of the TwinCAT system is 200µs. The PLC task, and therefore the function blocks of the Toolbox, are called cyclically every 400µs.

Setting the global structure:

```
stCtrl_GLOBAL_CycleTimeInterpretation.bInterpretCycleTimeAsTicks := TRUE;
stCtrl_GLOBAL_CycleTimeInterpretation.fBaseTime := 200E-6;
```

Parameterization of a function block from the Toolbox:

```
stParams.tTaskCycleTime := T#2ms; (* 2*200µs=400µs *)
stParams.tCtrlCycleTime := T#4ms; (* 4*200µs=800µs *)
stParams. ...
```

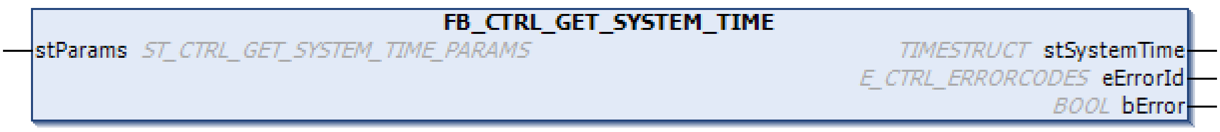
The **TaskCycleTime** specified on the function blocks is "2•200E-6s = 400µs" and thus corresponds to the set PLC cycle time. The **CtrlCycleTime** is set to "800µs = 4•200E-6s" so that the control loop operates with a sampling time of 800µs, i.e. it is processed in every second PLC cycle.

4.2 Reference

4.2.1 Function blocks

4.2.1.1 Auxiliary

4.2.1.1.1 FB_CTRL_GET_SYSTEM_TIME



The function block reads the current Windows system time and makes it available in the **SystemTimeStruct.**

Description

This function block makes the current system time available in its output structure. The resolution is specified through the `tCtrlCycleTime` parameter; the maximum resolution is 10 ms, and it is necessary to observe the condition "tCtrlCycleTime > 2 • tTaskCycleTime". Otherwise the resolution is reduced to "2 • tCtrlCycleTime".

Outputs

```
VAR_OUTPUT
    stSystemTime     : TIMESTRUCT;
    eErrorId         : E_CTRL_ERRORCODES;
    bError           : BOOL;
END_VAR
TYPE TIMESTRUCT
```

```
STRUCT
    wYear           : WORD;
    wMonth          : WORD;
    wDayOfWeek      : WORD;
    wDay            : WORD;
    wHour           : WORD;
    wMinute         : WORD;
    wSecond         : WORD;
    wMilliseconds   : WORD;
END_STRUCT
END_TYPE
```

Name	Type	Description
stSystemTime	TIME STRUCT	Structure in which the system time is output.
eErrorId	E_CTRL_ ERROR CODES	Supplies the error number [▶ 173] when the output bError is set.
bError	BOOL	Becomes TRUE as soon as an error situation occurs.
wYear	WORD	The year: 1970 ~ 2106;
wMonth	WORD	The month: 1 ~ 12 (January = 1, February = 2 etc.);
wDayOfWeek	WORD	The day of the week: 0 ~ 6 (Sunday = 0, Monday = 1 etc.);
wDay	WORD	The day of the month: 1 ~ 31;
wHour	WORD	Hour: 0 ~ 23;
wMinute	WORD	Minute: 0 ~ 59;
wSecond	WORD	Second: 0 ~ 59;
WMilliseconds	WORD	Millisecond: 0 ~ 999;

Inputs/ outputs

```
VAR_IN_OUT
    stParams     : ST_CTRL_GET_SYSTEM_TIME;
END_VAR
```

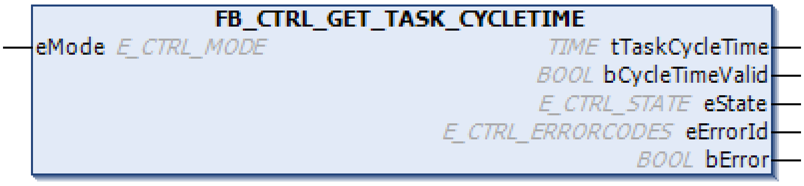
Name	Type	Description
stParams	ST_CTRL_ GET_SYSTEM_TI ME	Parameter structure of the function block

stParams consists of the following elements:

```
TYPE
ST_CTRL_GET_SYSTEM_TIME:
STRUCT
    tTaskCycleTime : TIME;
    tCtrlCycleTime : TIME;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every task cycle this corresponds to the task cycle time of the calling task.

4.2.1.1.2 FB_CTRL_GET_TASK_CYCLETIME



This function block allows the task cycle time of a program to be determined with a resolution of 1 ms.

Correct use of the function block

The **TaskCycleTime** can only be correctly determined if the program does not contain any active breakpoints.
The task cycle time is only determined once. No further measurements are taken if either of the `bCycleTimeValid` or `bError` outputs is TRUE.

Do not use this function block if you use cycle times that are less than 1 ms or not multiples of 1 ms

Inputs

```
VAR_INPUT
    eMode : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
eMode	E_CTRL_MODE	Input that specifies the operation mode [▸ 173] of the function block.

Outputs

```
VAR_OUTPUT
    tTaskCycleTime    : TIME;
    bCycleTimeValid   : BOOL;
    eState            : E_CTRL_STATE;
    eErrorId          : E_CTRL_ERRORCODES;
    bError            : BOOL;
END_VAR
```

Name	Type	Description
tTaskCycleTime	TIME	This output indicates the current task cycle time, with a resolution of 1 ms.
bCycleTimeValid	BOOL	The time contained in the `tTaskCycleTime` output is valid when this output is TRUE.
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERRORCODES	Supplies the error number [▸ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

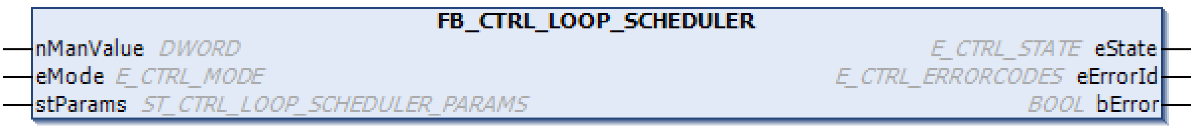
Sample:

```
PROGRAM PRG_GET_TASK_CYCLETIME_TEST
VAR
    tTaskCycleTime      : TIME;
    bCycleTimeValid     : BOOL;
    eState              : E_CTRL_STATE;
    eErrorId            : E_CTRL_ERRORCODES;
    bError              : BOOL;
    fbCTRL_GET_TASK_CYCLETIME : FB_CTRL_GET_TASK_CYCLETIME;
    bInit               : BOOL := TRUE;
    fSetpointValue      : FLOAT := 45.0;
    fActualValue        : FLOAT;
    fbCTRL_PI           : FB_CTRL_PI;
    stCTRL_PI_Params    : ST_CTRL_PI_PARAMS;
```

```
    fbCTRL_PT1          : FB_CTRL_PT1;
    stCTRL_PT1_Params   : ST_CTRL_PT1_PARAMS;
END_VAR
```

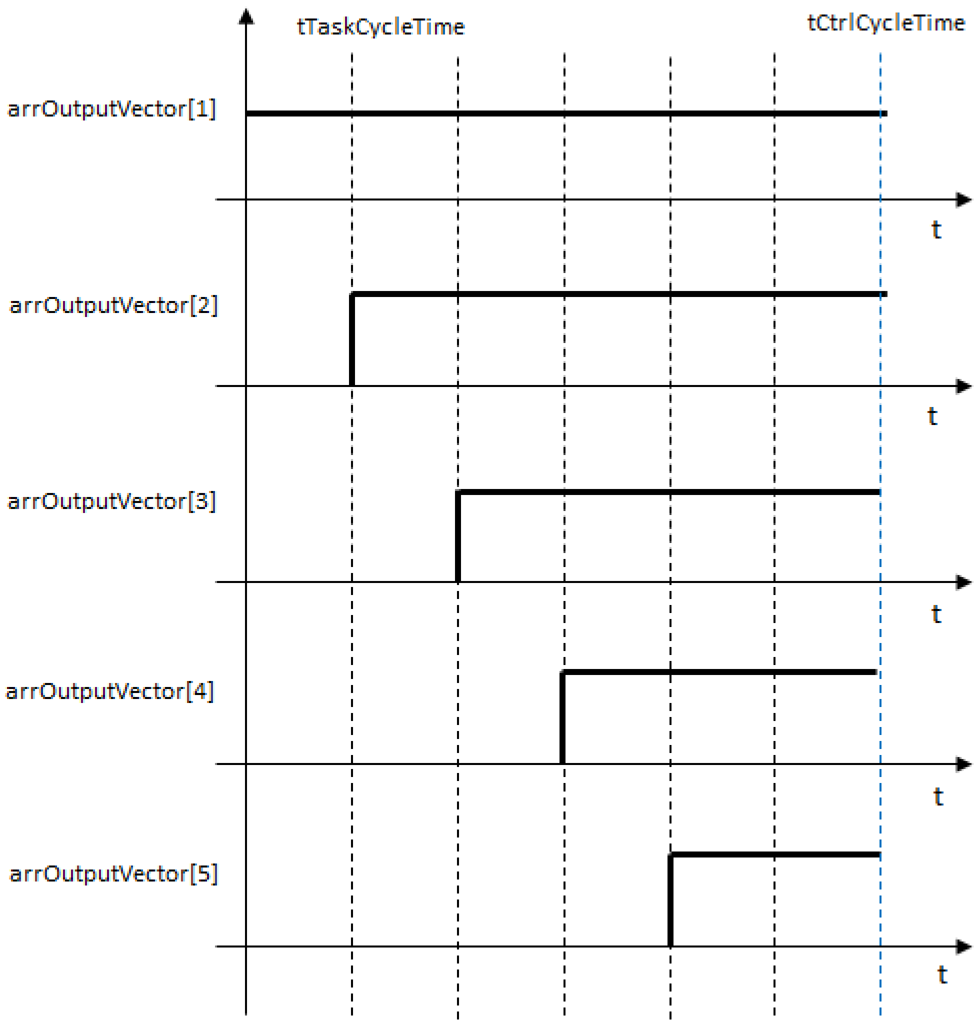
```
fbCTRL_GET_TASK_CYCLETIME( eMode    := eCTRL_MODE_ACTIVE,
                           tTaskCycleTime  => tTaskCycleTime,
                           bCycleTimeValid => bCycleTimeValid,
                           eState          => eCTRL_MODE_ACTIVE,
                           eErrorId        => eErrorId,
                           bError          => bError );
IF fbCTRL_GET_TASK_CYCLETIME.bCycleTimeValid
THEN
    IF bInit
    THEN
        stCTRL_PT1_Params.tTaskCycleTime := fbCTRL_GET_TASK_CYCLETIME.tTaskCycleTime;
        stCTRL_PT1_Params.tCtrlCycleTime := T#100ms;
        stCTRL_PT1_Params.fKp := 1.0;
        stCTRL_PT1_Params.tT1 := T#10s;
        stCTRL_PI_Params.tTaskCycleTime := fbCTRL_GET_TASK_CYCLETIME.tTaskCycleTime;
        stCTRL_PI_Params.tCtrlCycleTime := T#100ms;
        stCTRL_PI_Params.fKp := 0.5;
        stCTRL_PI_Params.tTn := T#5s;
        stCTRL_PI_Params.fOutMaxLimit := 100.0;
        stCTRL_PI_Params.fOutMinLimit := 0.0;
        bInit := FALSE;
END_IF
fbCTRL_PI( fActualValue := fbCTRL_PT1.fOut,
         fSetpointValue := fSetpointValue,
         eMode := eCTRL_MODE_ACTIVE,
         stParams := stCTRL_PI_Params );
fbCTRL_PT1( fIn := fbCTRL_PI.fOut,
            eMode := eCTRL_MODE_ACTIVE,
            stParams := stCTRL_PT1_Params,
            fOut => fActualValue );
END_IF
```

4.2.1.1.3 FB_CTRL_LOOP_SCHEDULER



This function block allows the system load to be distributed over a number of control loops that are parameterized using the same `tCtrlCycleTime` and for which the condition "tCtrlCycleTime > tTaskCycleTime" is true. The output vector calculated by this function block is used to start the individual control loops at different times, so that the system loading is distributed.

Behavior of the output vector



5 control loops are managed in this diagram. In this case, "tCtrlCycleTime = 6 · tTaskCycleTime".

The programmer must create the following array in the PLC if this function block is to be used:

```
arrOutputVector : ARRAY[1..nNumberOfControlLoops] OF BOOL;
```

The function block sets the bits in this vector to TRUE or FALSE. The control loops that are managed with the loop scheduler are to be switched into the `eCTRL_MODE_ACTIVE` state when the corresponding bit in the output vector is TRUE. See sample code below.

Inputs

```
VAR_INPUT
    nManValue : DWORD;
    eMode     : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
nManValue	DWORD	This input allows the first 32 bits in the output vector to be set in `eCTRL_MODE_MANUAL`. A 1 sets the first bit, a 2 the second bit, a 3 the first and second bits, ...
eMode	E_CTRL_MODE	Input that specifies the operation mode [▶ 173] of the function block.

Outputs

```
VAR_OUTPUT
    eState      : E_CTRL_STATE;
    eErrorId    : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

Name	Type	Description
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▸ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams     : ST_CTRL_LOOP_SCHEDULER_PARAMS;
END_VAR
```

Name	Type	Description
stParams	ST_CTRL_ LOOP_ SCHEDULER_ PARAMS	Parameter structure for the loop scheduler.

`stParams` consists of the following elements:

```
TYPE
ST_CTRL_LOOP_SCHEDULER_PARAMS:
STRUCT
    tCtrlCycleTime          : TIME := T#0ms;
    tTaskCycleTime          : TIME := T#0ms;
    nNumberOfControlLoops   : UINT;
    pOutputVector_ADR       : POINTER TO BOOL := 0;
    nOutputVector_SIZEOF    : UINT := 0;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	The cycle time with which the control loops managed by the loop scheduler are processed. This must be greater than or equal to the **TaskCycleTime**.
tTaskCycle Time	TIME	The cycle time with which the loop scheduler and the function blocks associated with the control loops are called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
nNumberOfContr olLoops	UINT	The number of control loops being managed.
pOutputVector_A DR	POINTER TO BOOL	Address of the output vector
nOutputVector_S IZEOF	UINT	Size of the output vector in bytes

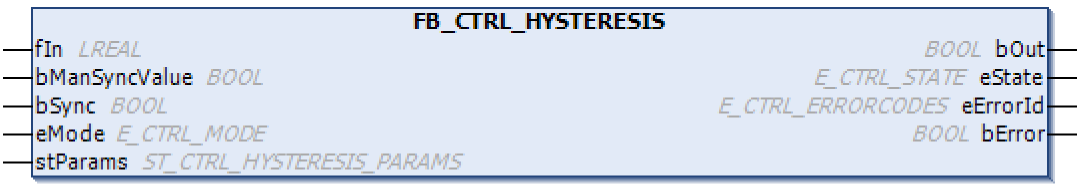
Sample:

```
PROGRAM PRG_LoopScheduler
VAR
    arrOutputVector         : ARRAY[1..5] OF BOOL;
    eMode                   : E_CTRL_MODE;
    stParams                : ST_CTRL_LOOP_SCHEDULER_PARAMS;
    eErrorId                : E_CTRL_ERRORCODES;
    bError                  : BOOL;
    fbCTRL_LoopScheduler    : FB_CTRL_LOOP_SCHEDULER;
    bInit                   : BOOL := TRUE;
    eMode_CtrlLoop_1        : E_CTRL_MODE;
    eMode_CtrlLoop_2        : E_CTRL_MODE;
```

```
    eMode_CtrlLoop_3       : E_CTRL_MODE;
    eMode_CtrlLoop_4       : E_CTRL_MODE;
    eMode_CtrlLoop_5       : E_CTRL_MODE;
END_VAR
IF bInit
THEN
    stParams.tCtrlCycleTime          := T#10ms;
    stParams.tTaskCycleTime          := T#2ms;
    stParams.nNumberOfControlLoops   := 5;
    bInit                            := FALSE;
END_IF
stParams.nOutputVector_SIZEOF   := SIZEOF(arrOutputVector);
stParams.pOutputVector_ADR      := ADR(arrOutputVector);
fbCTRL_LoopScheduler( eMode      := eMode,
                 stParams   := stParams,
                 eErrorId   => eErrorId,
                 bError     => bError);
IF arrOutputVector[1] THEN
    eMode_CtrlLoop_1 := eCTRL_MODE_ACTIVE;
END_IF
IF arrOutputVector[2] THEN
    eMode_CtrlLoop_2 := eCTRL_MODE_ACTIVE;
END_IF
IF arrOutputVector[3] THEN
    eMode_CtrlLoop_3 := eCTRL_MODE_ACTIVE;
END_IF
IF arrOutputVector[4] THEN
    eMode_CtrlLoop_4 := eCTRL_MODE_ACTIVE;
END_IF
IF arrOutputVector[5]
THEN
    eMode_CtrlLoop_5 := eCTRL_MODE_ACTIVE;
END_IF
```

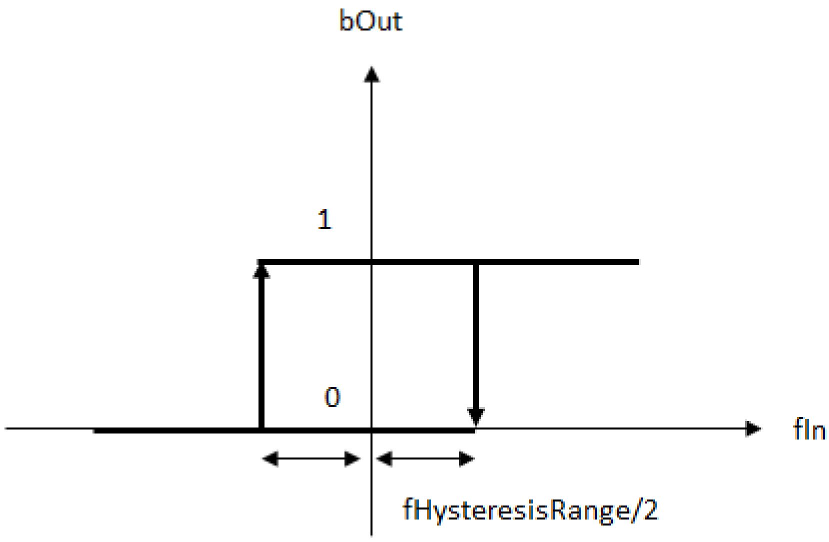
4.2.1.2 Base

4.2.1.2.1 FB_CTRL_HYSTERESIS



The function block provides a hysteresis transfer element in a functional diagram.

Transfer function



Inputs

```
VAR_INPUT
    fIn             : FLOAT;
    bManSyncValue   : BOOL;
    bSync           : BOOL;
    eMode           : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input of the hysteresis element
bManSyncValue	BOOL	Input through which the hysteresis element can be set to one of the two junctions.
bSync	BOOL	A rising edge at this input sets the hysteresis element to the value "fManSyncValue".
eMode	E_CTRL_MODE	Input that specifies the operation mode [▸ 173] of the function block.

Outputs

```
VAR_OUTPUT
    bOut        : BOOL;
    eState      : E_CTRL_STATE;
    eErrorId    : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

Name	Type	Description
bOut	BOOL	Output of the hysteresis element
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERRORCODES	Supplies the error number [▸ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE as soon as an error situation occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams    : ST_CTRL_HYSTERESIS_PARAMS;
END_VAR
```

Name	Type	Description
stParams	ST_CTRL_ HYSTERESIS_ PARAMS	Parameter structure of the hysteresis element

stParams consists of the following elements:

```
TYPE
ST_CTRL_HYSTERESIS_PARAMS :
STRUCT
    tCtrlCycleTime      : TIME := T#0ms;
    tTaskCycleTime      : TIME := T#0ms;
    fHysteresisRange    : FLOAT;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
fHysteresis Range	FLOAT	Hysteresis range, see diagram above.

4.2.1.2.2 FB_CTRL_P

The function block provides an P-transfer element in the functional diagram.

Transfer function:

$$G(s) = K_p$$

Inputs

```
VAR_INPUT
    fIn       :FLOAT;
    eMode     :E_CTRL_MODE;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input of the P element
eMode	E_CTRL_MODE	Input that specifies the operation mode of the function block.

Outputs

```
VAR_OUTPUT
    fOut        :FLOAT;
    eState      :E_CTRL_STATE;
    eErrorId    :E_CTRL_ERRORCODES;
    bError      :BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output of the P element
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▸ 173] when the output bError is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams    :ST_CTRL_P_PARAMS;
END_VAR
```

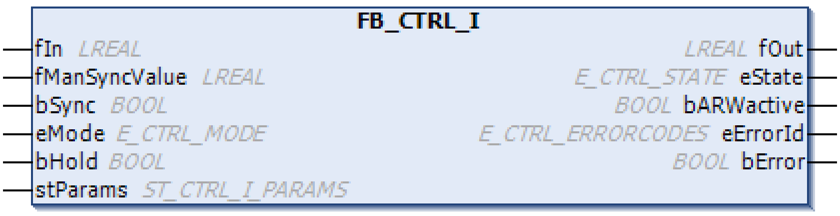
Name	Type	Description
stParams	ST_CTRL_P_ PARAMS	Parameter structure of the P element

stParams consists of the following elements:

```
TYPE ST_CTRL_P_PARAMS:
STRUCT
    tCtrlCycleTime      : TIME := T#0ms;
    tTaskCycleTime      : TIME := T#0ms;
    fKp                 : FLOAT := 0.0;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
fKp	FLOAT	Proportional gain of the P-element

4.2.1.2.3 FB_CTRL_I



The function block provides an I-transfer element in the functional diagram.

Transfer function

$$G(s) = \frac{1}{T_I s}$$

Inputs

```
VAR_INPUT
    fIn                 : FLOAT;
    fManSyncValue       : FLOAT;
    bSync               : BOOL;
    eMode               : E_CTRL_MODE;
    bHold               : BOOL;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input of the I element
fManSyncValue	FLOAT	Input to which the I-element can be synchronized, or whose value is the present at the output in Manual Mode.
bSync	BOOL	A rising edge at this input sets the integrator to the value "fManSyncValue".
eMode	E_CTRL_MODE	Input that specifies the operation mode of the function block.
bHold	BOOL	A TRUE at this input holds the integrator fixed at the current value, independently of the input `fIn`.

Outputs

```
VAR_OUTPUT
    fOut             : FLOAT;
    bARWactive       : BOOL;
    eState           : E_CTRL_STATE;
    eErrorId         : E_CTRL_ERRORCODES;
    bError           : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output of the I element
bARWactive	BOOL	A TRUE at this output indicates that the integrator is being restricted.
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERROR CODES	Supplies the error number when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams     : ST_CTRL_I_PARAMS;
END_VAR
```

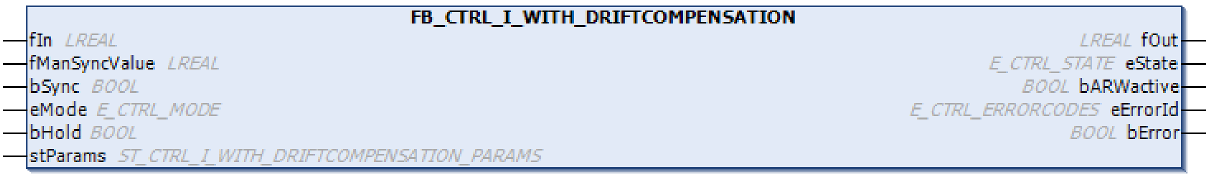
Name	Type	Description
stParams	ST_CTRL_I_ PARAMS	Parameter structure of the I element

`stParams` consists of the following elements:

```
TYPE ST_CTRL_I_PARAMS:
STRUCT
    tCtrlCycleTime     : TIME  := T#0ms;
    tTaskCycleTime     : TIME  := T#0ms;
    tTi                : TIME  := T#0ms;
    fOutMaxLimit       : FLOAT := 1E38;
    fOutMinLimit       : FLOAT := -1E38;
END_STRUCT
END_TYPE
```

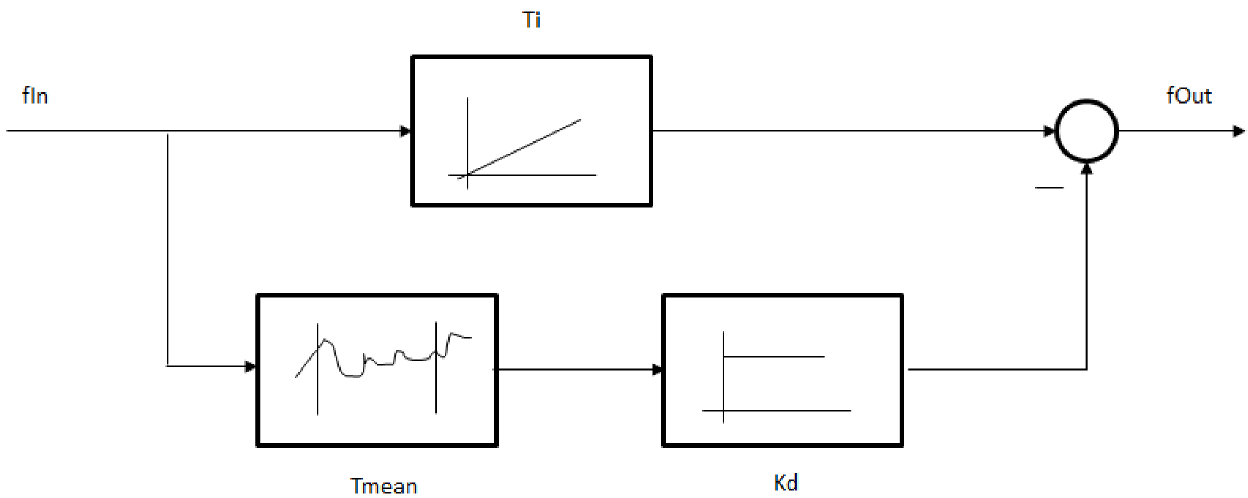
Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the `TaskCycleTime`. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
tTi	TIME	Integration time of the I-element
fOutMaxLimit	FLOAT	Upper limit at which integration is halted (ARW measure). Reaching this limit is indicated by a TRUE at the `bARWActive` output.
fOutMinLimit	FLOAT	Lower limit at which integration is halted (ARW measure). Reaching this limit is indicated by a TRUE at the `bARWActive` output.

4.2.1.2.4 FB_CTRL_I_WITH_DRIFTCOMPENSATION



The function block represents an I transfer element with drift compensation.

Functional diagram



Inputs

```
VAR_INPUT
    fIn                 : FLOAT;
    fManSyncValue       : FLOAT;
    bSync               : BOOL;
    eMode               : E_CTRL_MODE;
    bHold               : BOOL;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input of the I element
fManSyncValue	FLOAT	Input to which the I-element can be synchronized, or whose value is the present at the output in Manual Mode.
bSync	BOOL	A rising edge at this input sets the integrator to the value "fManSyncValue".
eMode	E_CTRL_MODE	Input that specifies the operation mode of the function block.
bHold	BOOL	A TRUE at this input holds the integrator fixed at the current value, independently of the input `fIn`.

Outputs

```
VAR_OUTPUT
    fOut          : FLOAT;
    bARWactive    : BOOL;
    eState        : E_CTRL_STATE;
    eErrorId      : E_CTRL_ERRORCODES;
    bError        : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output of the I element
bARWactive	BOOL	A TRUE at this output indicates that the integrator is being restricted.
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERRORCODES	Supplies the error number [▸ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams    : ST_CTRL_I_WITH_DRIFTCOMPENSATION_PARAMS;
END_VAR
```

Name	Type	Description
stParams	ST_CTRL_I_WITH_DRIFTCOMPENSATION_PARAMS	Parameter structure of the I element

`stParams` consists of the following elements:

```
TYPE
ST_CTRL_I_WITH_DRIFTCOMPENSATION_PARAMS:
STRUCT
    tCtrlCycleTime          : TIME := T#0ms;
    tTaskCycleTime          : TIME := T#0ms;
    tTi                     : TIME := T#0ms;
    fOutMaxLimit            : FLOAT := 1E38;
    fOutMinLimit            : FLOAT := -1E38;
    fDampingCoefficient     : FLOAT := 0.0;
    tAveragingTime          : TIME := T#0ms;
    pWorkArray_ADR          : POINTER TO FLOAT := 0;
    nWorkArray_SIZEOF       : UINT := 0;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
tTi	TIME	Integration time of the I-element
fOutMaxLimit	FLOAT	Upper limit at which integration is halted (ARW measure). Reaching this limit is indicated by a TRUE at the `bARWActive` output.
fOutMinLimit	FLOAT	Lower limit at which integration is halted (ARW measure). Reaching this limit is indicated by a TRUE at the `bARWActive` output.
fDampingCoefficient	FLOAT	Factor k_d in the functional diagram
tAveragingTime	TIME	Time across which the sliding mean value filter calculates the mean value.
pWorkArray_ADR	POINTER TO FLOAT	Address of an Array of FLOAT with size `tAveragingTime / tCtrlCycleTime` (see description of the function block FB_CTRL_MOVING_AVERAGE)
nWorkArray_SIZEOF	UINT	Size of an Array of FLOAT with size `tAveragingTime / tCtrlCycleTime` (see description of the function block FB_CTRL_MOVING_AVERAGE)

4.2.1.2.5 FB_CTRL_D

The function block provides a DT_1 transfer element (a real D-element) in a functional diagram.

Transfer function (continuous)

$$G(s) = \frac{T_v s}{1 + T_d s}$$

Inputs

```
VAR_INPUT
    fIn                : FLOAT;
    fManSyncValue      : FLOAT;
    bSync              : BOOL;
    eMode              : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input of the D element
fManSyncValue	FLOAT	Input to which the D-element can be synchronized, or whose value is present at the output in Manual Mode.
bSync	BOOL	A rising edge at this input sets the D-element to the value "fManSyncValue".
eMode	E_CTRL_MODE	Input that specifies the operation mode [▸ 173] of the function block.

Outputs

```
VAR_OUTPUT
    fOut        : FLOAT;
    eState      : E_CTRL_STATE;
    eErrorId    : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output of the D element
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERRORCODES	Supplies the error number [▸ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams    : ST_CTRL_D_PARAMS;
END_VAR
```

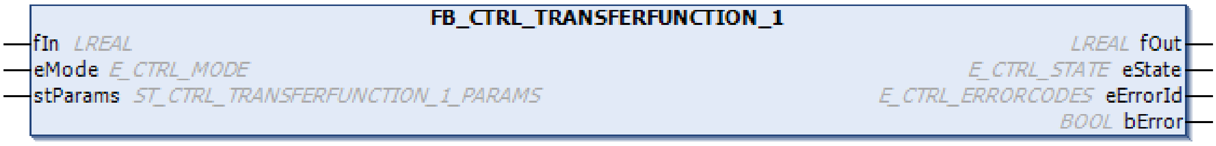
Name	Type	Description
stParams	ST_CTRL_D_PARAMS	Parameter structure of the D-element

`stParams` consists of the following elements:

```
TYPE ST_CTRL_D_PARAMS:
STRUCT
    tCtrlCycleTime   : TIME   := T#0ms;
    tTaskCycleTime   : TIME   := T#0ms;
    tTv              : TIME   := T#0ms;
    tTd              : TIME   := T#0ms;
    fOutMaxLimit     : FLOAT  := 1E38;
    fOutMinLimit     : FLOAT  := -1E38;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
tTv	TIME	Differentiation time constant
tTd	TIME	Damping time constant
fOutMaxLimit	FLOAT	Upper limit to which the output of the D-element is restricted.
fOutMinLimit	FLOAT	Lower limit to which the output of the D-element is restricted.

4.2.1.2.6 FB_CTRL_TRANSFERFUNCTION_1



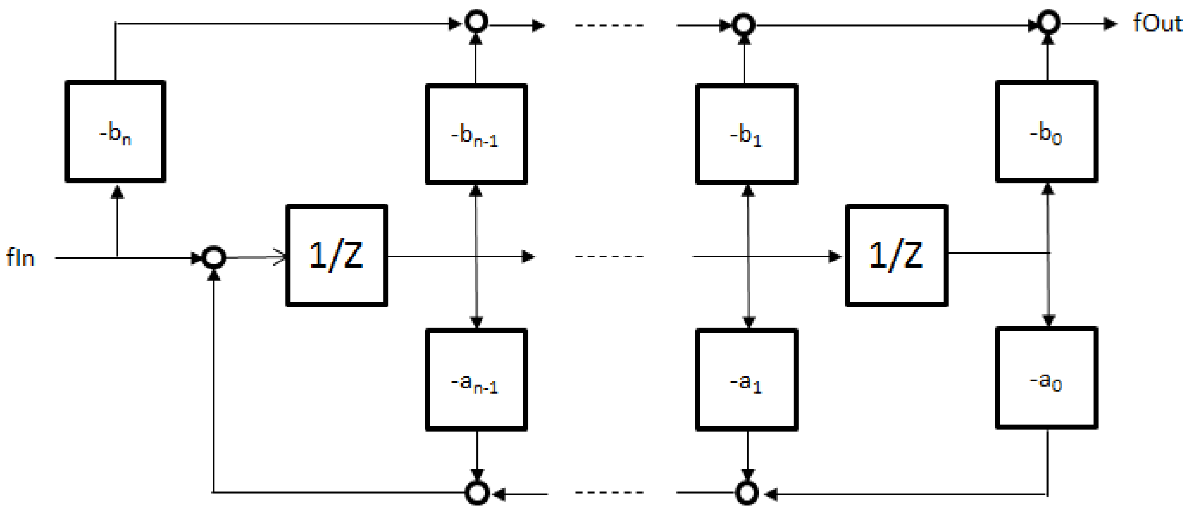
This function block calculates a discrete transfer function with the first standard form shown below. The transfer function here can be of any order, n.

The coefficients for the following transfer functions are stored in the parameter arrays:

$$G(z) = \frac{b_n z^n + b_{(n-1)} z^{(n-1)} + \ldots + b_1 z + b_0}{z^n + a_{(n-1)} z^{(n-1)} + \ldots + a_1 z + a_0}$$

Description of the transfer behavior

The above transfer function is calculated with the first standard form after some transformations in each scanning step.



The programmer must create the following arrays in the PLC if this function block is to be used:

```
aNumArray         : ARRAY[0..nNumOrder] OF FLOAT;
aDenArray         : ARRAY[0..nDenOrder] OF FLOAT;
aStructTfData     : ARRAY[0..nTfOrder] OF ST_CTRL_TRANSFERFUNCTION_1_DATA;
```

The coefficients "b_0" to "b_n" are stored in the array `aNumArray`. This must be organized as follows:

```
aNumArray[0]  := b0;
aNumArray[1]  := b1;
...
aNumArray[n-1] := bn-1;
aNumArray[n]  := bn;
```

The coefficients "a_0" to "a_n" are stored in the array `ar_DenominatorArray`. This must be organized as follows:

```
aDenArray[0]  := a0;
aDenArray[1]  := a1;
...
aDenArray[n-1] := an-1;
aDenArray[n]  := an;
```

The internal data required by the function block is stored in the `ar_stTransferfunction1Data` array. This data must never be modified from within the PLC program. This procedure is also illustrated in the sample program listed below.

Inputs

```
VAR_INPUT
    fIn           : FLOAT;
    eMode         : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input to the transfer function
eMode	E_CTRL_MODE	Input that specifies the operation mode of the function block.

Outputs

```
VAR_OUTPUT
    fOut        : FLOAT;
    eState      : E_CTRL_STATE;
    bError      : BOOL;
    eErrorId    : E_CTRL_ERRORCODES;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output from the transfer function
eState	E_CTRL_STATE	State of the function block
bError	BOOL	Supplies the error number [▸ 173] when the output `bError` is set.
eErrorId	E_CTRL_ERROR CODES	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams    : ST_CTRL_TRANSFERFUNCTION_1_PARAMS;
END_VAR
```

Name	Type	Description
stParams	ST_CTRL_TRAN SFERFUNCTION _1_PARAMS	Parameter structure of the function block

`stParams` consists of the following elements:

```
TYPE
ST_CTRL_TRANSFERFUNCTION_1_PARAMS:
STRUCT
    tTaskCycleTime                  : TIME;
    tCtrlCycleTime                  : TIME := T#0ms;
    nOrderOfTheTransferfunction     : USINT;
    pNumeratorArray_ADR             : POINTER TO FLOAT := 0;
    nNumeratorArray_SIZEOF          : UINT;
    pDenominatorArray_ADR           : POINTER TO FLOAT := 0;
    nDenomiantorArray_SIZEOF        : UINT;
    pTransferfunction1Data_ADR      : POINTER TO
ST_CTRL_TRANSFERFUNCTION_1_DATA;
    nTransferfunction1Data_SIZEOF    : UINT;
END_STRUCT
END_TYPE
```

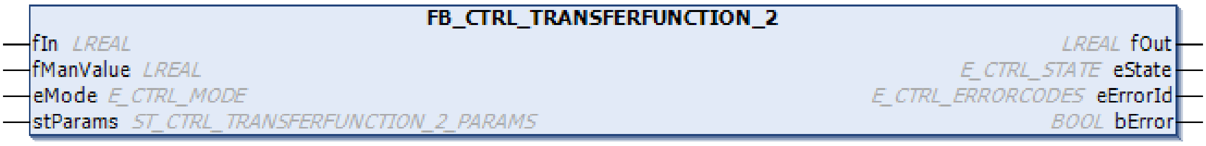
Name	Type	Description
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
nOrderOfTheTransferfunction	USINT	Order of the transfer function [0...]
pNumeratorArray_ADR	POINTER TO FLOAT	Address of the array containing the numerator coefficients
nNumeratorArray_SIZEOF	UINT	Size of the array containing the numerator coefficients, in bytes
pDenominatorArray_ADR	POINTER TO FLOAT	Address of the array containing the denominator coefficients
nDenomiantorArray_SIZEOF	UINT	Size of the array containing the denominator coefficients, in bytes
pTransferfunction1Data_ADR	POINTER TO ST_CTRL_TRANSFERFUNCTION_1_DATA	Address of the data array
nTransferfunction1Data_SIZEOF	UINT	Size of the data array in bytes

Example:

```
PROGRAM PRG_TRANSFERFUNCTION_1_TEST
VAR CONSTANT
    nTfOrder     : USINT := 2;
END_VAR
VAR
    aNumArray           : ARRAY[0..nNumOrder] OF FLOAT;
    aDenArray           : ARRAY[0..nDenOrder] OF FLOAT;
    aStTfData           : ARRAY[0..nTfOrder] OF ST_CTRL_TRANSFERFUNCTION_1_DATA;
    eMode               : E_CTRL_MODE;
    stParams            : ST_CTRL_TRANSFERFUNCTION_1_PARAMS;
    eErrorId            : E_CTRL_ERRORCODES;
    bError              : BOOL;
    fbTansferfunction   : FB_CTRL_TRANSFERFUNCTION_1;
    bInit               : BOOL := TRUE;
    fIn                 : FLOAT := 0;
    fOut                : FLOAT;
    b_0, b_1, b_2       : FLOAT;
    a_0,a_1,a_2         : FLOAT;
END_VAR
IF bInit THEN
    aNumArray[0] := 1.24906304658218E-007;
    aNumArray[1] := 2.49812609316437E-007;
    aNumArray[2] := 1.24906304658218E-007;
    aDenArray[0] := 0.998501124344101;
    aDenArray[1] := -1.99850062471888;
    aDenArray[2] := 1.0;
    stParams.tTaskCycleTime        := T#2ms;
    stParams.tCtrlCycleTime        := T#2ms;
    stParams.nOrderOfTheTransferfunction := nTfOrder;
    eMode := eCTRL_MODE_ACTIVE;
    bInit := FALSE;
END_IF
stParams.pNumeratorArray_ADR            := ADR(aNumArray);
stParams.nNumeratorArray_SIZEOF         := SIZEOF(aNumArray);
stParams.pDenominatorArray_ADR          := ADR(aDenArray);
stParams.nDenominatorArray_SIZEOF       := SIZEOF(aDenArray);
stParams.pTransferfunction1Data_ADR     := ADR(aStructTfData);
stParams.nTransferfunction1Data_SIZEOF  := SIZEOF(aStructTfData);
fbTansferfunction (fIn := fIn,
                   eMode := eMode,
                   stParams := stParams,
```

```
                fOut => fOut,
                eErrorId => eErrorId,
                bError => bError);
```

4.2.1.2.7 FB_CTRL_TRANSFERFUNCTION_2



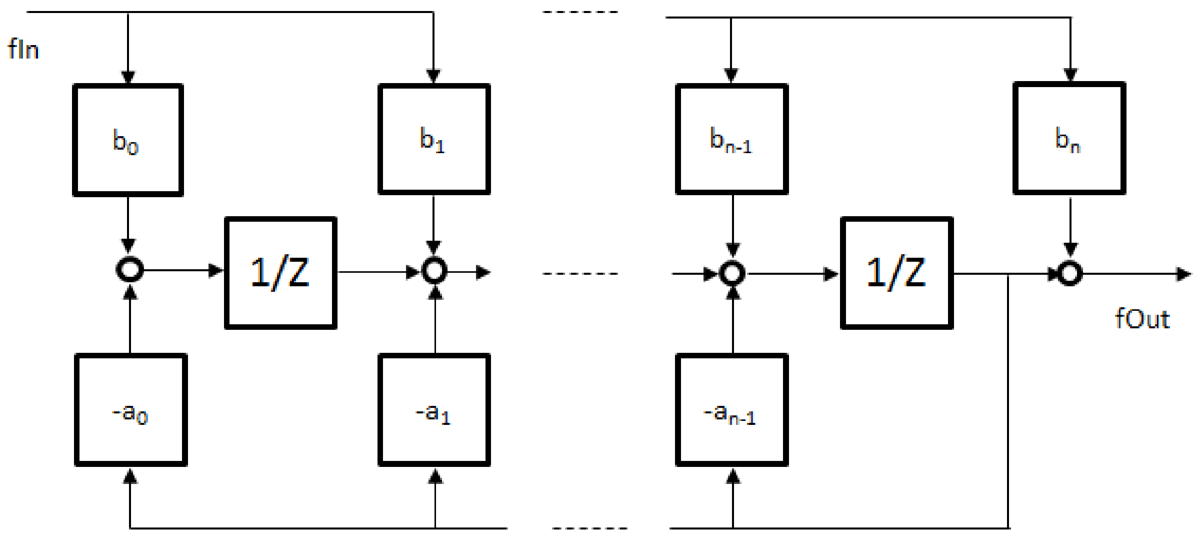
This function block calculates a discrete transfer function with the second standard form shown below. The transfer function here can be of any order, n.

The coefficients for the following transfer functions are stored in the parameter arrays:

$$G(z) = \frac{b_n z^n + b_{(n-1)} z^{(n-1)} + \ldots + b_1 z + b_0}{z^n + a_{(n-1)} z^{(n-1)} + \ldots + a_1 z + a_0}$$

Description of the transfer behavior

The internal calculation is performed in each sampling step according to the second standard form, for which the following block diagram applies:



The transfer function can be brought into the form shown in the block diagram by means of some transformations:

$$G(z) = \tilde{b} + \frac{\tilde{b}_{(n-1)} z^{-1} + \ldots + \tilde{b}_1 z^{-(n-1)} + \tilde{b}_0 z^{-n}}{1 + a_{(n-1)} z^{-1} + \ldots + a_1 z^{-(n-1)} + a_0 z^{-n}}$$

The coefficients of the counter polynomial are calculated according to the following rule:

$$\tilde{b}_i = b_i - b_n a_i \qquad \forall 0 \le i < n$$

$$\tilde{b}_n = b_n$$

The programmer must create the following arrays in the PLC if this function block is to be used:

```
aNumArray     : ARRAY[0..nTfOrder] OF FLOAT;
aDenArray     : ARRAY[0..nTfOrder] OF FLOAT;
aStTfData     : ARRAY[0..nTfOrder] OF ST_CTRL_TRANSFERFUNCTION_2_DATA;
```

The coefficients "b_0" to "b_n" are stored in the array `aNumArray`. This must be organized as follows:

```
aNumArray[0]:= b0;
aNumArray[1]:= b1;
...
aNumArray[n-1] := bn-1;
aNumArray[n] := bn;
```

The coefficients "a_0" to "a_n" are stored in the array `aDenArray`. This must be organized as follows:

```
aDenArray[0]:= a0;
aDenArray[1]:= a1;
...
aDenArray[n-1] := an-1;
aDenArray[n] := an;
```

The internal data required by the function block is stored in the array `aStTfData`. This data must never be modified from within the PLC program. This procedure is also illustrated in the sample program listed below.

Inputs

```
VAR_INPUT
    fIn           : FLOAT;
    fManValue     : FLOAT;
    eMode         : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input to the transfer function
fManValue	FLOAT	Input whose value is present at the output in manual mode.
eMode	E_CTRL_MODE	Input that specifies the operation mode of the function block.

Outputs

```
VAR_OUTPUT
    fOut         : FLOAT;
    eState       : E_CTRL_STATE;
    bError       : BOOL;
    eErrorId     : E_CTRL_ERRORCODES;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output from the transfer function
eState	E_CTRL_STATE	State of the function block
bError	BOOL	Supplies the error number [▸ 173] when the output `bError` is set.
eErrorId	E_CTRL_ERRORCODES	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams     : ST_CTRL_TRANSFERFUNCTION_2_PARAMS;
END_VAR
```

Name	Type	Description
stParams	ST_CTRL_TRANSFERFUNCTION_2_PARAMS	Parameter structure of the function block

`stParams` consists of the following elements:

```
TYPE
ST_CTRL_TRANSFERFUNCTION_2_PARAMS:
STRUCT
```

```
    tTaskCycleTime                 : TIME;
    tCtrlCycleTime                 : TIME := T#0ms;
    nOrderOfTheTransferfunction    : USINT;
    pNumeratorArray_ADR            : POINTER TO FLOAT := 0;
    nNumeratorArray_SIZEOF         : UINT;
    pDenominatorArray_ADR          : POINTER TO FLOAT := 0;
    nDenomiantorArray_SIZEOF       : UINT;
    pTransferfunction2Data_ADR     : POINTER TO
ST_CTRL_TRANSFERFUNCTION_2_DATA;
    nTransferfunction2Data_SIZEOF : UINT;
END_STRUCT
END_TYPE
```

Name	Type	Description
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
nOrderOfTheTransferfunction	USINT	Order of the transfer function [0...]
pNumeratorArray_ADR	POINTER TO FLOAT	Address of the array containing the numerator coefficients
nNumeratorArray_SIZEOF	UINT	Size of the array containing the numerator coefficients, in bytes
pDenominatorArray_ADR	POINTER TO FLOAT	Address of the array containing the denominator coefficients
nDenomiantorArray_SIZEOF	UINT	Size of the array containing the denominator coefficients, in bytes
pTransferfunction2Data_ADR	POINTER TO ST_CTRL_TRANSFERFUNCTION_2_DATA	Address of the data array
nTransferfunction2Data_SIZEOF	UINT	Size of the data array in bytes

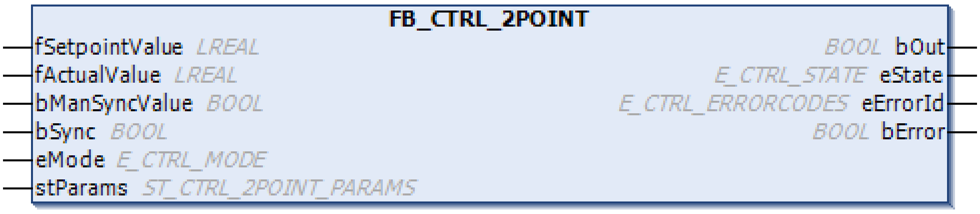
Sample:

```
PROGRAM PRG_TRANSFERFUNCTION_2_TEST
VAR CONSTANT
    nTfOrder : USINT := 2;
END_VAR
VAR
    aNumArray            : ARRAY[0..nTfOrder] OF FLOAT;
    aDenArray            : ARRAY[0..nTfOrder] OF FLOAT;
    aStTfData            : ARRAY[0..nTfOrder] OF ST_CTRL_TRANSFERFUNCTION_2_DATA;
    eMode                : E_CTRL_MODE;
    stParams             : ST_CTRL_TRANSFERFUNCTION_2_PARAMS;
    eErrorId             : E_CTRL_ERRORCODES;
    bError               : BOOL;
    fbTansferfunction    : FB_CTRL_TRANSFERFUNCTION_2;
    bInit                : BOOL := TRUE;
    fIn                  : FLOAT := 0;
    fOut                 : FLOAT;
    b_0, b_1, b_2        : FLOAT;
    a_0,a_1,a_2          : FLOAT;
END_VAR
IF bInit THEN
    aNumArray[0] := 1.24906304658218E-007;
    aNumArray[1] := 2.49812609316437E-007;
    aNumArray[2] := 1.24906304658218E-007;
    aDenArray[0] := 0.998501124344101;
    aDenArray[1] := -1.99850062471888;
    aDenArray[2] := 1.0;
    stParams.tTaskCycleTime    := T#2ms;
    stParams.tCtrlCycleTime    := T#2ms;
    stParams.nOrderOfTheTransferfunction := nTfOrder;
    eMode := eCTRL_MODE_ACTIVE;
```

```
    bInit := FALSE;
END_IF
stParams.pNumeratorArray_ADR := ADR(aNumArray);
stParams.nNumeratorArray_SIZEOF := SIZEOF(aNumArray);
stParams.pDenominatorArray_ADR := ADR(aDenArray );
stParams.nDenominatorArray_SIZEOF := SIZEOF(aDenArray );
stParams.pTransferfunction2Data_ADR := ADR(aStTfData );
stParams.nTransferfunction2Data_SIZEOF := SIZEOF(aStTfData);
fbTansferfunction (fIn := fIn,
                   eMode := eMode,
                   stParams := stParams,
                   fOut => fOut,
                   eErrorId => eErrorId,
                   bError => bError);
```

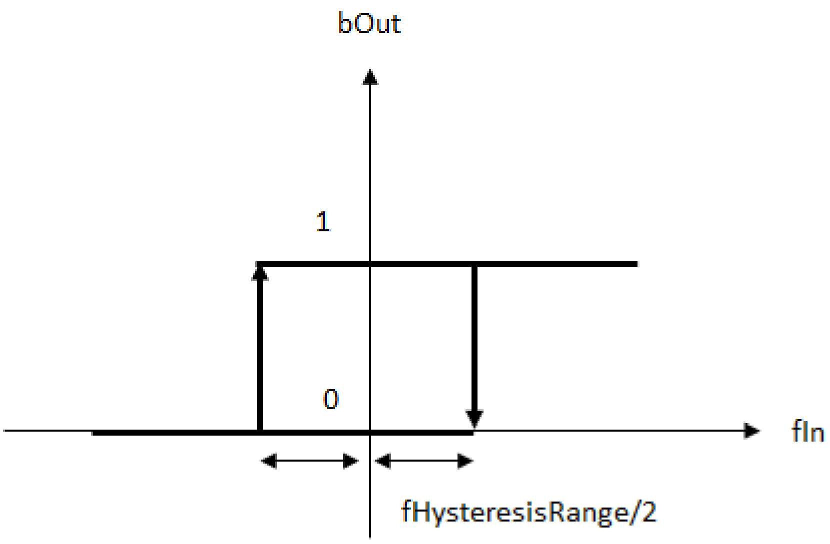
4.2.1.3 Controller

4.2.1.3.1 FB_CTRL_2POINT



The function block provides a 2-point transfer element in the functional diagram.

Behavior of the output



Inputs

```
VAR_INPUT
    fSetpointValue     : FLOAT;
    fActualValue       : FLOAT;
    bManSyncValue      : BOOL;
    bSync              : BOOL;
    eMode              : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
fSetpointValue	FLOAT	Setpoint of the controlled variable
fActualValue	FLOAT	Actual value of the controlled variable
bManSyncValue	BOOL	Input through which the 2-point element can be set to one of the two junctions.
bSync	BOOL	A rising edge at this input sets the 2-point element to the value "bManSyncValue".
eMode	E_CTRL_MODE	Input that specifies the operation mode [▶ 173] of the function block.

Outputs

```
VAR_OUTPUT
    bOut       : BOOL;
    eState     : E_CTRL_STATE;
    eErrorId   : E_CTRL_ERRORCODES;
    bError     : BOOL;
END_VAR
```

Name	Type	Description
bOut	BOOL	Output of the 2-point element
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▶ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Input/ outputs

```
VAR_IN_OUT
    stParams    : ST_CTRL_2POINT_PARAMS;
END_VAR
```

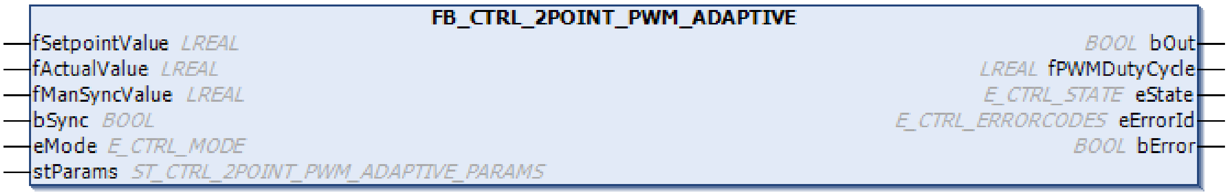
Name	Type	Description
stParams	ST_CTRL_ 2POINT_ PARAMS	Parameter structure of the 2-point element

`stParams` consists of the following elements:

```
TYPE
ST_CTRL_2POINT_PARAMS :
STRUCT
    tCtrlCycleTime       : TIME := T#0ms;
    tTaskCycleTime       : TIME := T#0ms;
    fHysteresisRange     : FLOAT;
END_STRUCT
END_TYPE
```

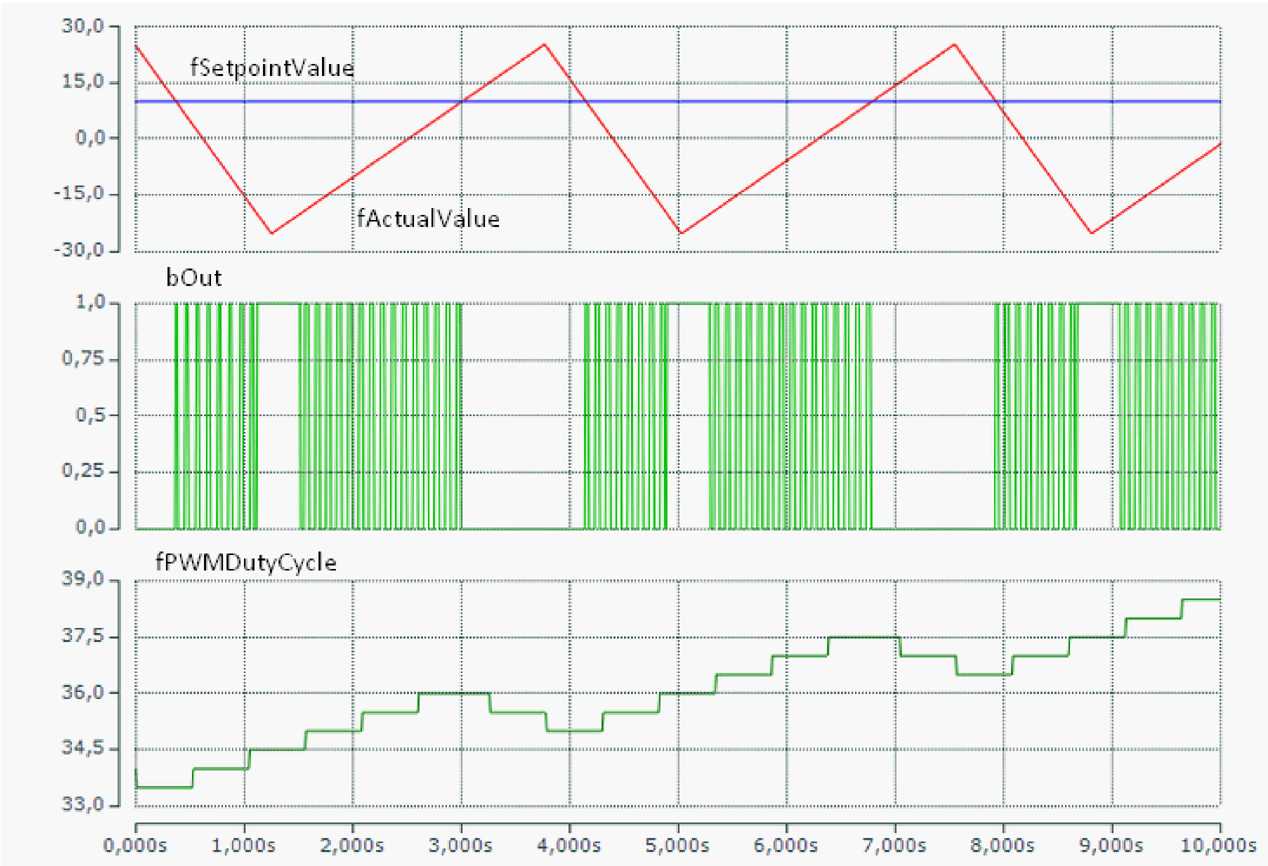
Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycle Time	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
fHysteresis Range	FLOAT	Hysteresis range, see diagram above.

4.2.1.3.2 FB_CTRL_2POINT_PWM_ADAPTIVE



The function block provides an adaptive two-position controller. It is particularly suitable for single-area controllers in which high inlet temperatures are present and which make use of a thermal actuator.

Behavior of the output



Description of the function

Internally, the controller uses a PWM function block that is used to control the thermal actuator. The mark-to-space ratio of the PWM function block is adaptively adjusted to the behavior of the controlled system. The PWM output is switched on when the control deviation "fE = setpoint – actual value" is greater than "0" and switched off when the control deviation is less than "0". The mark-to-space ratio is not changed as long as the control deviation remains within the range "[-fOkRange ... fOkRange]". If "fE > fOkRange", the mark-to-space ratio is increased by "fStepSize". After such an increase, time `tWaitTime` must elapse before the mark-to-space ratio can be changed again. If fE falls below "-fOkRange", the mark-to-space ratio is reduced by "fStepSize". The mark-to-space ratio is only modified over the range "[fMinLimit ... fMaxLimit]". The period of the PWM signal is specified by the parameter `tPWMPeriod`.

Inputs

```
VAR_INPUT
    fSetpointValue      : FLOAT;
    fActualValue        : FLOAT;
```

```
    fManSyncValue      : FLOAT;
    bSync              : BOOL;
    eMode              : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
fSetpointValue	FLOAT	Setpoint of the controlled variable
fActualValue	FLOAT	Actual value of the controlled variable
fManSync Value	FLOAT	Input to which the controller's mark-to-space ratio can be set, or with which the output can be set in Manual Mode. The output is set in Manual Mode if "fManSyncValue > 0.0".
bSync	BOOL	A rising edge at this input will set the mark-to-space ratio of the internal PWM function block to the value "fManSyncValue".
eMode	E_CTRL_MODE	Input that specifies the operation mode [▸ 173] of the function block.

Outputs

```
VAR_OUTPUT
    bOut            : BOOL;
    fPWMDutyCycle   : FLOAT;
    eState          : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError          : BOOL;
END_VAR
```

Name	Type	Description
bOut	BOOL	Controller output
fPWMDutyCycle	FLOAT	Current mark-to-space ratio of the internal PWM function block
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▸ 173] when the output bError is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams     : ST_CTRL_2POINT_PWM_ADAPTIVE_PARAMS;
END_VAR
```

Name	Type	Description
stParams	ST_CTRL_2POIN T_PWM_ ADAPTIVE_ PARAMS	Parameter structure of the 2-point element

stParams consists of the following elements:

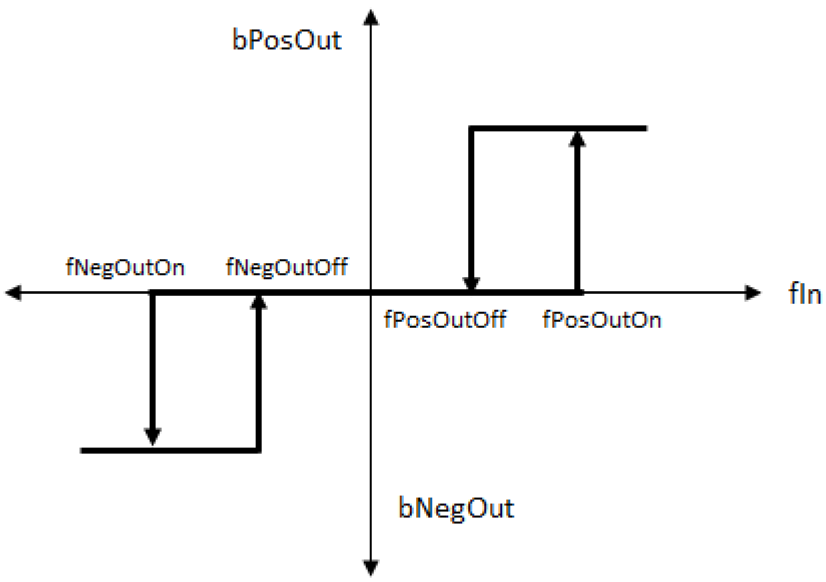
```
TYPE
ST_CTRL_2POINT_PWM_ADAPTIVE_PARAMS:
STRUCT
    tCtrlCycleTime     : TIME := T#0ms;
    tTaskCycleTime     : TIME := T#0ms;
    tPWMPeriod         : TIME;
    fOkRange           : FLOAT;
    fForceRange        : FLOAT;
    fStepSize          : FLOAT;
    fMinLimit          : FLOAT;
    fMaxLimit          : FLOAT;
    tWaitTime          : TIME;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
tPWMPeriod	TIME	Period of the PWM signal
fOkRange	FLOAT	The range of "fE" over which the mark-to-space ratio will not be modified.
fForceRange	FLOAT	If "fE" exceeds this range, the output is permanently set to TRUE.
fStepSize	FLOAT	Value by which the mark-to-space ratio is varied each time it is adapted. [0% ... 100%]
fMinLimit	FLOAT	Maximum mark-to-space ratio in percent [0% ... 100%]
fMaxLimit	FLOAT	Minimum mark-to-space ratio in percent [0% ... 100%]
tWaitTime	TIME	Waiting time between individual modifications of the mark-to-space ratio

4.2.1.3.3 FB_CTRL_3POINT

The function block provides a 3-point transfer element in the functional diagram.

Behavior of the output



Inputs

```
VAR_INPUT
    fSetpointValue   : FLOAT;
    fActualValue     : FLOAT;
    nManSyncValue    : INT;
    bSync            : BOOL;
    eMode            : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
fSetpointValue	FLOAT	Setpoint of the controlled variable
fActualValue	FLOAT	Actual value of the controlled variable

nManSyncValue	INT	Input with which the 3-point link can be set to one of the three junctions.
		`nManSyncValue >= 1 → bPosOut = TRUE, bNegOut = FALSE`
		`nManSyncValue <= -1 → bPosOut = FALSE, bNegOut = TRUE`
		`otherwise → bPosOut = FALSE, bNegOut = FALSE`

bSync	BOOL	A rising edge at this input sets the 3-point element to the value "bManSyncValue".
eMode	E_CTRL_MODE	Input that specifies the operation mode [▸ 173] of the function block.

Outputs

```
VAR_OUTPUT
    bPosOut    : BOOL;
    bNegOut    : BOOL;
    eState     : E_CTRL_STATE;
    eErrorId   : E_CTRL_ERRORCODES;
    bError     : BOOL;
END_VAR
```

Name	Type	Description
bPosOut	BOOL	This output from the 3-point element is TRUE if the upper junction of the characteristic curve is active.
bNegOut	BOOL	This output from the 3-point element is TRUE if the lower junction of the characteristic curve is active.
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERRORCODES	Supplies the error number [▸ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams    : ST_CTRL_3POINT_PARAMS;
END_VAR
```

Name	Type	Description
stParams	ST_CTRL_3POINT_PARAMS	Parameter structure of the 3-point element

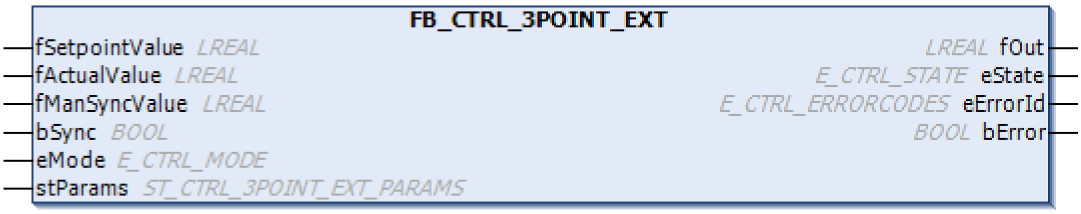
`stParams` consists of the following elements:

```
TYPE
ST_CTRL_3POINT_PARAMS :
STRUCT
    tCtrlCycleTime     : TIME := T#0ms;
    tTaskCycleTime     : TIME := T#0ms;
    fPosOutOn          : FLOAT;
    fPosOutOff         : FLOAT;
    fNegOutOn          : FLOAT;
```

```
    fNegOutOff          : FLOAT;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycle Time	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
fPosOutOn	FLOAT	Control deviation that will result in bPosOut = FALSE being switched to bPosOut = TRUE (bNegOut = FALSE).
fPosOutOff	FLOAT	Control deviation that will result in bPosOut = TRUE being switched to bPosOut = FALSE (bNegOut = FALSE).
fNegOutOn	FLOAT	Control deviation that will result in bNegOut = FALSE being switched to bNegOut = TRUE (bPosOut = FALSE).
fNegOutOff	FLOAT	Control deviation that will result in bNegOut = TRUE being switched to bNegOut = FALSE (bPosOut = FALSE).

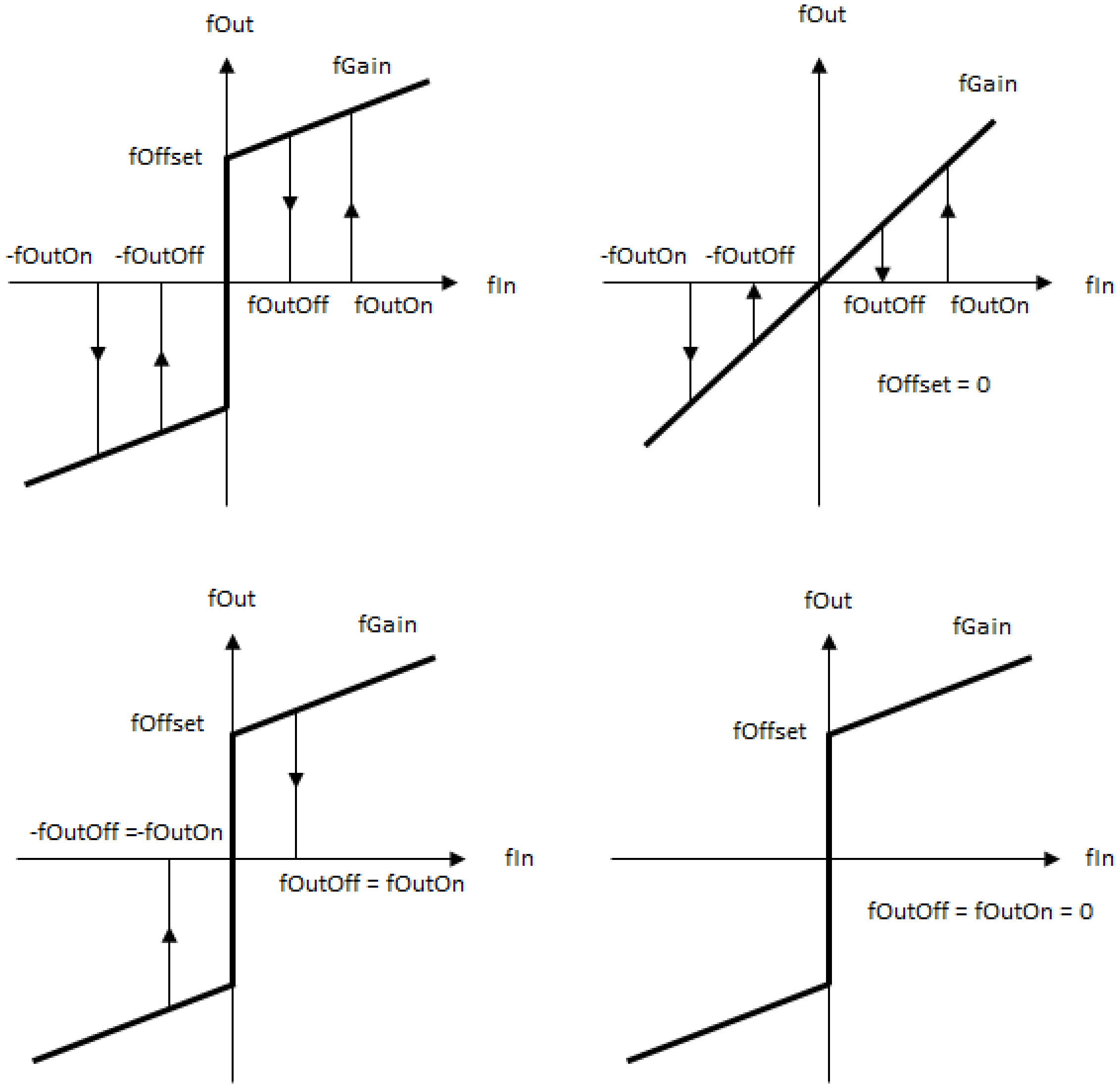
4.2.1.3.4 FB_CTRL_3POINT_EXT



The function block provides an extended 3-point element in the functional diagram.

Behavior of the output

```
fIn := fSetpointValue - fActualValue;
```



Inputs

```
VAR_INPUT
    fSetpointValue      : FLOAT;
    fActualValue        : FLOAT;
    fManSyncValue       : FLOAT;
    bSync               : BOOL;
    eMode               : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
fSetpointValue	FLOAT	Setpoint of the controlled variable
fActualValue	FLOAT	Actual value of the controlled variable
fManSyncValue	FLOAT	Input through which the extended 3-point element can be set to one of the output branches. | fManSyncValue | < 1 → fOut = 0.0 | fManSyncValue | >= 1 → fOut = fE * fGain + fOffset
bSync	BOOL	A rising edge at this input sets the 3-point element to the value "„fManSyncValue".
eMode	E_CTRL_MODE	Input that specifies the operation mode [▶ 173] of the function block.

Outputs

```
VAR_OUTPUT
    fOut        : FLOAT;
    eState      : E_CTRL_STATE;
    eErrorId    : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output of the extended 3-point element
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▸ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams    : ST_CTRL_3POINT_EXT_PARAMS;
END_VAR
```

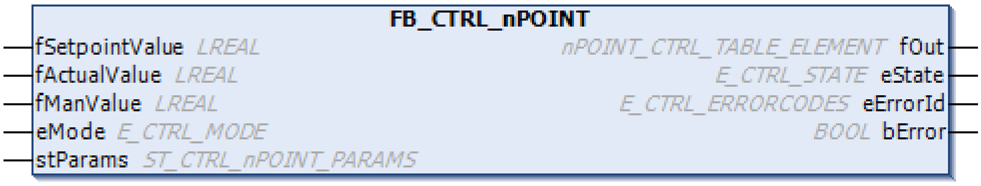
Name	Type	Description
stParams	ST_CTRL_ 3POINT_EXT_ PARAMS	Parameter structure of the extended 3-point element

`stParams` consists of the following elements:

```
TYPE
ST_CTRL_3POINT_EXT_PARAMS :
STRUCT
    tCtrlCycleTime      : TIME := T#0ms;
    tTaskCycleTime      : TIME := T#0ms;
    fOutOff             : FLOAT;
    fOutOn              : FLOAT;
    fGain               : FLOAT;
    fOffset             : FLOAT;
END_STRUCT
END_TYPE
```

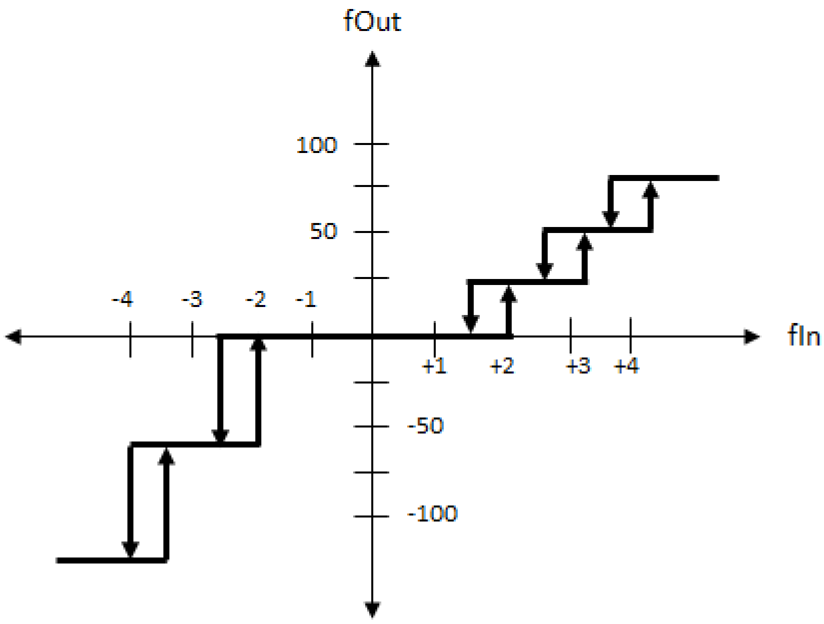
Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycle Time	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
fOutOff	FLOAT	If the control deviation falls below this value, the output is switched off (set to zero).
fOutOn	FLOAT	If the control deviation exceeds this value, the output is switched on.
fGain	FLOAT	Gain factor
fOffset	FLOAT	Offset

4.2.1.3.5 FB_CTRL_nPOINT



The function block provides an n-point transfer element in the functional diagram.

Behavior of the output



Data array for the example:

fE	fOut
xx	-100
-4	-50
-2	0
1	25
2	50
3	75
4	100

The value of the array with the index (1,1), i.e. the left value in the first row can be chosen freely, since it is not evaluated.

Inputs

```
VAR_INPUT
    fSetpointValue      : FLOAT;
    fActualValue        : FLOAT;
```

```
    fManValue           : BOOL;
    eMode               : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
fSetpointValue	FLOAT	Setpoint of the controlled variable
fActualValue	FLOAT	Actual value of the controlled variable
fManValue	BOOL	Input whose value is output in manual mode.
eMode	E_CTRL_MODE	Input that specifies the operation mode [▸ 173] of the function block.

Outputs

```
VAR_OUTPUT
    fOut        : nPOINT_CTRL_TABLE_ELEMENT;
    eState      : E_CTRL_STATE;
    eErrorId    : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

Name	Type	Description
fOut	nPOINT_CTRL_TABLE_ELEMENT	Output of the n-point element
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERRORCODES	Supplies the error number [▸ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams    : ST_CTRL_nPOINT_PARAMS;
END_VAR
```

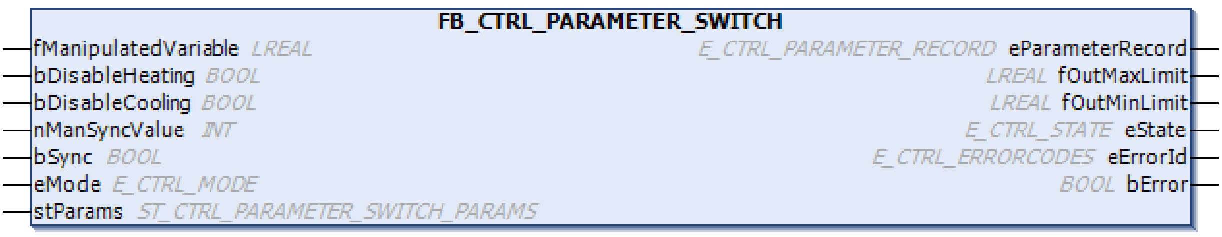
Name	Type	Description
stParams	ST_CTRL_nPOINT_PARAMS	Parameter structure of the n-point element

`stParams` consists of the following elements:

```
TYPE
ST_CTRL_nPOINT_PARAMS :
STRUCT
    tCtrlCycleTime            : TIME := T#0ms;
    tTaskCycleTime            : TIME := T#0ms;
    pDataTable_ADR            : POINTER TO nPOINT_CTRL_TABLE_ELEMENT
     := 0;
    nDataTable_SIZEOF         : UINT := 0;
    nDataTable_NumberOfRows   : UINT := 0;
    fHysteresisRange          : FLOAT;
END_STRUCT
END_TYPE
```

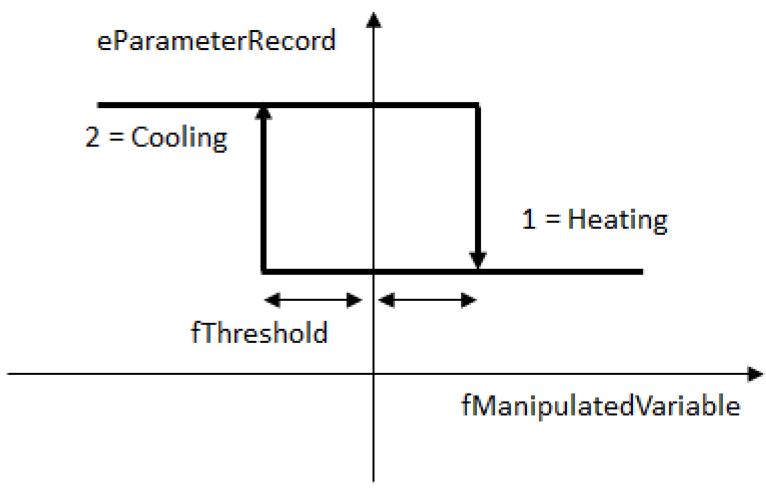
Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycle Time	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
pDataTable_ ADR	POINTER TO nPOINT_CTRL_ TABLE_ ELEMENT	Address of the data table
nDataTable_ SIZEOF	UINT	Size of the data table in bytes
nDataTable_ NumberOf Rows	UINT	Number of rows in the data table
fHysteresis Range	FLOAT	Hysteresis range, see diagram above. The hysteresis range functions as described for FB_CTRL_2POINT [▸ 41].

4.2.1.3.6 FB_CTRL_PARAMETER_SWITCH



This function block can be used to switch the parameter set used by `FB_CTRL_PID_SPLITRANGE`.

Behavior of the output



Description of the function block

This function block is used to switch over the parameter set used by `FB_CTRL_PID_SPLITRANGE`. This function block is particularly intended to switch the parameter sets of controllers that can use two actuators to heat and to cool, and to set the limits for the controller. The time `tMinWaitTime` is specified as an input parameter. At least this time must elapse when switch-over is requested to allow for the parameter range to be changed, and for the controller limits set in such a way that it is possible to switch from heating operation to cooling operation. The intention of this is to prevent the operation mode being changed immediately simply because the controller overshoots slightly.

For heating operation, the parameter range "**eCTRL_PARAMETER_RECORD_HEATING = heating"** is selected, while for cooling operation the parameter range is "**eCTRL_PARAMETER_RECORD_COOLING = cooling"**. The controller's parameter sets must be specified in accordance with this arrangement.

The request for a changeover itself is provided by a 2-point element (see diagram). The controller's output value, in other words the control value, should be used as the input value for the illustrated characteristic hysteresis curve. A request for changeover created by the hysteresis element must be present for at least the specified waiting time, so that the parameter range can be changed.

The `bDisableRange1` and `bDisableRange2` Inputs makes it possible to prevent switching into one of the two ranges. It is therefore possible, for instance, to deactivate heating operation in summer and to deactivate cooling in winter. It would also be possible to make the change in the operation mode depend on the current control deviation. In summer, for instance, it might have to be 2°C too hot before switching into cooling operation. This can also be achieved by connecting the Inputs appropriately.

Maximum and minimum limits are output in addition to providing the output of the parameter range, and these can be copied into the PID controller's parameter set. If the `FB_CTRL_PARAMETER_SWITCH` is in the **Heating** operation mode, the limits are set as follows:

```
fOutMinLimit = -1.0 stParams.fThreshold;

fOutMaxLimit = stParams.fOutMaxLimit;
```

In the **Cooling mode**, the limits are set as follows:

```
fOutMinLimit = stParams.fOutMaxLimit;

fOutMaxLimit = stParams.fThreshold;
```

Inputs

```
VAR_INPUT
    fManipulatedVariable   : FLOAT;
    nManSyncValue          : eCTRL_PARAMETER_RECORD_HEATING;
    bSync                  : BOOL;
    eMode                  : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
fManipulated Variable	FLOAT	Input value of the `FB_Parameter_Switch`. This should be equal to the output value of the controller.
nManSync Value	eCTRL_PARAMETER_RECORD_HEATING	The input with which the function block can be set to one of the parameter ranges.
bSync	BOOL	A rising edge at this input sets the function block to the value "nManSyncValue".
eMode	E_CTRL_MODE	Input that specifies the operation mode [▸ 173] of the function block.

Outputs

```
VAR_OUTPUT
    eParameterRecord     : E_CTRL_PARAMETER_RECORD;
    fOutMaxLimit         : FLOAT;
    fOutMinLimit         : FLOAT;
    eState               : E_CTRL_STATE;
```

```
    eErrorId            : E_CTRL_ERRORCODES;
    bError              : BOOL;
END_VAR
```

Name	Type	Description
eParameterRecord	E_CTRL_PARAMETER_RECORD	The output of the function block, identifying the parameter range.
fOutMaxLimit	FLOAT	The maximum output value of which the controller is limited. (This should be copied into the controller's parameter structure.)
fOutMinLimit	FLOAT	The minimum output value of which the controller is limited. (This should be copied into the controller's parameter structure.)
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERRORCODES	Supplies the error number [▶ 173] when the output bError is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ Outputs

```
VAR_IN_OUT
    stParams    : ST_CTRL_PARAMETER_SWITCH_PARAMS;
END_VAR
```

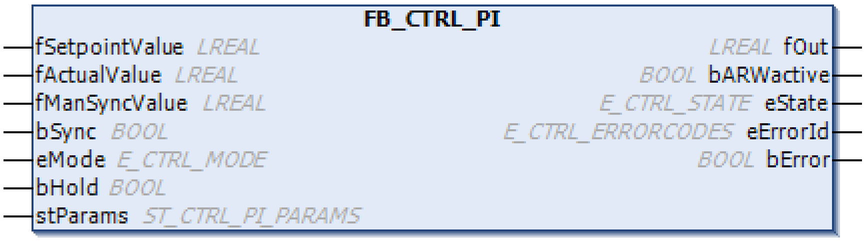
Name	Type	Description
stParams	ST_CTRL_PARAMETER_SWITCH_PARAMS	Parameter structure of the function block

stParams consists of the following elements:

```
TYPE
ST_CTRL_2POINT_PARAMS :
STRUCT
    tTaskCycleTime     : TIME;
    tCtrlCycleTime     : TIME;
    fThreshold         : FLOAT;
    fOutMaxLimit       : FLOAT;
    fOutMinLimit       : FLOAT;
    tMinWaitTime       : TIME;
END_STRUCT
END_TYPE
```

Name	Type	Description
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
fThreshold	FLOAT	Switching threshold, see illustration above.
fOutMaxLimit	FLOAT	The maximum limit; it is passed on to the controller.
fOutMinLimit	FLOAT	The minimum limit; it is passed on to the controller.
tMinWaitTime	TIME	Waiting time, see description above.

4.2.1.3.7 FB_CTRL_PI

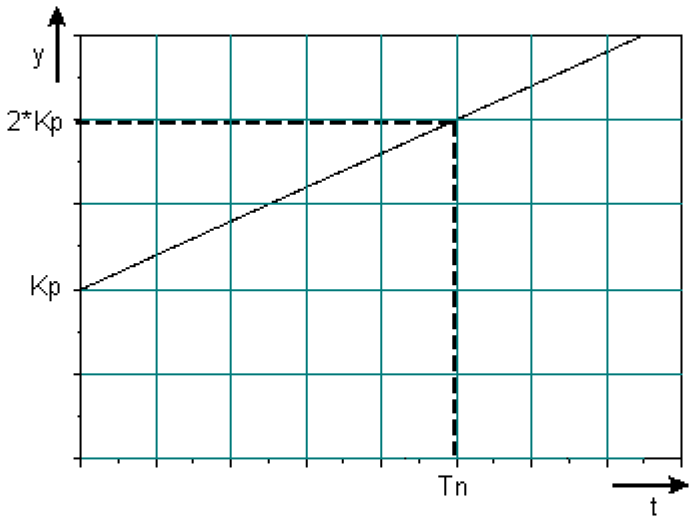


The function block provides a PI transfer element in the functional diagram.

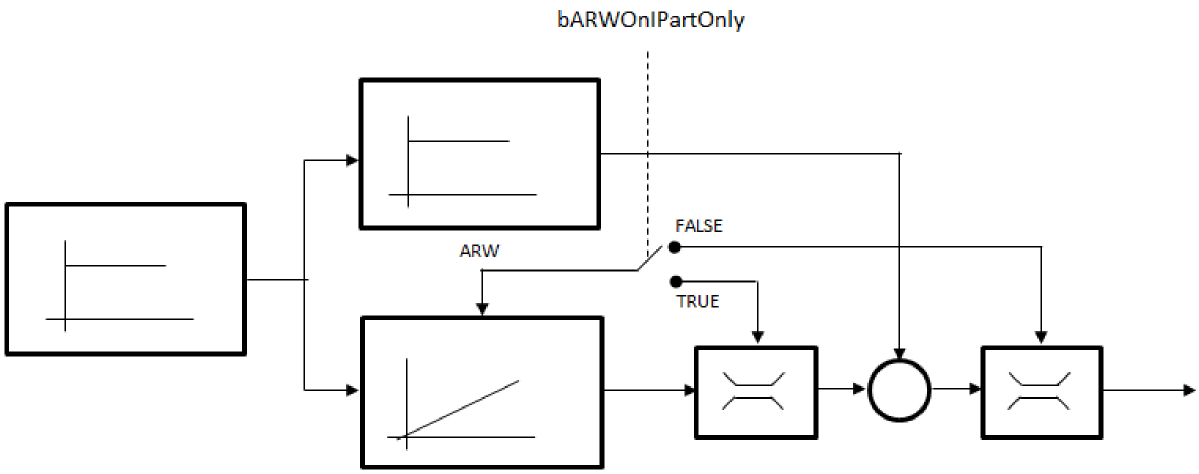
Behavior of the output

$$G(s) = K_p \left(1 + \frac{1}{T_n s}\right)$$

Step response



ARW



Inputs

```
VAR_INPUT
    fSetpointValue    : FLOAT;
    fActualValue      : FLOAT;
    fManSyncValue     : FLOAT;
    bSync             : BOOL;
    eMode             : E_CTRL_MODE;
    bHold             : BOOL;
END_VAR
```

Name	Type	Description
fSetpointValue	FLOAT	Setpoint of the controlled variable
fActualValue	FLOAT	Actual value of the controlled variable
fManSyncValue	FLOAT	Input with which the PI element can be set.
bSync	BOOL	A rising edge at this input sets the PI-element to the value "fManSyncValue".
eMode	E_CTRL_MODE	Input that specifies the operation mode [▸ 173] of the function block.
bHold	BOOL	A TRUE at this input will hold the internal state (and therefore also the output) constant at its current value, independently of the control deviation.

Outputs

```
VAR_OUTPUT
    fOut            : FLOAT;
    bARWactive      : BOOL;
    eState          : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError          : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output of the PI element
bARWactive	BOOL	A TRUE at this output indicates that the PI-element is being restricted.
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ ERRORCODES	Supplies the error number [▸ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams          : ST_CTRL_PI_PARAMS;
END_VAR
```

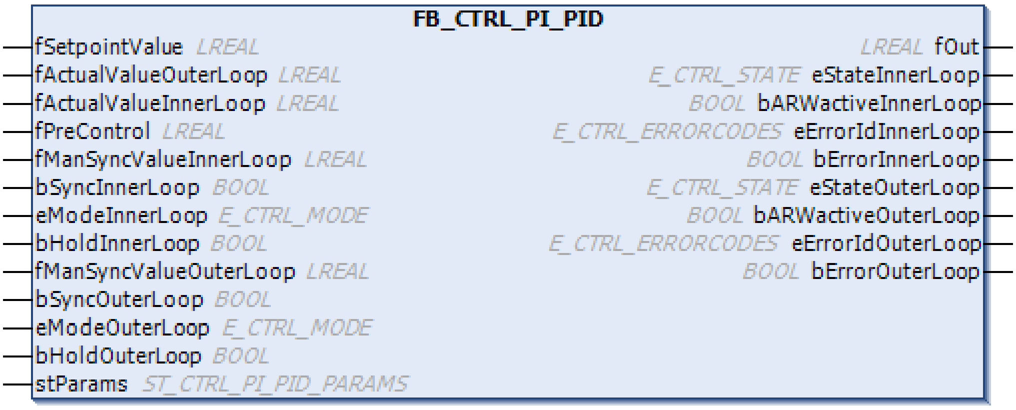
Name	Type	Description
stParams	ST_CTRL_PI_ PARAMS	Parameter structure of the PI element

`stParams` consists of the following elements:

```
TYPE ST_CTRL_PI_PARAMS:
STRUCT
    tCtrlCycleTime     : TIME := T#0ms;
    tTaskCycleTime     : TIME := T#0ms;
    tTn                : TIME := T#0ms;
    fKp                : FLOAT := 0;
    fOutMaxLimit       : FLOAT := 1E38;
    fOutMinLimit       : FLOAT := -1E38;
    bARWOnIPartOnly    : BOOL := FALSE;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
tTn	TIME	Integral action time
fKp	FLOAT	Controller amplification / transfer coefficient
fOutMaxLimit	FLOAT	Upper limit at which integration is halted and to which the output is limited (ARW measure). Reaching this limit is indicated by a TRUE at the `bARWActive` output.
fOutMinLimit	FLOAT	Lower limit at which integration is halted and to which the output is limited (ARW measure). Reaching this limit is indicated by a TRUE at the `bARWActive` output.
bARWOnIPartOnly	BOOL	If this parameter is FALSE (the standard setting), the integration of the I-component is halted if the complete controller output reaches the upper or lower limit. If it is TRUE, the integration is halted if the I-component (the output of the integrator) reaches some limit. (Cf. functional diagram.)

4.2.1.3.8 FB_CTRL_PI_PID



The function block provides a cascaded PI-PID controller in the functional diagram. Internally, this function block uses the `FB_CTRL_PI`, `FB_CTRL_LIMITER` and `FB_CTRL_PID` transfer elements.

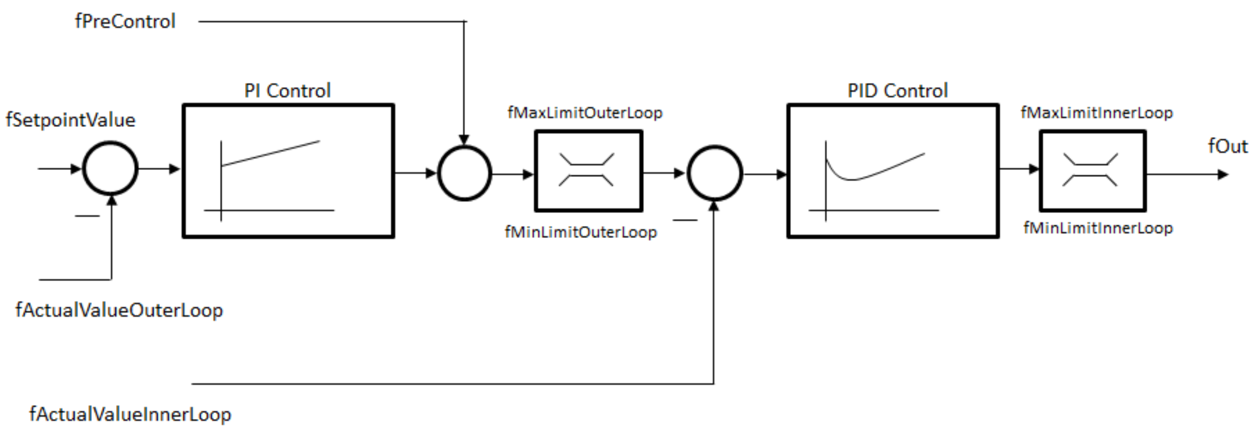
Transfer function of the PI element

$$G(s) = K_p \left(1 + \frac{1}{T_n s}\right)$$

Transfer function of the PID-element

$$G(s) = K_p \left(1 + \frac{1}{T_n s} + \frac{T_v s}{1 + T_d s}\right)$$

Functional diagram for the cascaded transfer element



Inputs

```
VAR_INPUT
    fSetpointValue            : FLOAT;
    fActualValueOuterLoop     : FLOAT;
    fActualValueInnerLoop     : FLOAT;
    fPreControl               : FLOAT;
    fManSyncValueInnerLoop    : FLOAT;
    bSyncInnerLoop            : BOOL;
    eModeInnerLoop            : E_CTRL_MODE;
    bHoldInnerLoop            : BOOL;
    fManSyncValueOuterLoop    : FLOAT;
    bSyncOuterLoop            : BOOL;
    eModeOuterLoop            : E_CTRL_MODE;
    bHoldOuterLoop            : BOOL;
END_VAR
```

Name	Type	Description
fSetpointValue	FLOAT	Setpoint of the controlled variable
fActualValueOuterLoop	FLOAT	The actual value of the controlled variable that is fed back to the PI controller of the outer control loop.
fActualValueInnerLoop	FLOAT	The actual value of the controlled variable that is fed back to the PID controller of the inner control loop.
fPreControl	FLOAT	Pre-control that is connected behind the PI controller.
fManSyncValueInnerLoop	FLOAT	Input, to whose value it is possible to set the internal state of the PID-element (the inner control loop).
bSyncInnerLoop	BOOL	A rising edge at this input sets the PID-element (the inner control loop) to the value "fManSyncValueInnerLoop".
eModeInnerLoop	E_CTRL_MODE	Input that specifies the operation mode [▶ 173] of the PID-element (the inner control loop).
bHoldInnerLoop	BOOL	A TRUE at this input holds the internal state of the PID-element (the inner control loop) constant at the current value.
fManSyncValueOuterLoop	FLOAT	Input, to whose value it is possible to set the internal state of the PI element (the outer control loop).
bSyncOuterLoop	BOOL	A rising edge at this input sets the PI element (the outer control loop) to the value "fManSyncValueOuterLoop".
eModeOuterLoop	E_CTRL_MODE	Input that specifies the operation mode [▶ 173] of the PI element (the outer control loop).
bHoldOuterLoop	BOOL	A TRUE at this input holds the internal state of the PI element (the outer control loop) constant at the current value.

Outputs

```
VAR_OUTPUT
    fOut                    : FLOAT;
    eStateInnerLoop         : E_CTRL_STATE;
    bARWactiveInnerLoop     : BOOL;
    eErrorIdInnerLoop       : E_CTRL_ERRORCODES;
    bErrorInnerLoop         : BOOL;
    eStateOuterLoop         : E_CTRL_STATE;
    bARWactiveOuterLoop     : BOOL;
    eErrorIdOuterLoop       : E_CTRL_ERRORCODES;
    bErrorOuterLoop         : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output of the PI-PID-element
eStateInnerLoop	E_CTRL_STATE	State of the internal PID-element (inner control loop)
bARWactive InnerLoop	BOOL	A TRUE at this output indicates that the output of the PID-element (the inner control loop) is being restricted.
eErrorIdInnerLoop	E_CTRL_ERRORCODES	Returns the error number [▸ 173] of the PID-element (the inner control loop) when the bError output is set.
bErrorInnerLoop	BOOL	Is set to TRUE as soon as an error occurs in the PID-element (the inner control loop).
eStateOuterLoop	E_CTRL_STATE	State of the internal PI element (outer control loop)
bARWactiveOuterLoop	BOOL	A TRUE at this output indicates that the PI element's (the outer control loop) output is being restricted.
eErrorIdOuterLoop	E_CTRL_ERRORCODES	Returns the error number [▸ 173] of the PI element (the outer control loop) when the bError output is set.
bErrorOuterLoop	BOOL	Is set to TRUE as soon as an error occurs in the PI element (the outer control loop).

Inputs/ outputs

```
VAR_IN_OUT
    stParams         : ST_CTRL_PI_PID_PARAMS;
END_VAR
```

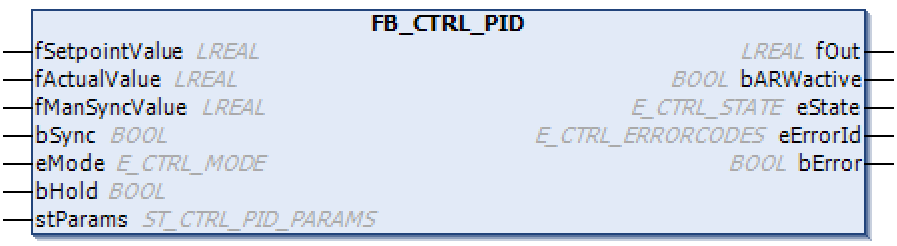
Name	Type	Description
stParams	ST_CTRL_PI_PID_PARAMS	Parameter structure of the PI-PID-element

stParams consists of the following elements:

```
TYPE
ST_CTRL_PI_PID_PARAMS :
STRUCT
    tCtrlCycleTime          : TIME := T#0ms;
    tTaskCycleTime          : TIME := T#0ms;
    fKp_OuterLoop           : FLOAT := 0;
    tTn_OuterLoop           : TIME := T#0s;
    fMaxLimit_OuterLoop     : FLOAT := 1E38;
    fMinLimit_OuterLoop     : FLOAT := -1E38;
    fKp_InnerLoop           : FLOAT := 0;
    tTn_InnerLoop           : TIME := T#0ms;
    tTv_InnerLoop           : TIME := T#0ms;
    tTd_InnerLoop           : TIME := T#0ms;
    fMaxLimit_InnerLoop     : FLOAT := 1E38;
    fMinLimit_InnerLoop     : FLOAT := -1E38;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
fKp_OuterLoop	FLOAT	Controller amplification / controller coefficient of the PI element (outer control loop)
tTn_OuterLoop	TIME	Integral action time of the internal PI element (outer control loop). The I-component is deactivated if this is parameterized as T#0s.
fMaxLimit_OuterLoop	FLOAT	Upper limit at which integration in the PID-element is halted and to which the output is limited (ARW measure). Reaching this limit is indicated by a TRUE at the `bARWactiveOuterLoop` output.
fMinLimit_OuterLoop	FLOAT	Lower limit at which integration in the PID-element is halted and to which the output is limited (ARW measure). Reaching this limit is indicated by a TRUE at the `bARWactiveOuterLoop` output.
fKp_InnerLoop	FLOAT	Controller amplification / controller coefficient of the PID element (inner control loop)
tTn_InnerLoop	TIME	Integral action time of the PID-element (inner control loop). The I-component is deactivated if this is parameterized as T#0s.
tTv_InnerLoop	TIME	Derivative action time of the PID-element (inner control loop). The D-component is deactivated if this is parameterized as T#0s.
tTd_InnerLoop	TIME	Damping time of the PID-element (inner control loop)
fMaxLimit_InnerLoop	FLOAT	Upper limit at which integration in the PID-element is halted and to which the output is limited (ARW measure). Reaching this limit is indicated by a TRUE at the `bARWactiveInnerLoop` output.
fMinLimit_InnerLoop	FLOAT	Lower limit at which integration in the PID-element is halted and to which the output is limited (ARW measure). Reaching this limit is indicated by a TRUE at the `bARWactiveInnerLoop` output.

4.2.1.3.9 FB_CTRL_PID



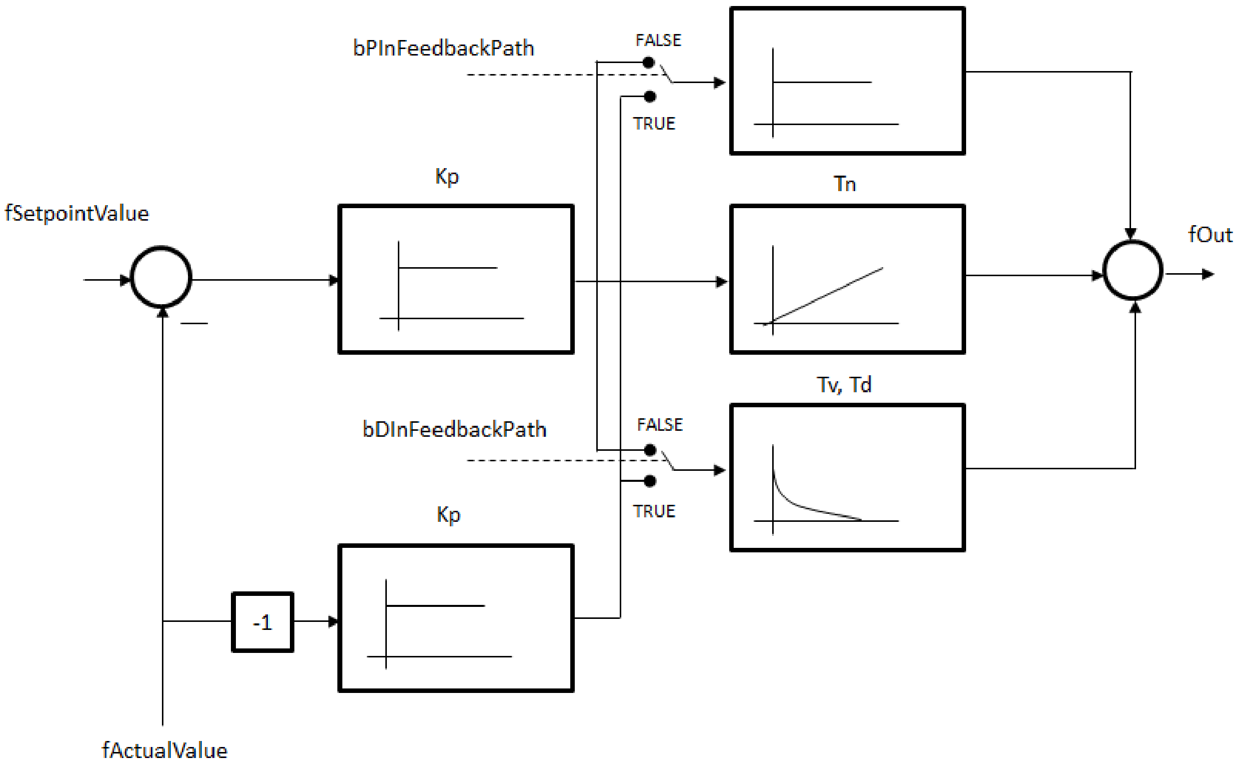
The function block provides a PID transfer element in the functional diagram.

Transfer function

The following transfer function can be specified if the boolean Inputs `bPInTheFeedbackPath` and `bDInTheFeedbackPath` are FALSE. Otherwise the transfer function only describes part of the transfer behavior of the function block.

$$G_{PID}(s) = K_p\left(1 + \frac{1}{T_n s} + \frac{T_v s}{1 + T_d s}\right)$$

Functional diagram



The standard functional diagram of a PID controller in additive form has been expanded by the two active boolean Inputs `bPInTheFeedbackPath` and `bDInTheFeedbackPath` (which act as "switches"), so that a modified functional diagram can be activated.

Control background: due to the differential component of the control algorithm, large control values are generated at setpoint step-changes, which cause a strain on the control elements and may cause the control system to oscillate. A control algorithm whose differential component is only applied to the control value (`bDInTheFeedbackPath := TRUE`) avoids this problem.

The `bPInTheFeedbackPath` and `bDInTheFeedbackPath` Inputs permit the closed control loop to implement the following transfer functions:

bPInTheFeedbackPath	bDInTheFeedbackPath	G(s)
false	false	$G(s) = \dfrac{G_{PID}(s) \cdot G_S(s)}{1 + G_{PID}(s) \cdot G_S(s)}$
true	false	$G(s) = \dfrac{G_{DI}(s) \cdot G_S(s)}{1 + G_{PID}(s) \cdot G_S(s)}$
false	true	$G(s) = \dfrac{G_{PI}(s) \cdot G_S(s)}{1 + G_{PID}(s) \cdot G_S(s)}$
true	true	$G(s) = \dfrac{G_I(s) \cdot G_S(s)}{1 + G_{PID}(s) \cdot G_S(s)}$

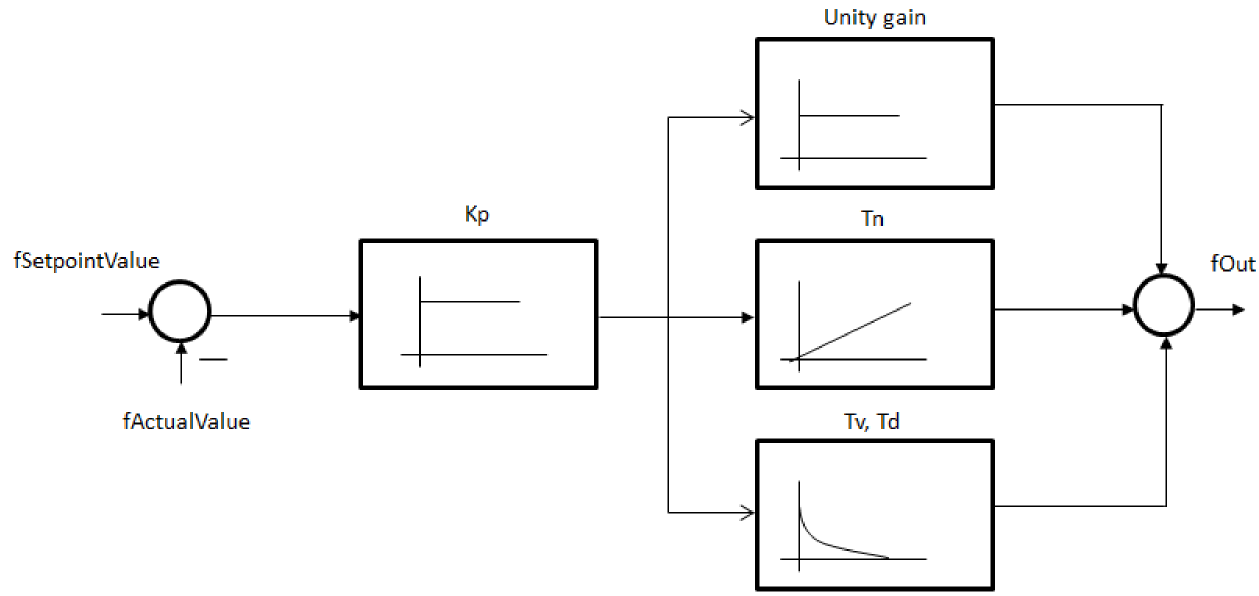
with:

$$G_{DI}(s) = K_p \left(\frac{1}{T_n s} + \frac{T_v s}{1 + T_d s} \right)$$

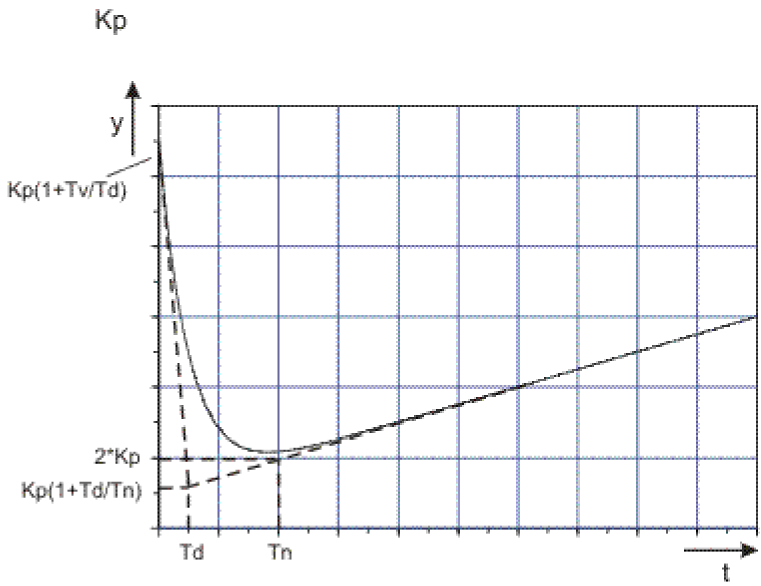
$$G_{PI}(s) = K_p\left(1+\frac{1}{T_n s}\right)$$

$$G_I(s) = K_p\left(\frac{1}{T_n s}\right)$$

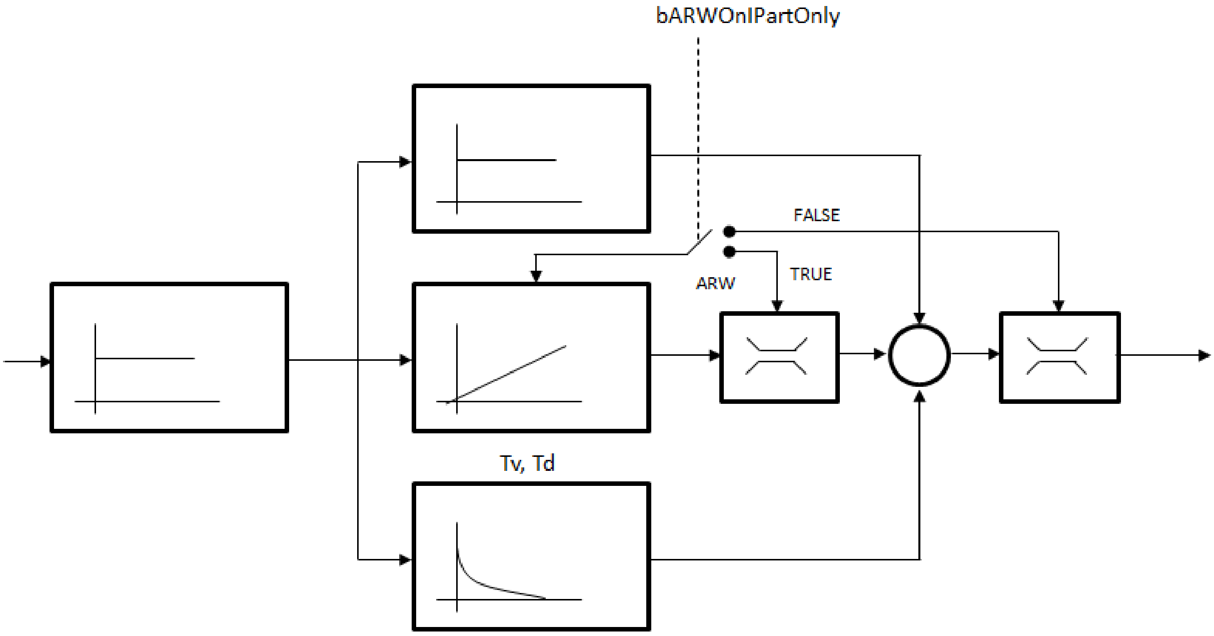
The standard setting for the two `bPInTheFeedbackPath` and `bDInTheFeedbackPath` Inputs is FALSE. The PID controller then corresponds to a standard PID controller in additive form.



Step response



ARW



Inputs

```
VAR_INPUT
    fSetpointValue     : FLOAT;
    fActualValue       : FLOAT;
    fManSyncValue      : FLOAT;
    bSync              : BOOL;
    eMode              : E_CTRL_MODE;
    bHold              : BOOL;
END_VAR
```

Name	Type	Description
fSetpointValue	FLOAT	Setpoint of the controlled variable
fActualValue	FLOAT	Actual value of the controlled variable
fManSyncValue	FLOAT	Input, to whose value it is possible to set the internal state of the PID-element.
bSync	BOOL	A rising edge at this input sets the PID-element to the value "fManSyncValue".
eMode	E_CTRL_MODE	Input that specifies the operation mode [▶ 173] of the function block.
bHold	BOOL	A TRUE at this input will hold the internal state (and therefore also the output) constant at its current value, independently of the control deviation.

Outputs

```
VAR_OUTPUT
    fOut           : FLOAT;
    bARWactive     : BOOL;
    eState         : E_CTRL_STATE;
    eErrorId       : E_CTRL_ERRORCODES;
    bError         : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output of the PID element
bARWactive	BOOL	A TRUE at this output indicates that the PID-element is being restricted.
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▶ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams         : ST_CTRL_PID_PARAMS;
END_VAR
```

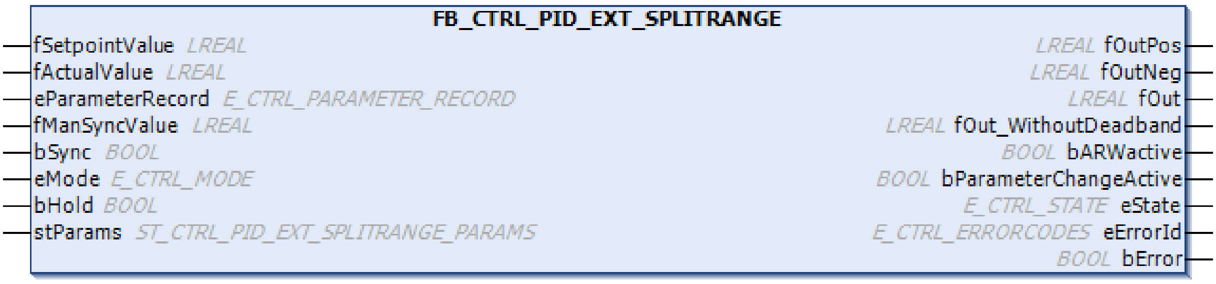
Name	Type	Description
stParams	ST_CTRL_PID_PARAMS	Parameter structure of the PID element

`stParams` consists of the following elements:

```
TYPE
ST_CTRL_PID_PARAMS :
STRUCT
    tCtrlCycleTime          : TIME := T#0ms;
    tTaskCycleTime          : TIME := T#0ms;
    fKp                     : FLOAT := 0;
    tTn                     : TIME := T#0ms;
    tTv                     : TIME := T#0ms;
    tTd                     : TIME := T#0ms;
    fOutMaxLimit            : FLOAT := 1E38;
    fOutMinLimit            : FLOAT := -1E38;
    bPInTheFeedbackPath     : BOOL;
    bDInTheFeedbackPath     : BOOL;
    bARWOnIPartOnly         : BOOL;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
fKp	FLOAT	Controller amplification / controller coefficient
tTn	TIME	Integral action time; if this is parameterized to T#0s, the I component is deactivated.
tTv	TIME	Derivative action time; if this is parameterized to T#0s, the D component is deactivated.
tTd	TIME	Damping time
fOutMaxLimit	FLOAT	Upper limit at which integration is halted and to which the output is limited (ARW measure). Reaching this limit is indicated by a TRUE at the `bARWActive` output.
fOutMinLimit	FLOAT	Lower limit at which integration is halted and to which the output is limited (ARW measure). Reaching this limit is indicated by a TRUE at the `bARWActive` output.
bPInTheFeedbackPath	BOOL	Input value of the internal P-element can be selected with this input (see functional diagram). Standard setting: FALSE
bDInTheFeedbackPath	BOOL	Input value of the internal D-element can be selected with this input (see functional diagram). Standard setting: FALSE
bARWOnIPartOnly	BOOL	If this parameter is FALSE (the standard setting), the integration of the I-component is halted if the complete controller output reaches the upper or lower limit. If it is TRUE, the integration is halted if the I-component (the output of the integrator) reaches some limit. (Cf. functional diagram.)

4.2.1.3.10 FB_CTRL_PID_EXT_SPLITRANGE



The function block provides an extended PID transfer element in the functional diagram. With this controller it is possible to switch between two different parameter sets while the regulation is active. The functionalities of the inner and outer windows and of the input and output dead bands are available in addition.

Description

This function block is an extension of `FB_CTRL_PID_EXT`, which means that the controller can be used to control systems with two controlled devices for which the transfer behavior are different. A system with one actuator for heating and another actuator for cooling would be a typical application. To optimize the regulation of such an arrangement, it is possible to switch between two PID parameter sets. The parameter set changeover is carried out in such a way that a continuous control value is maintained even during the changeover.

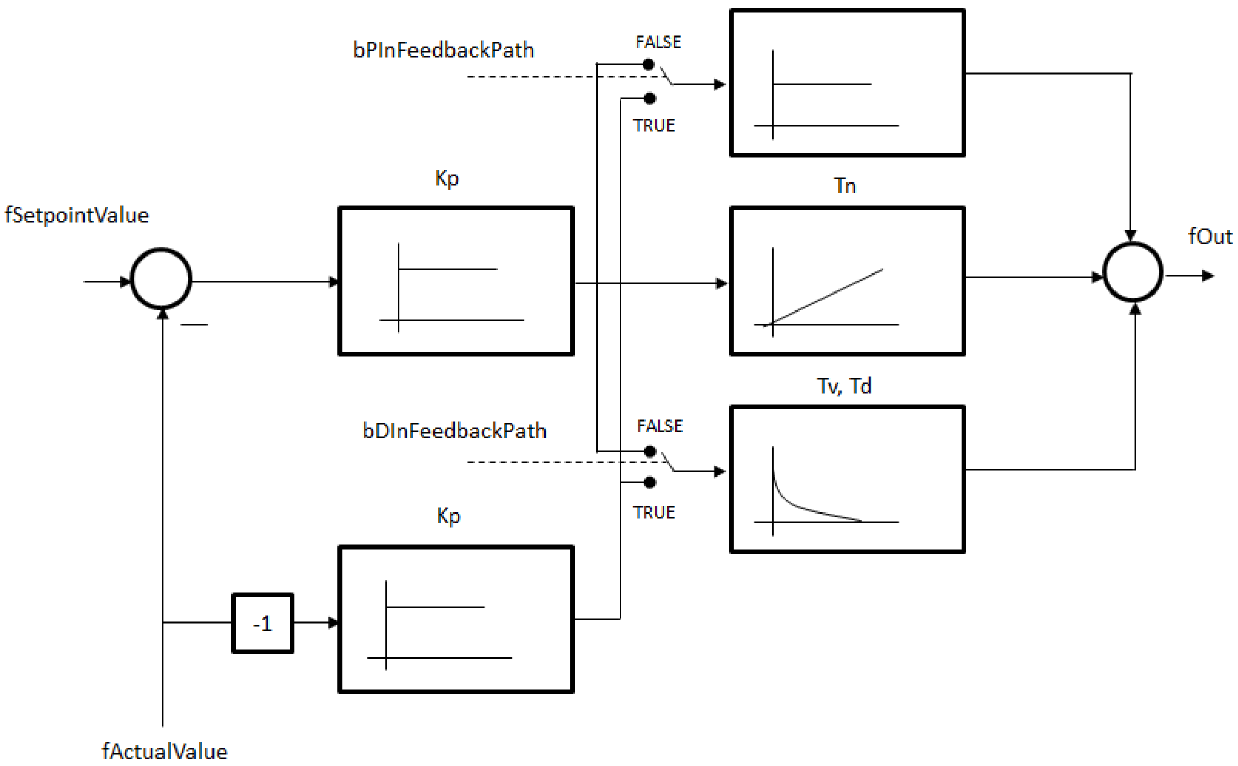
The switching algorithm calculates a linear, time-dependent transition between the two parameter sets. The `nParameterChangeCycleTicks` parameter can be used to specify the number of task cycles over which the continuous change between the two parameter sets takes place.

Transfer function

The following transfer function can be specified if the boolean Inputs `bPInTheFeedbackPath` and `bDInTheFeedbackPath` are FALSE. Otherwise the transfer function only describes part of the transfer behavior of the function block.

$$G_{PID}(s) = K_p \left(1 + \frac{1}{T_n s} + \frac{T_v s}{1 + T_d s} \right)$$

Functional diagram



The standard functional diagram of a PID controller in additive form has been expanded by the two active boolean Inputs `bPInTheFeedbackPath` and `bDInTheFeedbackPath` (which act as "switches"), so that a modified functional diagram can be activated.

Control background: due to the differential component of the control algorithm, large control values are generated at setpoint step-changes, which cause a strain on the control elements and may cause the control system to oscillate. A control algorithm with a differential component that is only applied to the controlled variable (`bDInTheFeedbackPath := TRUE`) can avoid this problem.

The `bPInTheFeedbackPath` and `bDInTheFeedbackPath` Inputs permit the closed control loop to implement the following transfer functions:

bPInTheFeedbackPath	bDInTheFeedbackPath	G(s)
false	false	$G(s) = \frac{G_{PID}(s) \cdot G_S(s)}{1 + G_{PID}(s) \cdot G_S(s)}$
true	false	$G(s) = \frac{G_{DI}(s) \cdot G_S(s)}{1 + G_{PID}(s) \cdot G_S(s)}$
false	true	$G(s) = \frac{G_{PI}(s) \cdot G_S(s)}{1 + G_{PID}(s) \cdot G_S(s)}$
true	true	$G(s) = \frac{G_I \cdot G_S(s)}{1 + G_{PID}(s) \cdot G_S(s)}$

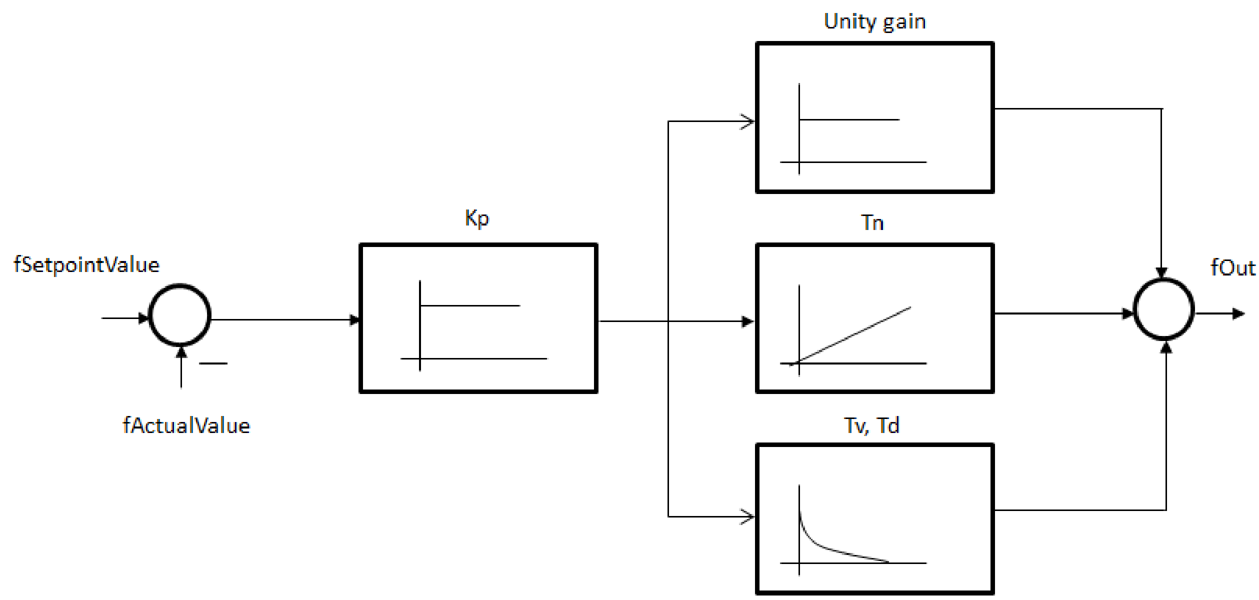
with:

$$G_{DI}(s) = K_p \left(\frac{1}{T_n s} + \frac{T_v s}{1 + T_d s} \right)$$

$$G_{PI}(s) = K_p \left(1 + \frac{1}{T_n s} \right)$$

$$G_I(s) = K_p \left(\frac{1}{T_n s} \right)$$

The standard setting for the two `bPInTheFeedbackPath` and `bDInTheFeedbackPath` Inputs is FALSE. The PID controller then acts as a standard PID controller in additive form.



Additional functions

Switching off the I-component in the Outer Window

Integration of the control deviation is halted if the control deviation is greater than the `fOuterWindow` parameter. In this way it is possible to prevent an extremely large I-component from developing if the control deviation is large, since this could lead to a marked overshoot. If it is not wanted, the function can be disabled by setting `fOuterWindow:= 0`.

Linear reduction of the I-component in the Inner Window

With this function it is possible to drive the I-component linearly down to zero in the range specified by the `fInnerWindow` parameter. If it is not wanted, the function can be disabled by setting `fInnerWindow := 0`.

Output dead band

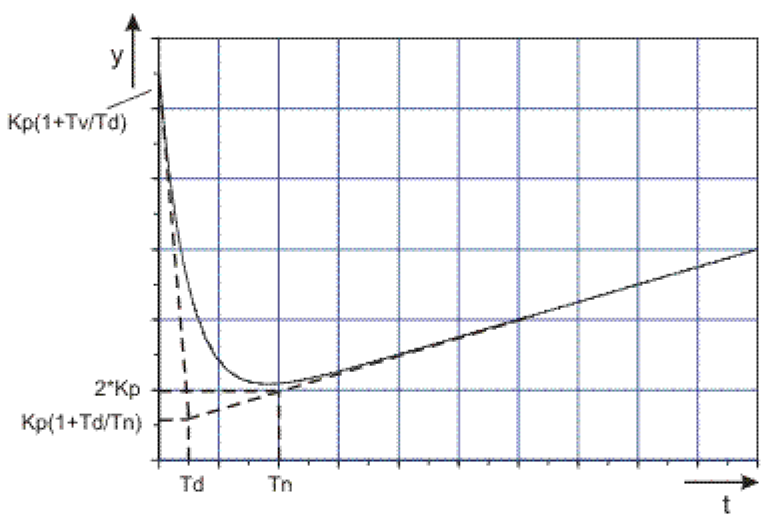
If the parameter `fDeadBandOutput > 0` is set, the output is set to zero when it is within the range of `[ -fDeadBandOutput ... fDeadBandOutput ]`.

Input dead band

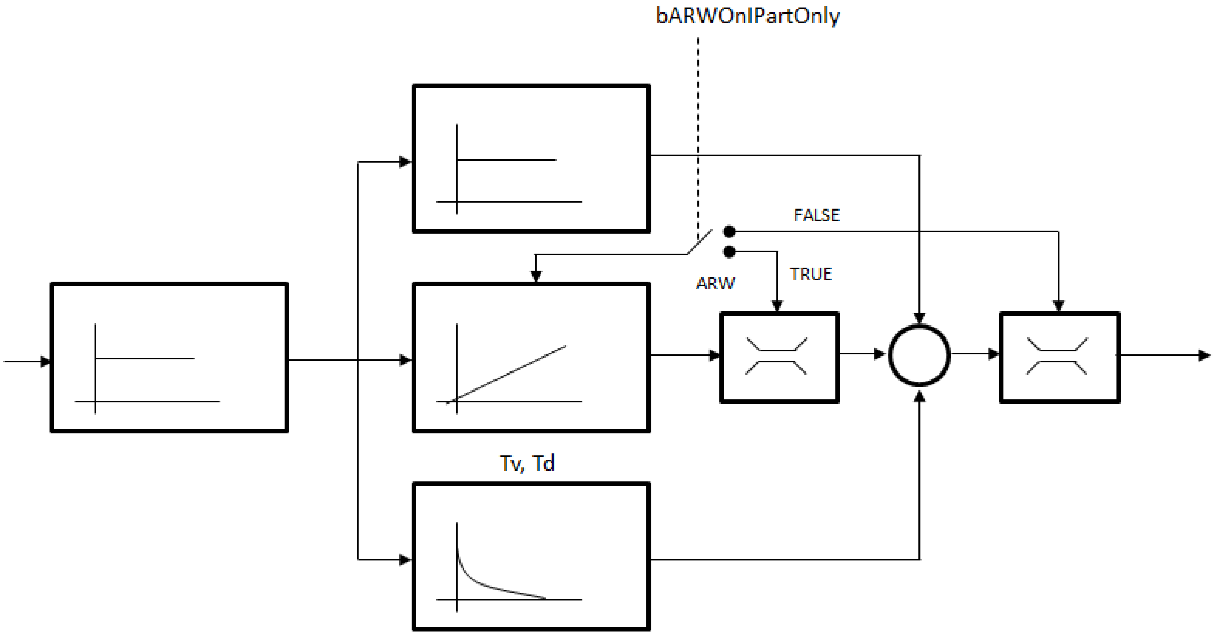
If the parameter `fDeadBandInput > 0` is set then the output is held constant for as long as the control deviation remains within the range of `[ -fDeadBandInput ... fDeadBandInput ]`.

Step response

Kp



ARW



Inputs

```
VAR_INPUT
    fSetpointValue        : FLOAT;
    fActualValue          : FLOAT;
    eParameterRecord      : E_CTRL_PARAMETER_RECORD;
    fManSyncValue         : FLOAT;
    bSync                 : BOOL;
    eMode                 : E_CTRL_MODE;
    bHold                 : BOOL;
END_VAR
```

Name	Type	Description
fSetpointValue	FLOAT	Setpoint of the controlled variable
fActualValue	FLOAT	Actual value of the controlled variable
eParameterReco rd	E_CTRL_PARAM ETER_RECORD	Index of the active parameter set
fManSyncValue	FLOAT	Input with which the PI element can be set.
bSync	BOOL	A rising edge at this input sets the PI-element to the value "fManSyncValue".
eMode	E_CTRL_MODE	Input that specifies the operation mode [▶ 173] of the function block.
bHold	BOOL	A TRUE at this input will hold the internal state (and therefore also the output) constant at its current value, independently of the control deviation.

Outputs

```
VAR_OUTPUT
    fOutPos                     : FLOAT;
    fOutNeg                     : FLOAT;
    fOut                        : FLOAT;
    bARWActive                  : BOOL := FALSE;
    bParameterChangeActive      : BOOL;
    bError                      : BOOL;
    eErrorId                    : E_CTRL_ERRORCODES;
END_VAR
```

Name	Type	Description
fOutPos	FLOAT	Output of the PID-element when the control value is positive. A zero is output otherwise.
fOutNeg	FLOAT	Output of the PID-element when the control value is negative. A zero is output otherwise.
fOut	FLOAT	Output of the PID element
bARWActive	BOOL	A TRUE at this output indicates that the PID-element is being restricted.
bParameter ChangeActive	BOOL	A TRUE at this output indicates that the change from one parameter set to the other is in progress.
bError	BOOL	Becomes TRUE, as soon as an error occurs.
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▸ 173] when the output `bError` is set.
fCtrlDerivation		A TRUE at this output indicates that the PID-element is being restricted.

Inputs/ outputs

```
VAR_IN_OUT
    stParams    : ST_CTRL_PID_EXT_SPLITRANGE_PARAMS;
END_VAR
```

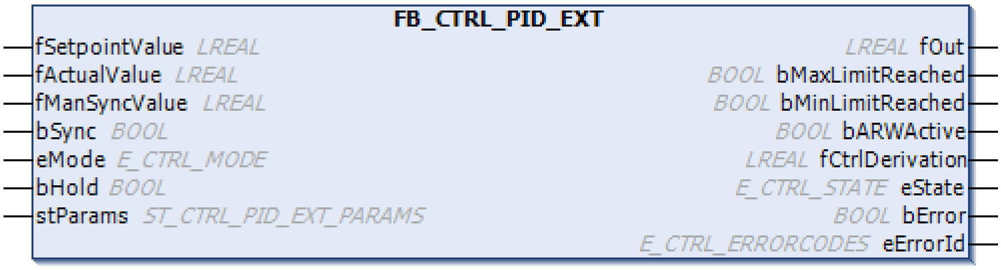
Name	Type	Description
stParams	ST_CTRL_ PID_EXT_SPLIT RANGE_PARAMS	Parameter structure of the PID element

`stParams` consists of the following elements:

```
TYPE
ST_CTRL_PID_EXT_SPLITRANGE_PARAMS :
STRUCT
    tCtrlCycleTime              : TIME := T#0ms;
    tTaskCycleTime              : TIME := T#0ms;
    fKp_heating                 : FLOAT := 0;
    tTn_heating                 : TIME := T#0ms;
    tTv_heating                 : TIME := T#0ms;
    tTd_heating                 : TIME := T#0ms;
    fKp_cooling                 : FLOAT := 0;
    tTn_cooling                 : TIME := T#0ms;
    tTv_cooling                 : TIME := T#0ms;
    tTd_cooling                 : TIME := T#0ms;
    nParameterChangeCycleTicks  : INT;
    fDeadBandInput              : REAL := 0.0;
    fDeadBandOutput             : REAL := 0.0;
    fInnerWindow                : REAL := 0.0;
    fOuterWindow                : REAL := 0.0;
    fOutMaxLimit                : FLOAT := 1E38;
    fOutMinLimit                : FLOAT := -1E38;
    bPInTheFeedbackPath         : BOOL;
    bDInTheFeedbackPath         : BOOL;
    bARWOnIPartOnly             : BOOL;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
Range **eCTRL_PARAMETER_RECORD_HEATING**:		
fKp_heating	FLOAT	Controller amplification / controller coefficient
tTn_heating	TIME	Integral action time: The I-component is deactivated if this is parameterized as T#0s.
tTv_heating	TIME	Derivative action time: The D-component is deactivated if this is parameterized as T#0s.
tTd_heating	TIME	Damping time
Range **eCTRL_PARAMETER_RECORD_COOLING:**		
fKp_cooling	FLOAT	Controller amplification / controller coefficient
tTn_cooling	TIME	Integral action time: The I-component is deactivated if this is parameterized as T#0s.
tTv_cooling	TIME	Derivative action time: The D-component is deactivated if this is parameterized as T#0s.
tTd_cooling	TIME	Damping time
nParameter ChangeCycle Ticks	INT	The number of task cycles over which the change from one parameter set to the other takes place.
fDeadBandInput	REAL	See description above: Input dead band
fDeadBandOutpu t	REAL	See description above: Output dead band
fInnerWindow	REAL	See description above: Linear reduction of the I-component in the Inner Window
fOuterWindow	REAL	See description above: Switching off the I-component in the Outer Window
fOutMaxLimit	FLOAT	Upper limit at which integration is halted and to which the output is limited (ARW measure). Reaching this limit is indicated by a TRUE at the `bARWActive` output.
fOutMinLimit	FLOAT	Lower limit at which integration is halted and to which the output is limited (ARW measure). Reaching this limit is indicated by a TRUE at the `bARWActive` output.
bPInTheFeed backPath	BOOL	Input value of the P-element can be selected with this input (see functional diagram).
bDInTheFeed backPath	BOOL	Input value of the D-element can be selected with this input (see functional diagram).
bARWOnIPart Only	BOOL	If this parameter is FALSE (the standard setting), the integration of the I-component is halted if the complete controller output reaches the upper or lower limit. If it is TRUE, the integration is halted if the I-component (the output of the integrator) reaches some limit. (Cf. functional diagram.)

4.2.1.3.11 FB_CTRL_PID_EXT

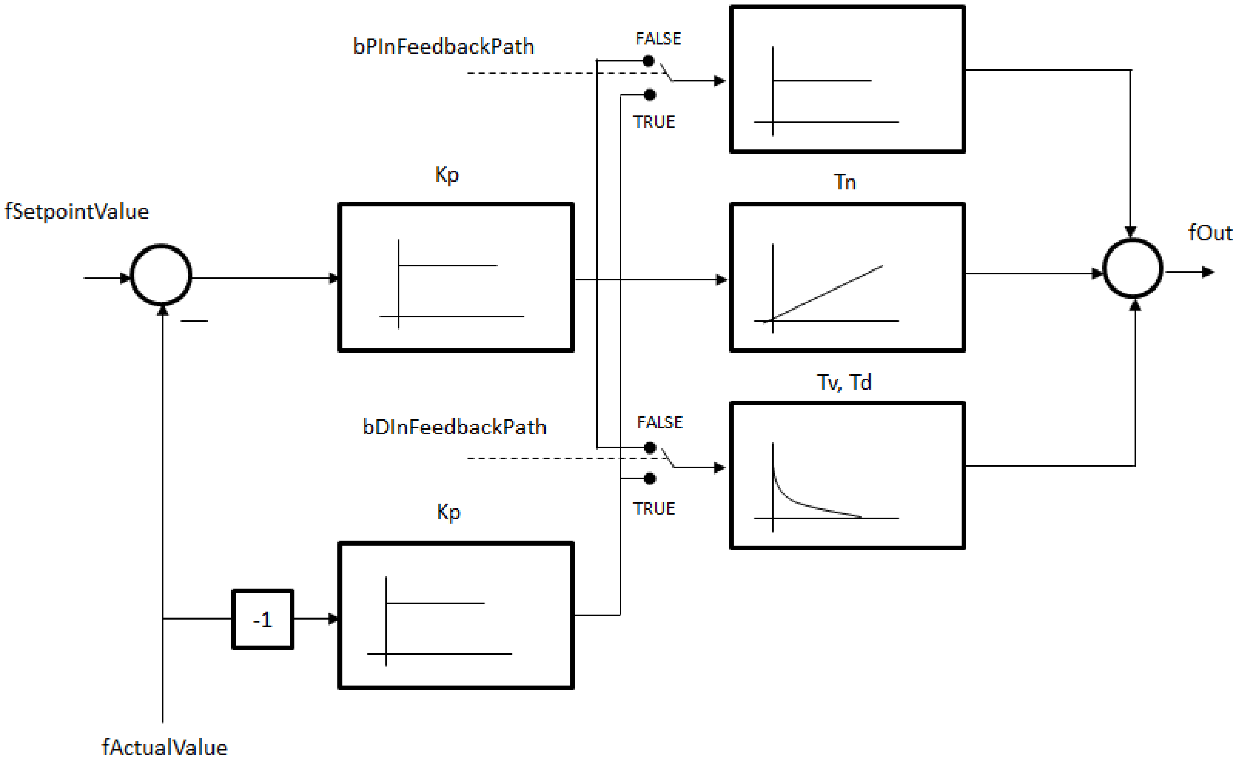


Transfer function

The following transfer function can be specified if the boolean Inputs `bPInTheFeedbackPath` and `bDInTheFeedbackPath` are FALSE. Otherwise the transfer function only describes part of the transfer behavior of the function block.

$$G_{PID}(s) = K_p \left(1 + \frac{1}{T_n s} + \frac{T_v s}{1 + T_d s}\right)$$

Functional diagram



The standard functional diagram of a PID controller in additive form has been expanded by the two active boolean Inputs `bPInTheFeedbackPath` and `bDInTheFeedbackPath` (which act as "switches"), so that a modified functional diagram can be activated.

Control background: due to the differential component of the control algorithm, large control values are generated at setpoint step-changes, which cause a strain on the control elements and may cause the control system to oscillate. A control algorithm with a differential component that is only applied to the controlled variable (`bDInTheFeedbackPath := TRUE`) can avoid this problem.

The `bPInTheFeedbackPath` and `bDInTheFeedbackPath` Inputs permit the closed control loop to implement the following transfer functions:

bPInTheFeedbackPath	bDInTheFeedbackPath	G(s)
false	false	$G(s) = \frac{G_{PID}(s) \cdot G_S(s)}{1 + G_{PID}(s) \cdot G_S(s)}$
true	false	$G(s) = \frac{G_{DI}(s) \cdot G_S(s)}{1 + G_{PID}(s) \cdot G_S(s)}$
false	true	$G(s) = \frac{G_{PI}(s) \cdot G_S(s)}{1 + G_{PID}(s) \cdot G_S(s)}$
true	true	$G(s) = \frac{G_I(s) \cdot G_S(s)}{1 + G_{PID}(s) \cdot G_S(s)}$

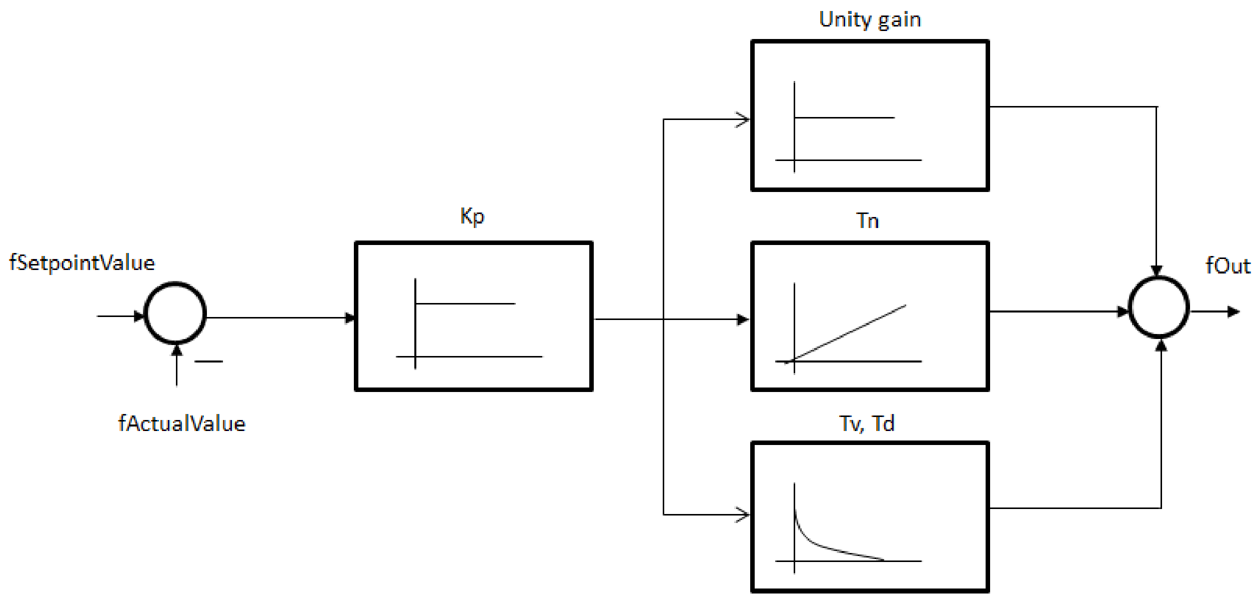
with:

$$G_{DI}(s) = K_p \left(\frac{1}{T_n s} + \frac{T_v s}{1 + T_d s} \right)$$

$$G_{PI}(s) = K_p \left(1 + \frac{1}{T_n s} \right)$$

$$G_I(s) = K_p \left(\frac{1}{T_n s} \right)$$

The standard setting for the two `bPInTheFeedbackPath` and `bDInTheFeedbackPath` Inputs is **FALSE**. The PID controller then acts as a standard PID controller in additive form.



Switching off the I-component in the Outer Window

Integration of the control deviation is halted if the control deviation is greater than the `fOuterWindow` parameter. In this way it is possible to prevent an extremely large I-component from developing if the control deviation is large, since this could lead to a marked overshoot. If it is not wanted, the function can be disabled by setting `fOuterWindow := 0`.

Linear reduction of the I-component in the Inner Window

With this function it is possible to drive the I-component linearly down to zero in the range specified by the `fInnerWindow` parameter. If it is not wanted, the function can be disabled by setting `fInnerWindow := 0`.

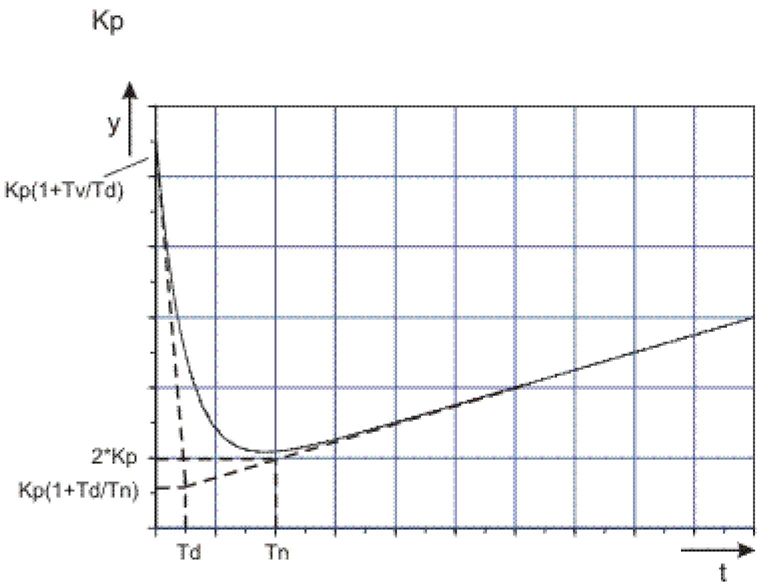
Output dead band

If the parameter `fDeadBandOutput > 0` is set, the output is set to zero when it is within the range of `[ -fDeadBandOutput ... fDeadBandOutput ]`.

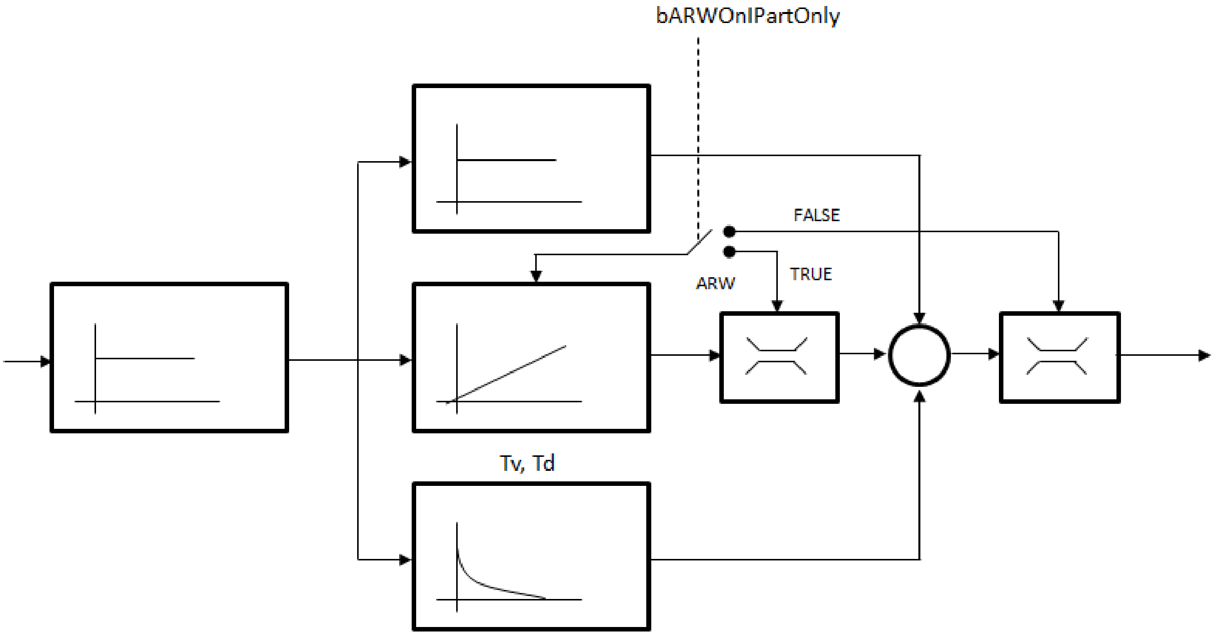
Input dead band

If the parameter `fDeadBandInput > 0` is set then the output is held constant for as long as the control deviation remains within the range of `[ -fDeadBandInput ... fDeadBandInput ]`.

Step response



ARW



Inputs

```
VAR_INPUT
    fSetpointValue    : FLOAT;
    fActualValue      : FLOAT;
    fManSyncValue     : FLOAT;
    bSync             : BOOL;
    eMode             : E_CTRL_MODE;
    bHold             : BOOL;
END_VAR
```

Name	Type	Description
fSetpointValue	FLOAT	Setpoint of the controlled variable
fActualValue	FLOAT	Actual value of the controlled variable
fManSyncValue	FLOAT	Input with which the PID-element can be set.
bSync	BOOL	A rising edge at this input sets the PID-element to the value "fManSyncValue".
eMode	E_CTRL_MODE	Input that specifies the operation mode [▶ 173] of the function block.
bHold	BOOL	A TRUE at this input will hold the internal state (and therefore also the output) constant at its current value, independently of the control deviation.

Outputs

```
VAR_OUTPUT
    fOut                : FLOAT;
    bMaxLimitReached    : BOOL := FALSE;
    bMinLimitReached    : BOOL := FALSE;
    bARWActive          : BOOL := FALSE;
    fCtrlDerivation     : FLOAT;
    eState              : E_CTRL_STATE;
    bError              : BOOL;
    eErrorId            : E_CTRL_ERRORCODES;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output of the PID element
bMaxLimit Reached	BOOL	The output is TRUE when the function block is at its upper limit.
bMinLimit Reached	BOOL	The output is TRUE when the function block is at its lower limit.
bARWActive	BOOL	A TRUE at this output indicates that the PID-element is being restricted.
fCtrlDerivation	FLOAT	The current value of the control deviation
eState	E_CTRL_STATE	State of the function block
bError	BOOL	Supplies the error number [▸ 173] when the output `bError` is set.
eErrorId	E_CTRL_ ERRORCODES	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams     : ST_CTRL_PID_EXT_PARAMS;
END_VAR
```

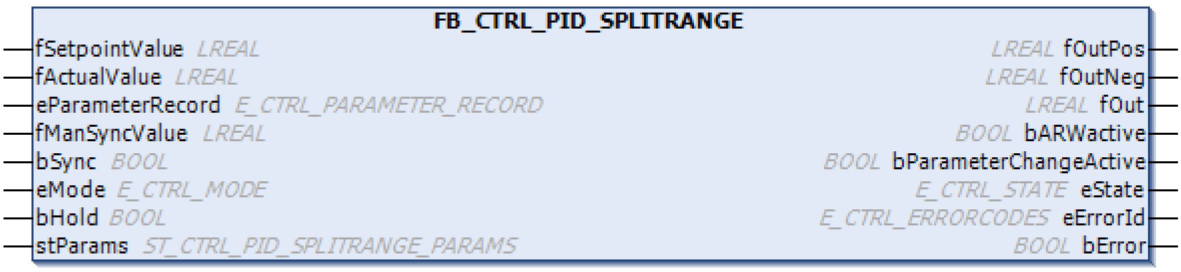
Name	Type	Description
stParams	ST_CTRL_PID_ EXT_PARAMS	Parameter structure of the PID element

`stParams` consists of the following elements:

```
TYPE
ST_CTRL_PID_EXT_PARAMS :
STRUCT
    tCtrlCycleTime          : TIME := T#0ms;
    tTaskCycleTime          : TIME := T#0ms;
    fKp                     : FLOAT := 0;
    tTn                     : TIME := T#0ms;
    tTv                     : TIME := T#0ms;
    tTd                     : TIME := T#0ms;
    fDeadBandInput          : REAL := 0.0;
    fDeadBandOutput         : REAL := 0.0;
    fInnerWindow            : REAL := 0.0;
    fOuterWindow            : REAL := 0.0;
    fOutMaxLimit            : FLOAT := 1E38;
    fOutMinLimit            : FLOAT := -1E38;
    bPInTheFeedbackPath     : BOOL;
    bDInTheFeedbackPath     : BOOL;
    bARWOnIPartOnly         : BOOL;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
fKp	FLOAT	Controller amplification / controller coefficient
tTn	TIME	Integral action time; if this is parameterized to T#0s, the I component is deactivated.
tTv	TIME	Derivative action time; if this is parameterized to T#0s, the D component is deactivated.
tTd	TIME	Damping time
fDeadBandInput	REAL	See description above.
fDeadBand Output	REAL	See description above.
fInnerWindow	REAL	See description above.
fOuterWindow	REAL	See description above.
fOutMaxLimit	FLOAT	Upper limit at which integration is halted and to which the output is limited (ARW measure). Reaching this limit is indicated by a TRUE at the `bARWActive` output.
fOutMinLimit	FLOAT	Lower limit at which integration is halted and to which the output is limited (ARW measure). Reaching this limit is indicated by a TRUE at the `bARWActive` output.
bPInTheFeed backPath	BOOL	Input value of the P-element can be selected with this input (see functional diagram).
bDInTheFeed backPath	BOOL	Input value of the D-element can be selected with this input (see functional diagram).
bARWOnIPart Only	BOOL	If this parameter is FALSE (the standard setting), the integration of the I-component is halted if the complete controller output reaches the upper or lower limit. If it is TRUE, the integration is halted if the I-component (the output of the integrator) reaches some limit. (Cf. functional diagram.)

4.2.1.3.12 FB_CTRL_PID_SPLITRANGE



The function block provides an extended PID transfer element in the functional diagram. This controller enables switching between two parameter sets when control is active.

Description

This function block is an extension of `FB_CTRL_PID`, which means that the controller can be used to control systems with two controlled devices for which the transfer behavior are different. A system with one actuator for heating and another actuator for cooling would be a typical application. To optimize the regulation of such an arrangement, it is possible to switch between two PID parameter sets. Parameter set switchover is implemented in such a way that the control value remains continuous even as the parameter sets are changed.

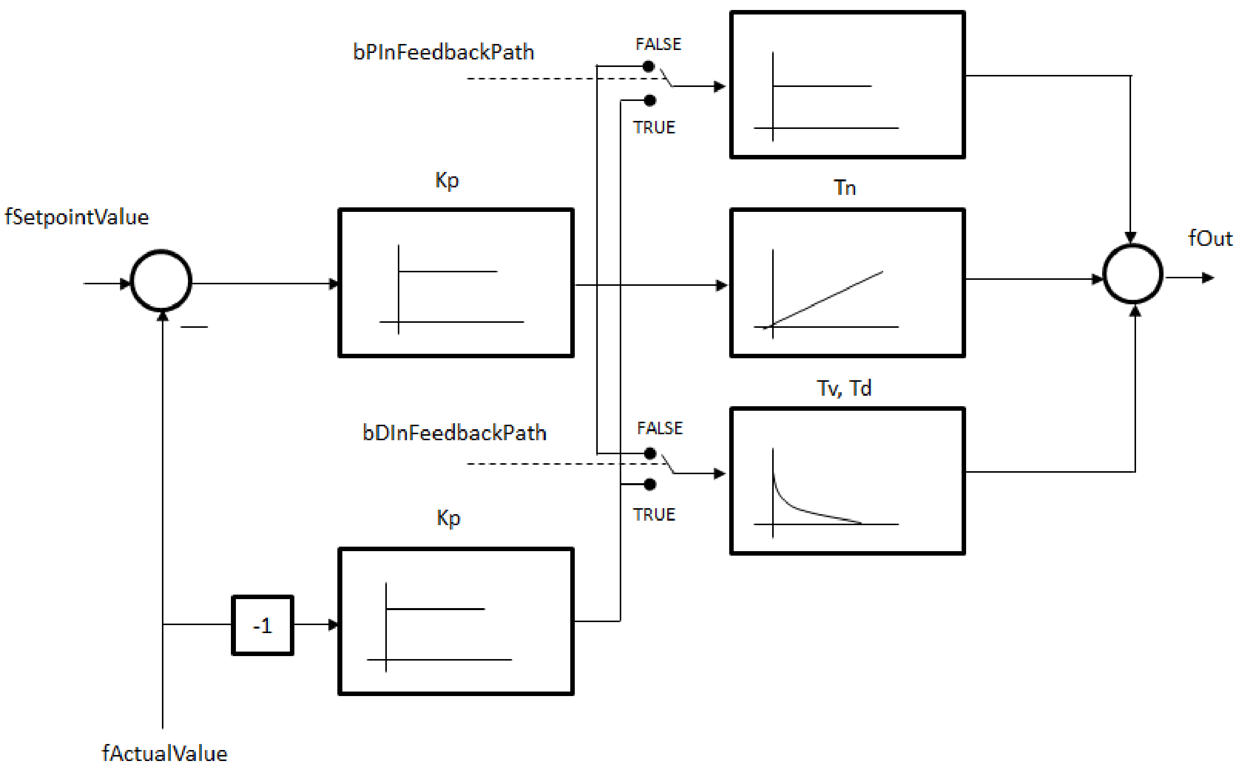
The switching algorithm calculates a linear, time-dependent transition between the two parameter sets. The `nParameterChangeCycleTicks` parameter can be used to specify the number of task cycles over which the continuous change between the two parameter sets takes place.

Transfer function

The following transfer function can be specified if the boolean Inputs `bPInTheFeedbackPath` and `bDInTheFeedbackPath` are FALSE. Otherwise the transfer function only describes part of the transfer behavior of the function block.

$$G_{PID}(s) = K_p \left(1 + \frac{1}{T_n s} + \frac{T_v s}{1 + T_d s} \right)$$

Functional diagram



The standard functional diagram of a PID controller in additive form has been expanded by the two active boolean Inputs `bPInTheFeedbackPath` and `bDInTheFeedbackPath` (which act as "switches"), so that a modified functional diagram can be activated.

Control background: due to the differential component of the control algorithm, large control values are generated at setpoint step-changes, which cause a strain on the control elements and may cause the control system to oscillate. A control algorithm with a differential component that is only applied to the controlled variable (`bDInTheFeedbackPath := TRUE`) can avoid this problem.

The `bPInTheFeedbackPath` and `bDInTheFeedbackPath` Inputs permit the closed control loop to implement the following transfer functions:

bPInTheFeedbackPath	bDInTheFeedbackPath	G(s)
false	false	$G(s)=\frac{G_{PID}(s)\cdot G_S(s)}{1+G_{PID}(s)\cdot G_S(s)}$
true	false	$G(s)=\frac{G_{DI}(s)\cdot G_S(s)}{1+G_{PID}(s)\cdot G_S(s)}$
false	true	$G(s)=\frac{G_{PI}(s)\cdot G_S(s)}{1+G_{PID}(s)\cdot G_S(s)}$
true	true	$G(s)=\frac{G_{I}(s)\cdot G_S(s)}{1+G_{PID}(s)\cdot G_S(s)}$

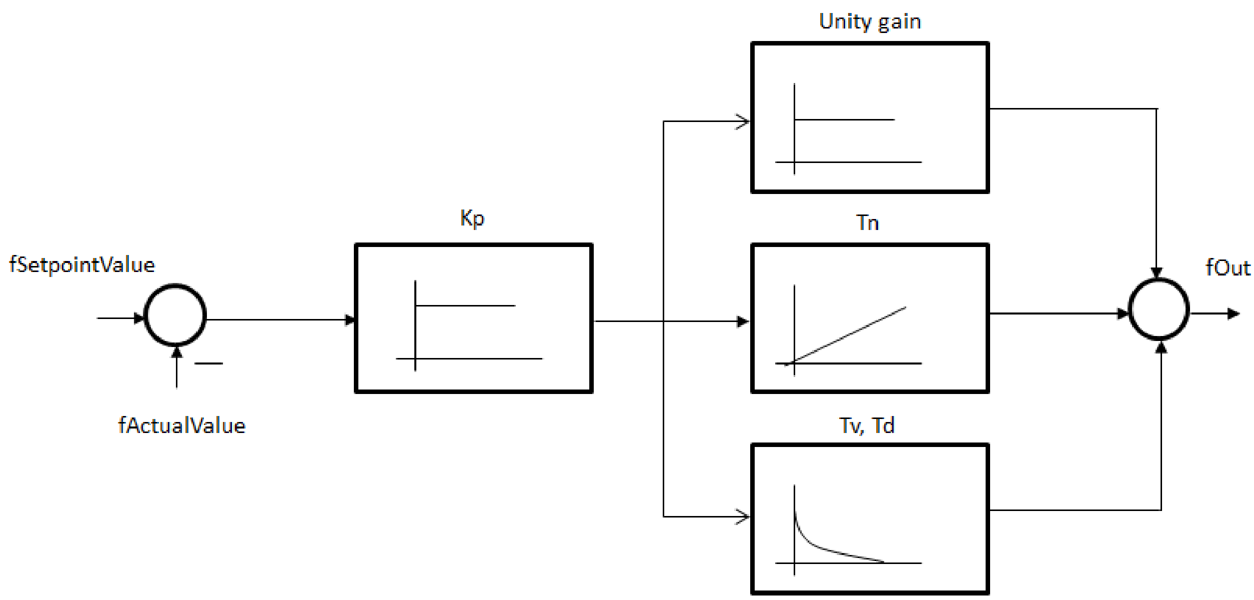
with:

$$G_{DI}(s)=K_p\left(\frac{1}{T_n s}+\frac{T_v s}{1+T_d s}\right)$$

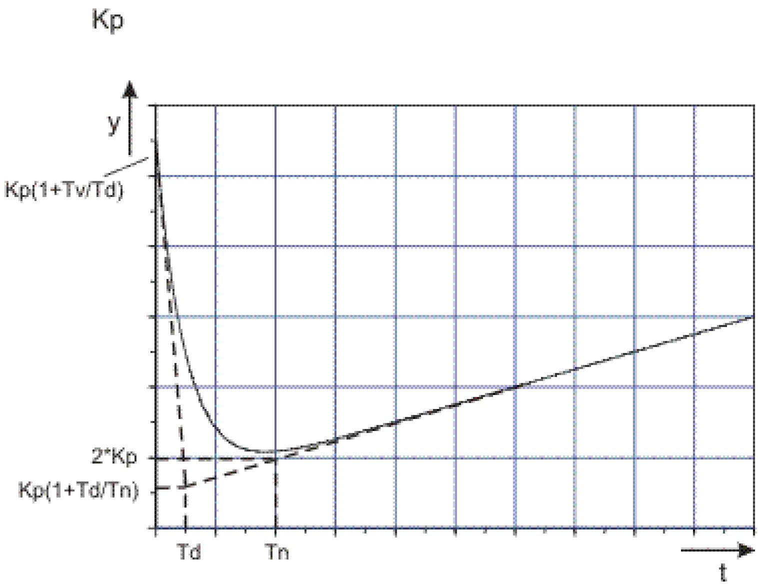
$$G_{PI}(s)=K_p\left(1+\frac{1}{T_n s}\right)$$

$$G_I(s)=K_p\left(\frac{1}{T_n s}\right)$$

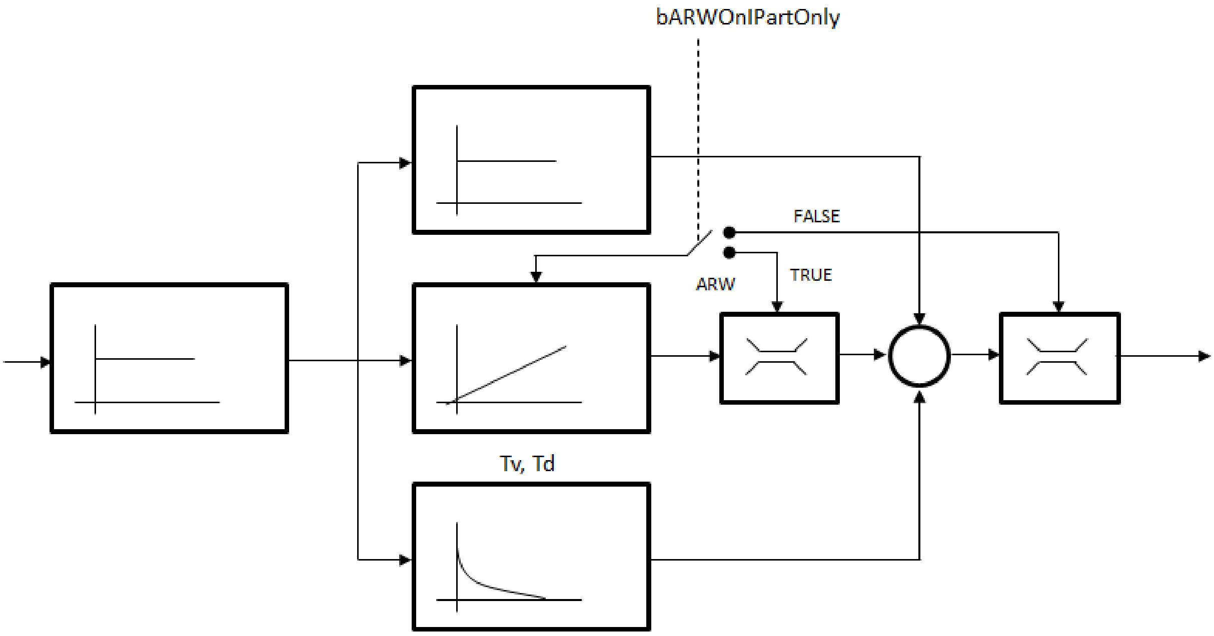
The standard setting for the two `bPInTheFeedbackPath` and `bDInTheFeedbackPath` Inputs is **FALSE**. The PID controller then corresponds to a standard PID controller in additive form.



Step response



ARW



Inputs

```
VAR_INPUT
    fSetpointValue       : FLOAT;
    fActualValue         : FLOAT;
    eParameterRecord     : E_CTRL_PARAMETER_RECORD;
    fManSyncValue        : FLOAT;
    bSync                : BOOL;
    eMode                : E_CTRL_MODE;
    bHold                : BOOL;
END_VAR
```

Name	Type	Description
fSetpointValue	FLOAT	Setpoint of the controlled variable
fActualValue	FLOAT	Actual value of the controlled variable
eParameter Record	E_CTRL_ PARAMETER_ RECORD	Index of the active parameter set
fManSyncValue	FLOAT	Input with which the PI element can be set.
bSync	BOOL	A rising edge at this input sets the PI-element to the value "fManSyncValue".
eMode	E_CTRL_MODE	Input that specifies the operation mode [▸ 173] of the function block.
bHold	BOOL	A TRUE at this input will hold the internal state (and therefore also the output) constant at its current value, independently of the control deviation.

Outputs

```
VAR_OUTPUT
    fOutPos                     : FLOAT;
    fOutNeg                     : FLOAT;
    fOut                        : FLOAT;
    bARWActive                  : BOOL := FALSE;
    bParameterChangeActive      : BOOL;
    eState                      : E_CTRL_STATE;
    bError                      : BOOL;
    eErrorId                    : E_CTRL_ERRORCODES;
END_VAR
```

Name	Type	Description
fOutPos	FLOAT	Output of the PID-element when the control value is positive. A zero is output otherwise.
fOutNeg	FLOAT	Output of the PID-element when the control value is negative. A zero is output otherwise.
fOut	FLOAT	Output of the PID element
bARWActive	BOOL	A TRUE at this output indicates that the PID-element is being restricted.
bParameter ChangeActive	BOOL	A TRUE at this output indicates that the change from one parameter set to the other is in progress.
eState	E_CTRL_STATE	State of the function block
bError	BOOL	Becomes TRUE, as soon as an error occurs.
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▸ 173] when the output `bError` is set.

Inputs/ outputs

```
VAR_IN_OUT
    stParams        : ST_CTRL_PID_SPLITRANGE_PARAMS;
END_VAR
```

Name	Type	Description
stParams	ST_CTRL_PID_ SPLITRANGE_ PARAMS	Parameter structure of the PID element

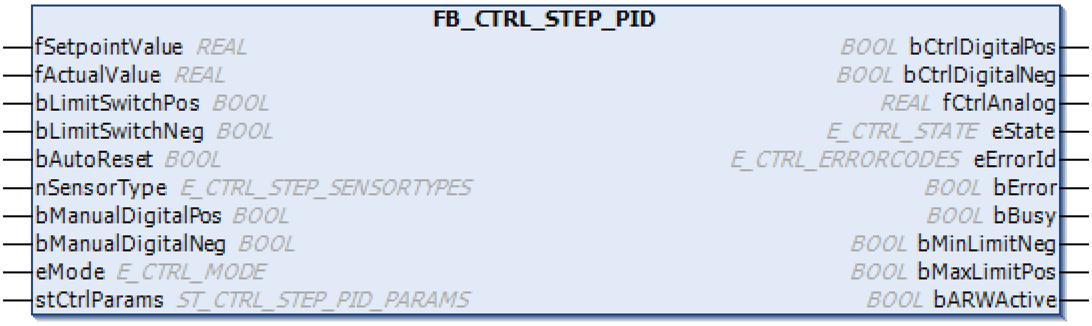
`stParams` consists of the following elements:

```
TYPE
ST_CTRL_PID_SPLITRANGE_PARAMS :
STRUCT
    tCtrlCycleTime                  : TIME := T#0ms;
    tTaskCycleTime                  : TIME := T#0ms;
    fKp_heating                     : FLOAT := 0;
    tTn_heating                     : TIME := T#0ms;
```

```
    tTv_heating                    : TIME := T#0ms;
    tTd_heating                    : TIME := T#0ms;
    fKp_cooling                    : FLOAT := 0;
    tTn_cooling                    : TIME := T#0ms;
    tTv_cooling                    : TIME := T#0ms;
    tTd_cooling                    : TIME := T#0ms;
    nParameterChangeCycleTicks     : INT;
    fOutMaxLimit                   : FLOAT := 1E38;
    fOutMinLimit                   : FLOAT := -1E38;
    bPInTheFeedbackPath            : BOOL;
    bDInTheFeedbackPath            : BOOL;
    bARWOnIPartOnly                : BOOL;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
Range **eCTRL_PARAMETER_RECORD_HEATING**:		
fKp_heating	FLOAT	Controller amplification / controller coefficient
tTn_heating	TIME	Integral action time: The I-component is deactivated if this is parameterized as T#0s.
tTv_heating	TIME	Derivative action time: The D-component is deactivated if this is parameterized as T#0s.
tTd_heating	TIME	Damping time
Range **eCTRL_PARAMETER_RECORD_COOLING**:		
fKp_cooling	FLOAT	Controller amplification / controller coefficient
tTn_cooling	TIME	Integral action time: The I-component is deactivated if this is parameterized as T#0s.
tTv_cooling	TIME	Derivative action time: The D-component is deactivated if this is parameterized as T#0s.
tTd_cooling	TIME	Damping time
nParameter ChangeCycle Ticks	INT	The number of task cycles over which the change from one parameter set to the other takes place.
fOutMaxLimit	FLOAT	Upper limit at which integration is halted and to which the output is limited (ARW measure). Reaching this limit is indicated by a TRUE at the `bARWActive` output.
fOutMinLimit	FLOAT	Lower limit at which integration is halted and to which the output is limited (ARW measure). Reaching this limit is indicated by a TRUE at the `bARWActive` output.
bPInTheFeed backPath	BOOL	Input value of the P-element can be selected with this input (see functional diagram).
bDInTheFeed backPath	BOOL	Input value of the D-element can be selected with this input (see functional diagram).
bARWOnIPartOn ly	BOOL	If this parameter is FALSE (the standard setting), the integration of the I-component is halted if the complete controller output reaches the upper or lower limit. If it is TRUE, the integration is halted if the I-component (the output of the integrator) reaches some limit. (Cf. functional diagram.)

4.2.1.3.13 FB_CTRL_STEP_PID



The function block controls a digitally manipulated variable for the integration of actuators. It is based on a standard PID controller with an additional binary output signal in conjunction with the analog output and works without a precise position feedback signal.

The position feedback signal required to calculate the controller output is estimated using the limiting values and the time required to reach these limits. The function block then generates pulses that are required to drive the actuator as a servomotor.

Inputs

```
VAR_INPUT
fSetpointValue         : LREAL;
fActualValue           : LREAL;
bLimitSwitchPos        : BOOL;
bLimitSwitchNeg        : BOOL;
bAutoReset             : BOOL;
nSensorType            : E_CTRL_STEP_SENSORTYPES;
fDisturbanceValue      : REAL;
bManualDigitalPos      : BOOL;
bManualDigitalNeg      : BOOL;
eMode                  : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
fSetpointValue	LREAL	Setpoint of the controlled variable
fActualValue	LREAL	Actual value of the control value
bLimitSwitchPos	BOOL	Limit switch, becomes TRUE when the upper stop is reached.
bLimitSwitchNeg	BOOL	Limit switch, becomes TRUE when the lower stop is reached.
bAutoReset	BOOL	If the variable is TRUE, it resets the controller to initial conditions.
nSensorType	E_CTRL_STEP_ SENSORTYPES	Enables the selection of the correct thermocouple type.
fDisturbance Value	REAL	Actual value of the disturbance variable
bManualDigital Pos	BOOL	If the variable is TRUE, the motor is manually set to the positive direction
bManualDigital Neg	BOOL	If TRUE, the motor is manually set to the negative direction
eMode	E_CTRL_MODE	Input specifying the operation mode of the controller

Outputs

```
VAR_OUTPUT
bCtrlDigitalPos    : BOOL;
bCtrlDigitalNeg    : BOOL;
fCtrlAnalog        : LREAL;
eState             : E_CTRL_STATE;
eErrorId           : E_CTRL_ERRORCODES;
bError             : BOOL;
```

```
bBusy              : BOOL;
bMinLimitNeg       : BOOL;
bMinLimitPos       : BOOL;
bARWActive         : BOOL;
END_VAR
```

Name	Type	Description
bCtrlDigitalPos	BOOL	Output required to run the motor in a positive direction.
bCtrlDigitalNeg	BOOL	Output required to run the motor in a negative direction.
fCtrlAnalog	LREAL	Analog output of the controller
eState	E_CTRL_STATE	Actual state of the controller
eErrorId	E_CTRL_ERROR CODES	Returns the error number if `bError` is TRUE.
bError	BOOL	TRUE if an error occurs in the function block.
bBusy	BOOL	TRUE when the function block is active.
bMinLimitNeg	BOOL	TRUE at this output indicates that the controller has reached its minimum limit value.
bMinLimitPos	BOOL	TRUE at this output indicates that the controller has reached its maximum limit value.
bARWActive	BOOL	TRUE at this output indicates that the controller is currently restricted.

Inputs/ outputs

```
VAR_IN_OUT
stCtrlParams : ST_CTRL_STEP_PID_PARAMS;
END_VAR
```

Name	Type	Description
stCtrlParams	ST_CTRL_STEP_ PID_PARAMS	Parameter structure of the controller.

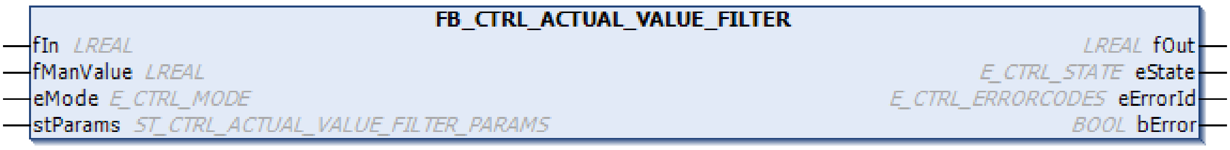
`stCtrlParams` consists of the following elements:

```
TYPE ST_CTRL_STEP_PID_PARAMS:
STRUCT
tCtrlCycleTime      : TIME;
tTaskCycleTime      : TIME;
fKp                 : REAL;
fTn                 : REAL;
fTv                 : REAL;
fTd                 : REAL;
fTM                 : REAL;
fDeadBandWidth      : REAL;
tMinimumPulseTime   : TIME;
tFilterTime         : TIME;
fWmax               : REAL;
fWmin               : REAL;
fYMax               : REAL;
fYMin               : REAL;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value for internal calculation to determine whether the state and output values have been updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle, this corresponds to the task cycle time of the calling task.
fKp	REAL	Controller gain or proportional gain
fTn	REAL	Integral action time: The I component is disabled if this parameter is zero.
fTv	REAL	Derivative action time; the D component is disabled if this parameter is zero.
fTd	REAL	Damping time
fTM	REAL	Digital valve actuation time in seconds
fDeadBandWidth	REAL	Permissible width of the controller deviation
tMinimumPulse Time	TIME	Minimum pulse time for pulse generator
tFilterTime	TIME	Filter time for actual value filter
fWmax	REAL	Maximum setpoint in °C
fWmin	REAL	Minimum setpoint in °C
fYMax	REAL	Maximum value of the controller output
fYMin	REAL	Minimum value of controller output

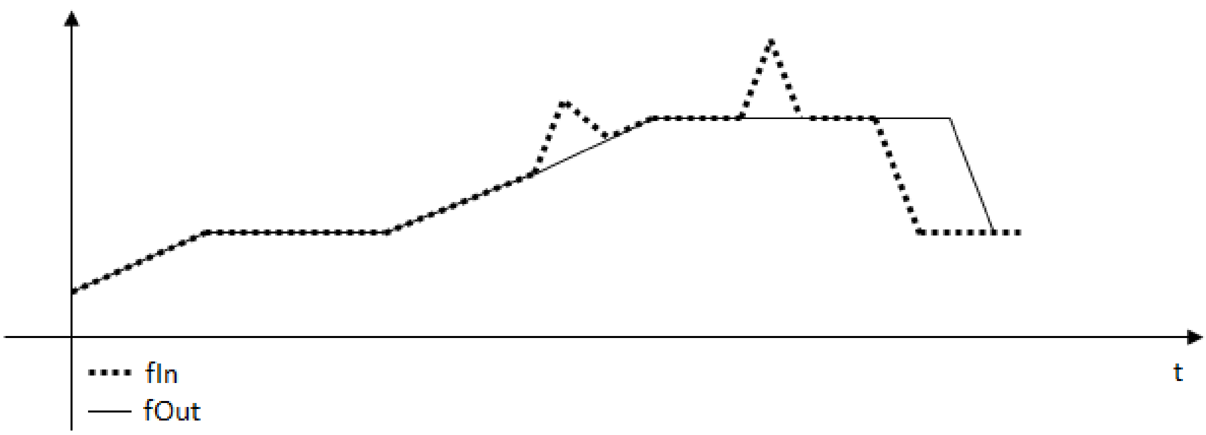
4.2.1.4 Filter / ControlledSystemSimulation

4.2.1.4.1 FB_CTRL_ACTUAL_VALUE_FILTER



The function block allows a measured input value to be checked for plausibility and filtered.

Behavior of the output



This function block allows a plausibility check to be carried out on a measured input value. If the difference between two sampling values in sequence (measurements) is larger than the specified window, `fDeltaMax`, then the current input value is suppressed for a maximum of three cycles. During this time the output value is extrapolated from the previous input values. If the specified window is exceeded for more than three cycles, the output will again follow the input value. The behavior of the output is illustrated in the diagram above.

Inputs

```
VAR_INPUT
    fIn         : FLOAT;
    fManValue   : FLOAT;
    eMode       : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input value of the filter
fManValue	FLOAT	Input value that is output in manual mode.
eMode	E_CTRL_MODE	Input that specifies the operation mode [▶ 173] of the function block.

Outputs

```
VAR_OUTPUT
    fOut       : FLOAT;
    eState     : E_CTRL_STATE;
    eErrorId   : E_CTRL_ERRORCODES;
    bError     : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output of the function block
eState	E_CTRL_ STATE	State of the function block
eErrorId	E_CTRL_ ERROR CODES	Supplies the error number [▶ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams     : ST_CTRL_ACTUAL_VALUE_FILTER_PARAMS;
END_VAR
```

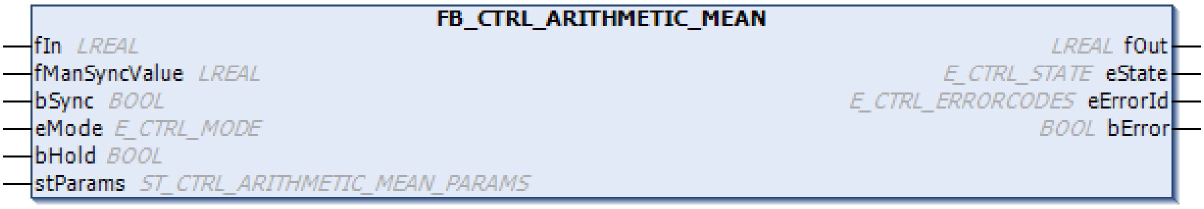
Name	Type	Description
stParams	ST_CTRL_ACTU AL_VALUE_FILTE R_ PARAMS	Parameter structure of the actual value filter element

`stParams` consists of the following elements:

```
TYPE
ST_CTRL_ACTUAL_VALUE_FILTER_PARAMS:
STRUCT
    tCtrlCycleTime     : TIME := T#0ms;
    tTaskCycleTime     : TIME := T#0ms;
    fDeltaMax          : FLOAT;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
fDeltaMax	FLOAT	Maximum difference between two input values in sequence. See description above.

4.2.1.4.2 FB_CTRL_ARITHMETIC_MEAN



$$\bar{\chi} = \frac{1}{n}\sum_{n} \chi_n$$

Inputs

```
VAR_INPUT
    fIn             : FLOAT;
    fManSyncValue   : FLOAT;
    bSync           : BOOL;
    eMode           : E_CTRL_MODE;
    bHold           : BOOL;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input value for the mean value filter
fManSync Value	FLOAT	Input value to which the mean value filter can be set, or that is issued at the output in manual mode.
bSync	BOOL	A rising edge at this input sets the mean value filter to the value "fManSyncValue".
eMode	E_CTRL_MODE	Input that specifies the operation mode [▸ 173] of the function block.
bHold	BOOL	A TRUE at this input will hold the internal state (and therefore also the output) constant at its current value, independently of the input value.

Outputs

```
VAR_OUTPUT
    fOut       : FLOAT;
    eState     : E_CTRL_STATE;
    eErrorId   : E_CTRL_ERRORCODES;
    bError     : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output of the mean value filter
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▶ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams     : ST_CTRL_ARITHMETIC_MEAN_PARAMS;
END_VAR
```

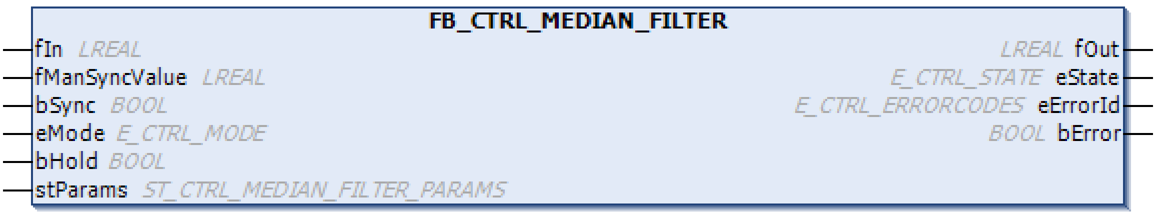
Name	Type	Description
stParams	ST_CTRL_ ARITHMETIC_ MEAN_PARAMS	Parameter structure of the mean value filter

`stParams` consists of the following elements:

```
TYPE
ST_CTRL_ARITHMETIC_MEAN_PARAMS:
STRUCT
    tCtrlCycleTime : TIME := T#0ms;
    tTaskCycleTime : TIME := T#0ms;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.

4.2.1.4.3 FB_CTRL_MEDIAN_FILTER



The function block specifies the median. The median is the mean value of a list of values ordered by size. This means that one half of the collected data values is smaller than the median value, the other half is larger.

The programmer must create an array "ARRAY [1..2(n+2)] OF LREAL", in which the function block can store the internal required data.

Inputs

```
VAR_INPUT
    fIn                 : FLOAT;
    fManSyncValue       : FLOAT;
    eMode               : E_CTRL_MODE;
    bSync               : FLOAT;
    bHold               : BOOL;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input value of the median filter
fManSyncValue	FLOAT	Input value to which the median filter can be set or which is output in manual mode.
eMode	E_CTRL_MODE	With a rising edge at this input the median filter is set to the value "fManSyncValue".
bSync	FLOAT	Input that specifies the operation mode of the function block.
bHold	BOOL	TRUE at this input holds the internal state and thus also the output.

Outputs

```
VAR_OUTPUT
    fOut        : FLOAT;
    eState      : E_CTRL_STATE;
    eErrorId    : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

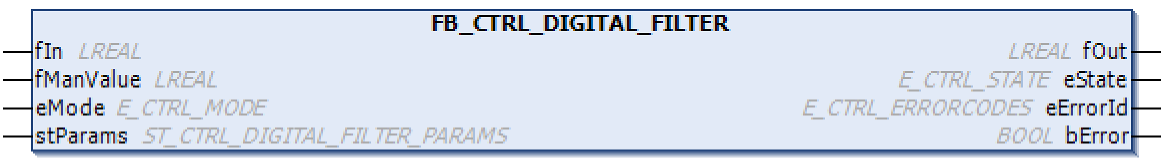
Name	Type	Description
fOut	FLOAT	Output of the median filter
eState	E_CTRL_STATE	Status of the function block
eErrorId	E_CTRL_ERROR CODES	Supplies the error number when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    tCtrlCycleTime          : TIME := T#0ms;
    tTaskCycleTime          : TIME := T#0ms;
    nSamplesToFilter        : UINT;
    pWorkArray_ADR          : POINTER TO FLOAT := 0;
    nWorkArray_SIZEOF       : UINT := 0;
END_VAR
```

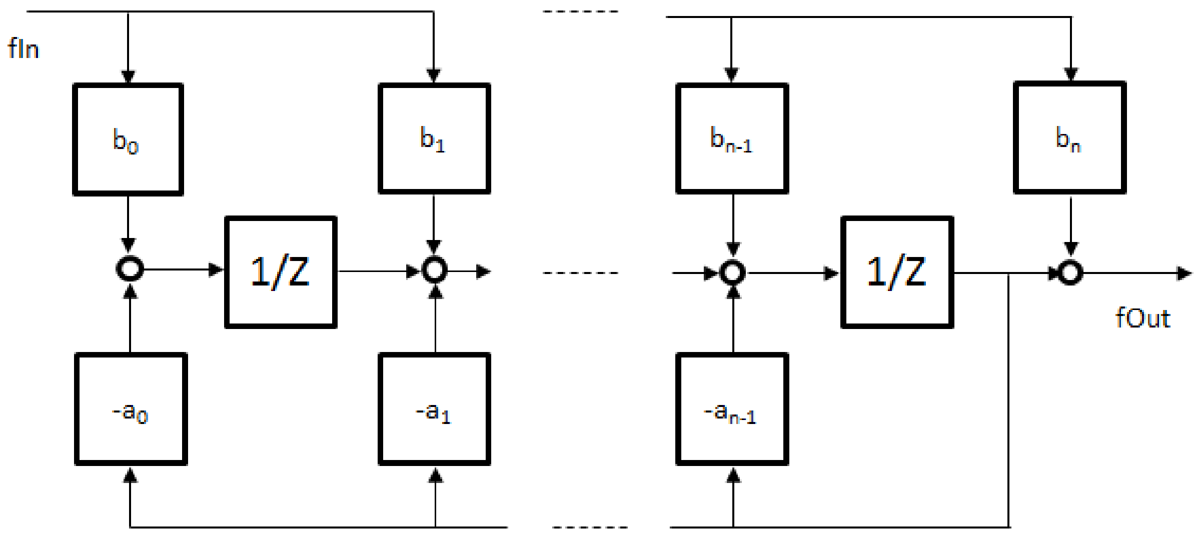
Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
nSamplesTo Filter	UINT	Number of values "n" used to form the median value.
pWorkArray_ ADR	POINTER TO FLOAT	The address of the array where the input values are temporarily stored.
nWorkArray_ SIZEOF	UINT	Size of the `WorkArrays`

4.2.1.4.4 FB_CTRL_DIGITAL_FILTER



This function block calculates a discrete transfer function with the structure described below. This structure allows either an FIR filter (**F**inite **I**mpulse **R**esponse) or an IIR filter (**I**nfinite **I**mpulse **R**esponse) to be implemented. The transfer function here can be of any order, n.

The coefficients for the following transfer structure are stored in the parameter arrays:



$$G(z)=\frac{Y(z)}{X(z)}=\frac{b_1+b_2z^{-1}+b_3z^{-2}+...+b_nz^{-(n-1)}}{1+a_2z^{-1}+a_3z^{-2}+...+a_nz^{-(n-1)}}$$

The programmer must create the following arrays in the PLC if this function block is to be used:

```
aCoefficientsArray_a         : ARRAY[1..nFilterOrder+1] OF FLOAT;
aCoefficientsArray_b         : ARRAY[1..nFilterOrder+1] OF FLOAT;
aStDigitalFilterData         : ARRAY[1..nFilterOrder] OF
ST_CTRL_DIGITAL_FILTER_DATA;
```

The coefficients "b_1" to "b_n" are stored in the array `aCoefficientsArray_b`. This must be organized as follows:

```
aCoefficientsArray_b[1]  := b1;
aCoefficientsArray_b[2]  := b2;
…
aCoefficientsArray_b[n-1]  := bn-1;
aCoefficientsArray_b[n]    := bn;
```

The coefficients "a1" to "an" are stored in the array `aCoefficientsArray_a`.
This must be organized as follows:

```
aCoefficientsArray_a[1]  := xxx;  (* not being evaluated *)
aCoefficientsArray_a[2]  := a2;
...
aCoefficientsArray_a[n-1]  := an-1;
aCoefficientsArray_a[n]    := an;
```

The internal data required by the function block is stored in the array `aStDigitalFilterData`. This data must never be modified from within the PLC program. This procedure is also illustrated in the sample program listed below.

Inputs

```
VAR_INPUT
    fIn         : FLOAT;
    fManValue   : FLOAT;
    eMode       : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input of the digital filter
fManValue	FLOAT	Input whose value is present at the output in manual mode.
eMode	E_CTRL_MODE	Input that specifies the operation mode of the function block.

Outputs

```
VAR_OUTPUT
    fOut       : FLOAT;
    eState     : E_CTRL_STATE;
    bError     : BOOL;
    eErrorId   : E_CTRL_ERRORCODES;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output of the digital filter
eState	E_CTRL_STATE	State of the function block
bError	BOOL	Becomes TRUE, as soon as an error occurs.
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▶ 173] when the output `bError` is set.

Inputs/ outputs

```
VAR_IN_OUT
    stParams          : ST_CTRL_DIGITAL_FILTER_PARAMS;
END_VAR
```

Name	Type	Description
stParams	ST_CTRL_ DIGITAL_FILTER _PARAMS	Parameter structure of the function block

`stParams` consists of the following elements:

```
TYPE
ST_CTRL_DIGITAL_FILTER_PARAMS :
STRUCT
    tTaskCycleTime                : TIME;
    tCtrlCycleTime                : TIME := T#0ms;
    nFilterOrder                  : USINT;
    pCoefficientsArray_a_ADR      : POINTER TO FLOAT := 0;
    nCoefficientsArray_a_SIZEOF   : UINT;
    pCoefficientsArray_b_ADR      : POINTER TO FLOAT := 0;
    nCoefficientsArray_b_SIZEOF   : UINT;
    pDigitalFilterData_ADR        : POINTER TO ST_CTRL_DIGITAL_FILTER_DATA;
    nDigitalFilterData_SIZEOF     : UINT;
END_STRUCT
END_TYPE
```

Name	Type	Description
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
nFilterOrder	USINT	Order of the digital filter [0...]
pCoefficientsArray_a_ADR	POINTER TO FLOAT	Address of the array containing the a coefficients
nCoefficientsArray_a_SIZEOF	UINT	Size of the array containing the a coefficients, in bytes
pCoefficientsArray_b_ADR	POINTER TO FLOAT	Address of the array containing the b coefficients
nCoefficientsArray_b_SIZEOF	UINT	Size of the array containing the b coefficients, in bytes
pDigitalFilterData_ADR	POINTER TO ST_CTRL_DIGITAL_FILTER_DATA	Address of the data array
nDigitalFilterData_SIZEOF	UINT	Size of the data array in bytes

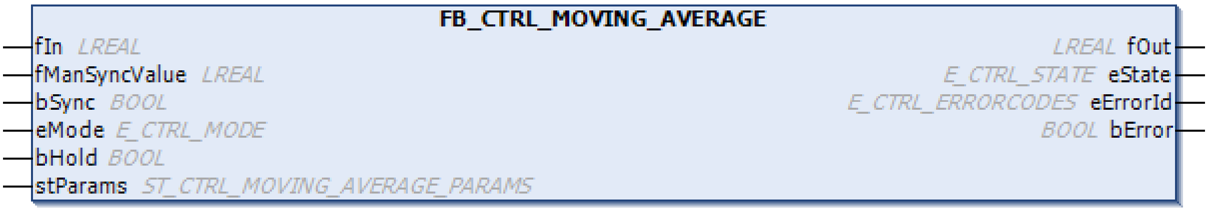
```
TYPE
ST_CTRL_DIGITAL_FILTER_DATA
STRUCT
 Internal structure. This must not be written to.
END_STRUCT
END_TYPE
```

Sample:

```
PROGRAM PRG_DIGITAL_FILTER_TEST
VAR
    fbDigitalFilter        : FB_CTRL_DIGITAL_FILTER;
    aCoefficientsArray_a    : ARRAY[1..3] OF FLOAT;
    aCoefficientsArray_b    : ARRAY[1..3] OF FLOAT;
    aStDigitalFilterData    : ARRAY[1..2] OF ST_CTRL_DIGITAL_FILTER_DATA;
    eMode                 : E_CTRL_MODE;
    stParams             : ST_CTRL_DIGITAL_FILTER_PARAMS;
    eErrorId             : E_CTRL_ERRORCODES;
    bError             : BOOL;
    fIn                 : FLOAT := 0;
    fOut                 : FLOAT;
    bInit                 : BOOL := TRUE;
END_VAR

IF bInit THEN
    aCoefficientsArray_a[1] := 0.0; (* not used *)
    aCoefficientsArray_a[2] := 0.2;
    aCoefficientsArray_a[3] := 0.1;
    aCoefficientsArray_b[1] := 0.6;
    aCoefficientsArray_b[2] := 0.4;
    aCoefficientsArray_b[3] := 0.2;

    stParams.tTaskCycleTime := T#2ms;
    stParams.tCtrlCycleTime := T#2ms;
    stParams.nFilterOrder := 2;

    eMode := eCTRL_MODE_ACTIVE;
    bInit := FALSE;
END_IF

stParams.pCoefficientsArray_a_ADR := ADR(aCoefficientsArray_a);
stParams.nCoefficientsArray_a_SIZEOF := SIZEOF(aCoefficientsArray_a);
stParams.pCoefficientsArray_b_ADR := ADR(aCoefficientsArray_b);
stParams.nCoefficientsArray_b_SIZEOF := SIZEOF(aCoefficientsArray_b);
stParams.pDigitalFilterData_ADR := ADR(astDigitalFilterData);
stParams.nDigitalFilterData_SIZEOF := SIZEOF(aStDigitalFilterData);
```

```
fbDigitalFilter ( fIn := fIn,
                  eMode := eMode,
                  stParams := stParams,
                  fOut => fOut,
                  eErrorId => eErrorId,
                  bError => bError);
```

4.2.1.4.5 FB_CTRL_MOVING_AVERAGE



The function block provides a moving average filter in the functional diagram.

The arithmetic mean value of the last "n" values is calculated.

$$\overline{\chi}_k^{\,n} = \frac{1}{n} \sum_{i=k-n} \chi_i$$

The programmer must create an array "ARRAY [1...n] of FLOAT" in the PLC, in which the function block can store the internally required data.

Inputs

```
VAR_INPUT
    fIn              : FLOAT;
    fManSyncValue    : FLOAT;
    bSync          : BOOL;
    eMode          : E_CTRL_MODE;
    bHold          : BOOL;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input value for the moving average filter
fManSyncValue	FLOAT	Input value to which the moving average filter can be set, or that is issued at the output in manual mode.
bSync	BOOL	A rising edge at this input sets the moving average filter to the value "fManSyncValue".
eMode	E_CTRL_MODE	Input that specifies the operation mode [▸ 173] of the function block.
bHold	BOOL	A TRUE at this input will hold the internal state (and therefore also the output) constant at its current value, independently of the input value.

Outputs

```
VAR_OUTPUT
    fOut         : FLOAT;
    eState     : E_CTRL_STATE;
    eErrorId     : E_CTRL_ERRORCODES;
    bError     : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output of the moving average filter
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▶ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams              : ST_CTRL_MOVING_AVERAGE_PARAMS;
END_VAR
```

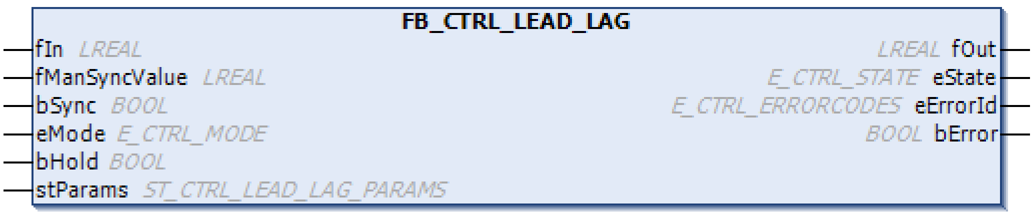
Name	Type	Description
stParams	ST_CTRL_MOVING_AVERAGE_PARAMS	Parameter structure of the moving average filter

`stParams` consists of the following elements:

```
TYPE
ST_CTRL_MOVING_AVERAGE_PARAMS:
STRUCT
    tCtrlCycleTime        : TIME := T#0ms;
    tTaskCycleTime        : TIME := T#0ms;
    nSamplesToFilter      : UINT;
    pWorkArray_ADR        : POINTER TO FLOAT := 0;
    nWorkArray_SIZEOF     : UINT     := 0;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
nSamplesToFilter	UINT	The number of values, "n", whose arithmetic mean value is calculated.
pWorkArray_ADR	POINTER TO FLOAT	The address of the array where the input values are temporarily stored.
nWorkArray_SIZEOF	UINT	Size of the `WorkArrays`

4.2.1.4.6 FB_CTRL_LEAD_LAG

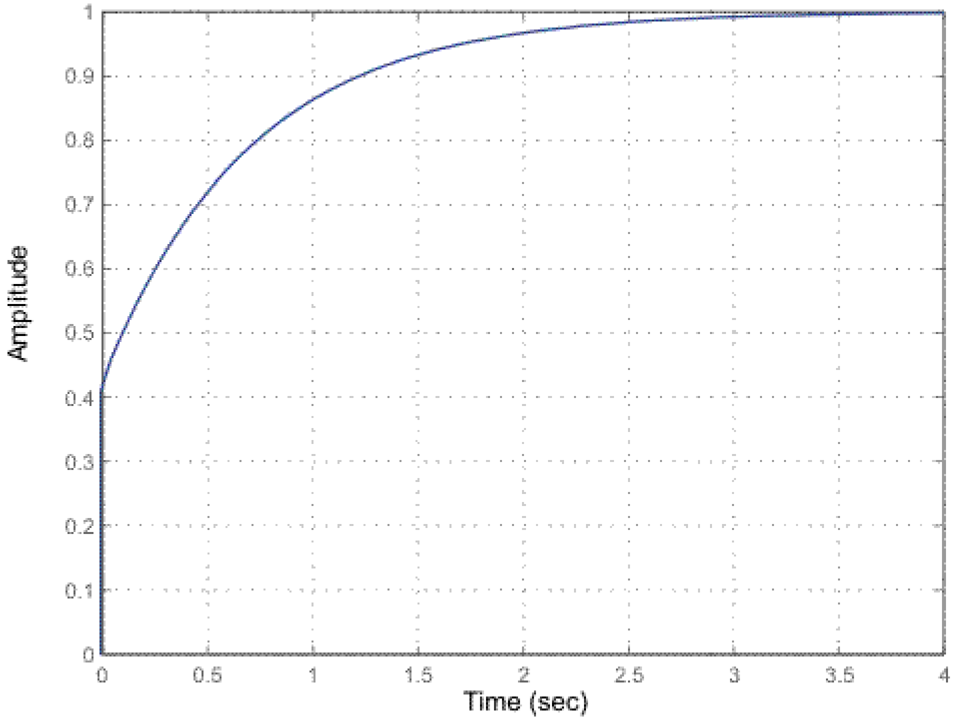


The function block represents a digital lead/lag filter.

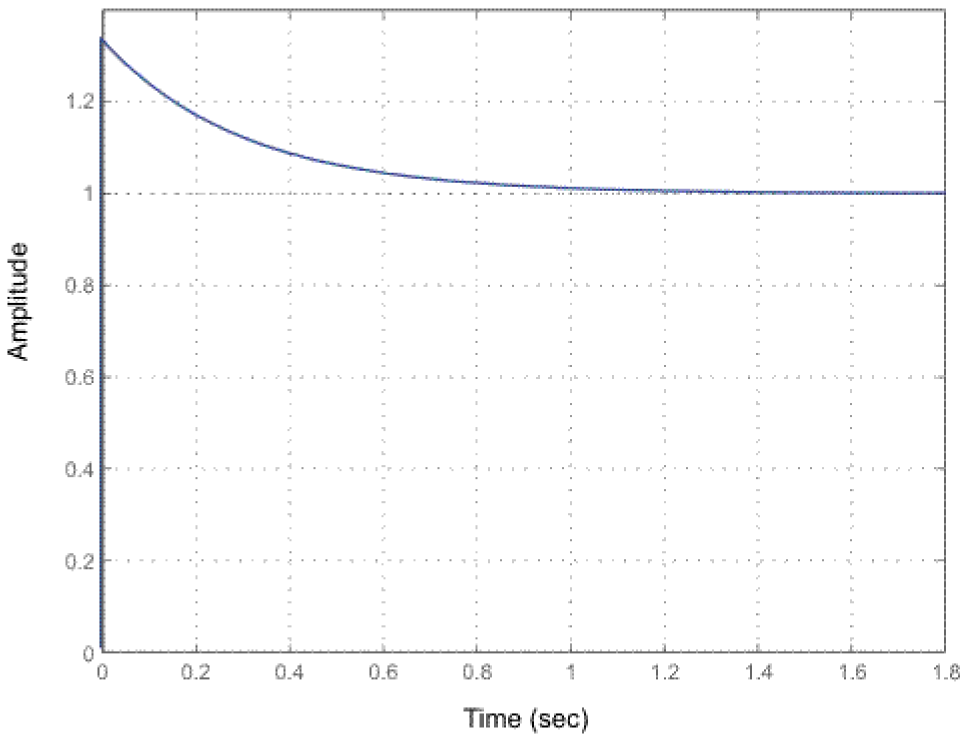
Transfer function

$$G(s) = \frac{1 + T_1 s}{1 + T_2 s}$$

Step response with T2 > T1



Step response with T1 > T2



The step response at time T=0 is T1 / T2.

Inputs

```
VAR_INPUT
    fIn                 : FLOAT;
    fManSyncValue       : FLOAT;
    bSync               : FLOAT;
    fManValue           : FLOAT;
    eMode               : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input value of the notch filter
fManSyncValue	FLOAT	Input value that is output at the output in manual mode.
bSync	FLOAT	reserve
fManValue	FLOAT	Input that specifies the operation mode [▸ 173] of the function block.
eMode	E_CTRL_MODE	A TRUE at this input will hold the internal state (and therefore also the output) constant at its current value, independently of the input value.

Outputs

```
VAR_OUTPUT
    fOut        : FLOAT;
    eState      : E_CTRL_STATE;
    eErrorId    : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Lead/lag filter output
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▸ 173] when the output `bError` is set. (p>)
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams    : ST_CTRL_LEAD_LAG_FILTER_PARAMS;
END_VAR
```

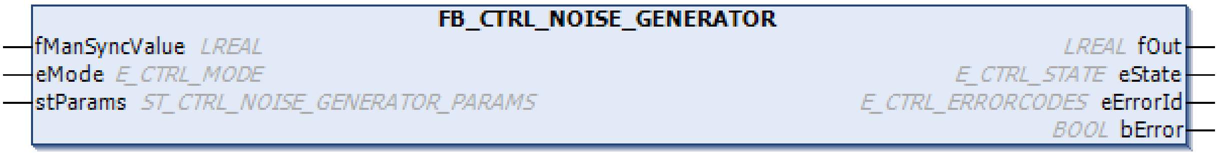
Name	Type	Description
stParams	ST_CTRL_LEAD_ LAG_FILTER_ PARAMS	Parameter structure of the lead/lag filter

`stParams` consists of the following elements:

```
TYPE
ST_CTRL_LEAD_LAG_FILTER_PARAMS:
STRUCT
    tCtrlCycleTime     : TIME := T#0ms;
    tTaskCycleTime     : TIME := T#0ms;
    tT1                : TIME := T#0ms;  (* T1 *)
    tT2                : TIME := T#0ms;  (* T2 *)
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
tT1	TIME	T1, see G(s).
tT2	TIME	T2, see G(s).

4.2.1.4.7 FB_CTRL_NOISE_GENERATOR



This function block generates a noise signal on the basis of the pseudo-random number in the range [-fAmplitude ... fAmplitude].

Output signal

Output signal with an amplitude of 5.0.

Inputs

```
VAR_INPUT
    fManSyncValue   : FLOAT;
    eMode           : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
fManSyncValue	FLOAT	Input value that is output at the output in manual mode.
eMode	E_CTRL_MODE	Input that specifies the operation mode [▶ 173] of the function block.

Outputs

```
VAR_OUTPUT
    fOut       : FLOAT;
    eState     : E_CTRL_STATE;
    eErrorId   : E_CTRL_ERRORCODES;
    bError     : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output of the noise generator
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERRORCODES	Supplies the error number [▶ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams        : ST_CTRL_NOISE_GENERATOR_PARAMS;
END_VAR
```

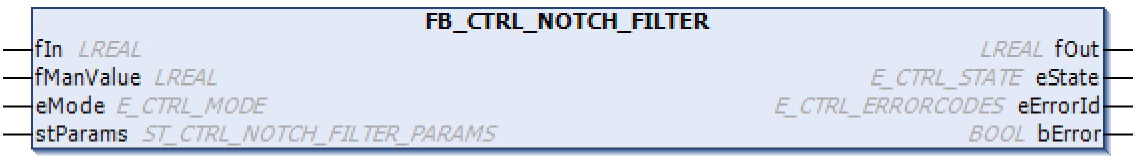
Name	Type	Description
stParams	ST_CTRL_NOISE_GENERATOR_PARAMS	Parameter structure of the noise generator

`stParams` consists of the following elements:

```
TYPE
ST_CTRL_NOISE_GENERATOR_PARAMS:
STRUCT
    tCtrlCycleTime : TIME := T#0ms;
    tTaskCycleTime : TIME := T#0ms;
    fAmplitude     : FLOAT := 0;
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
fAmplitude	FLOAT	Amplitude of the output signal: A noise signal extending over the range [-fAmplitude/2.0 ... fAmplitude/2.0] is created at the function block's output.

4.2.1.4.8 FB_CTRL_NOTCH_FILTER

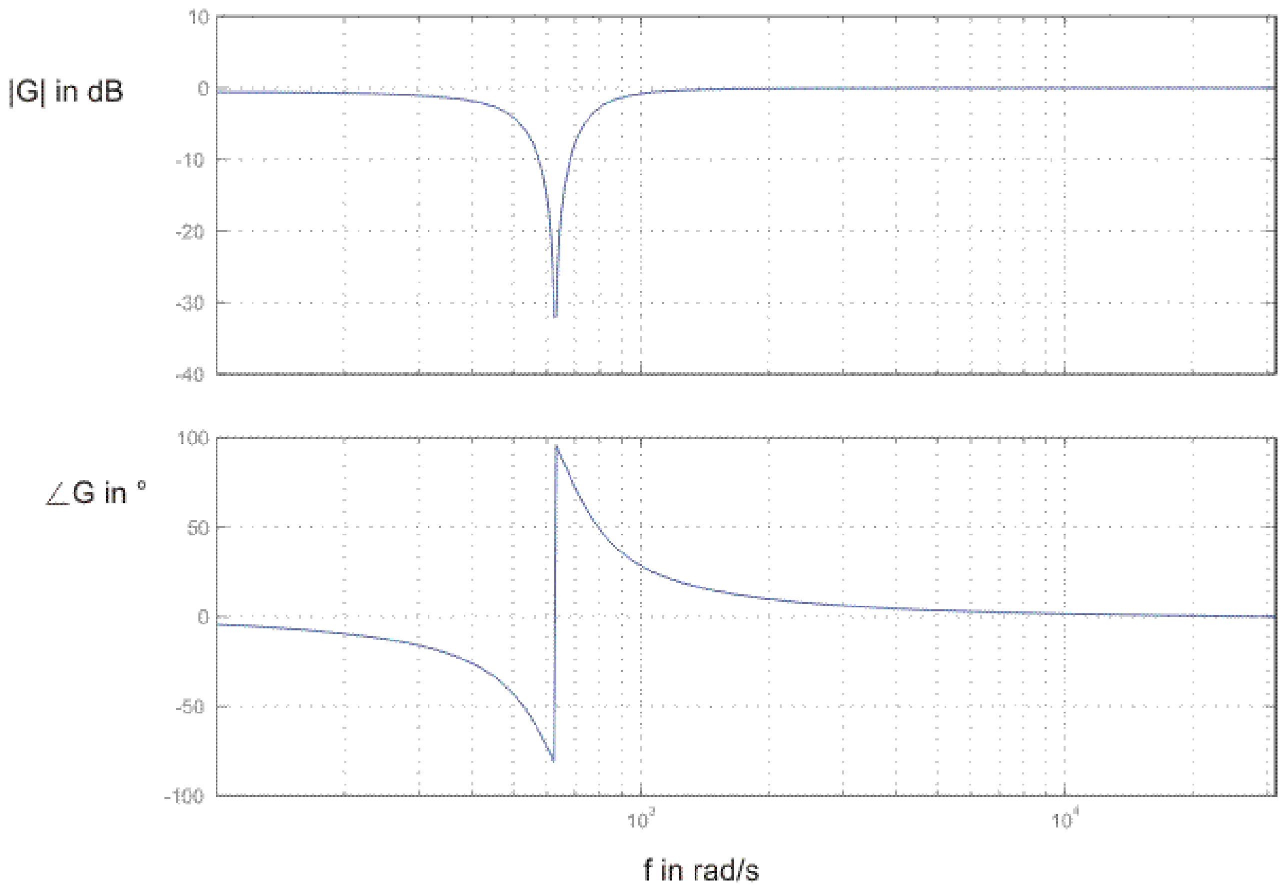


The function block represents a digital notch filter.

Transfer instruction

$$G(s) = \frac{s^2 + w_0^2}{s^2 + e w_0 s + w_0^2}$$

Bode diagram



with:

$$f_0 = 100Hz$$

$$e = 0.25$$

Inputs

```
VAR_INPUT
    fIn         : FLOAT;
    fManValue   : FLOAT;
    eMode       : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input value of the notch filter
fManValue	FLOAT	Input value that is output at the output in manual mode.
eMode	E_CTRL_MODE	Input that specifies the operation mode [▶ 173] of the function block.

Outputs

```
VAR_OUTPUT
    fOut       : FLOAT;
    eState     : E_CTRL_STATE;
    eErrorId   : E_CTRL_ERRORCODES;
    bError     : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Notch filter output
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▸ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams       : ST_CTRL_NOTCH_FILTER_PARAMS;
END_VAR
```

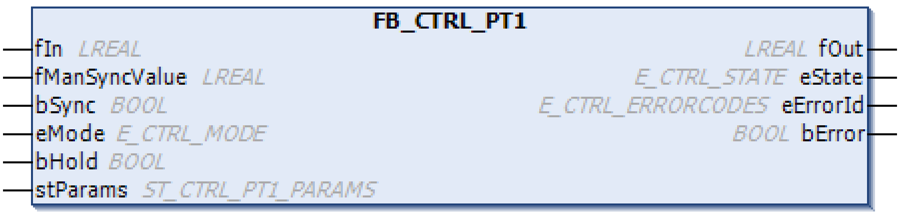
Name	Type	Description
stParams	ST_CTRL_ NOTCH_FILTER_ PARAMS	Parameter structure of the notch filter

`stParams` consists of the following elements:

```
TYPE
ST_CTRL_NOTCH_FILTER_PARAMS:
STRUCT
    tCtrlCycleTime   : TIME := T#0ms;
    tTaskCycleTime   : TIME := T#0ms;
    fNotchFreq       : FLOAT := 0;
    fBandwidth       : FLOAT := 0;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
fNotchFreq	FLOAT	notch frequency in Hz
fBandwidth	FLOAT	bandwidth relative to the notch frequency: bandwidth in Hz = fNotchFreq * fBandwidth

4.2.1.4.9 FB_CTRL_PT1

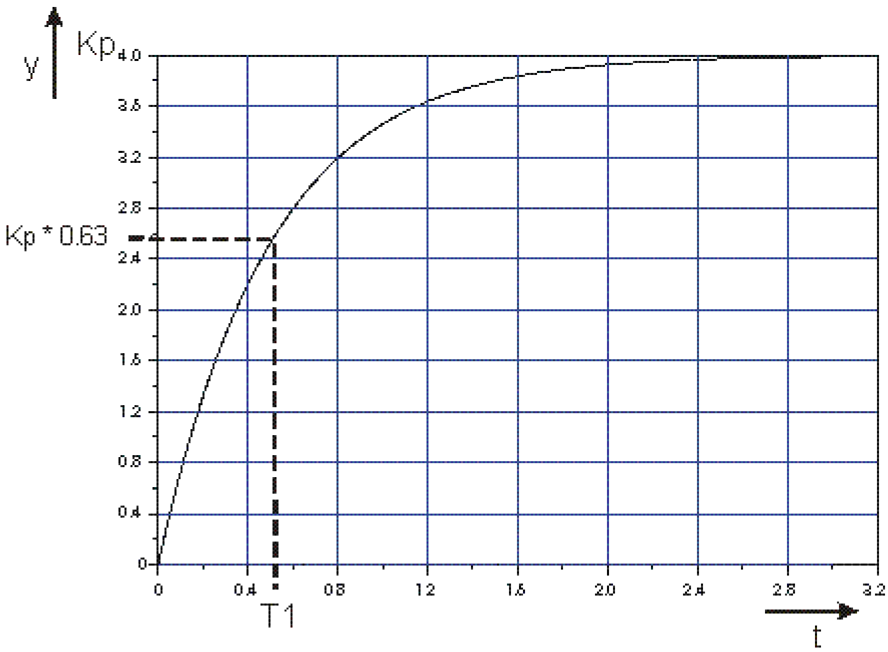


The function block provides a PT1 transfer element in the functional diagram.

Transfer function

$$G(s) = K_p \frac{1}{1 + T_1 s}$$

Step response



Inputs

```
VAR_INPUT
    fIn             : FLOAT;
    fManSyncValue   : FLOAT;
    bSync           : BOOL;
    eMode           : E_CTRL_MODE;
    bHold           : BOOL;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input value of the PT1 element
fManSyncValue	FLOAT	Input value to which the PT1 element can be set, or that is issued at the output in manual mode.
bSync	BOOL	A rising edge at this input sets the PT1-element to the value "fManSyncValue".
eMode	E_CTRL_MODE	Input that specifies the operation mode [▸ 173] of the function block.
bHold	BOOL	A TRUE at this input will hold the internal state (and therefore also the output) constant at its current value, independently of the input value.

Outputs

```
VAR_OUTPUT
    fOut      : FLOAT;
    eState    : E_CTRL_STATE;
    eErrorId  : E_CTRL_ERRORCODES;
    bError    : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output of the PT1 element
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▸ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams         : ST_CTRL_PT1_PARAMS;
END_VAR
```

Name	Type	Description
stParams	ST_CTRL_PT1_PARAMS	Parameter structure of the PT1 element

`stParams` consists of the following elements:

```
TYPE
ST_CTRL_PT1_PARAMS :
STRUCT
    tCtrlCycleTime   : TIME := T#0ms;
    tTaskCycleTime   : TIME := T#0ms;
    fKp              : FLOAT := 0;
    tT1              : TIME := T#0ms;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
fKp	FLOAT	Controller amplification / transfer coefficient
tT1	TIME	time constant

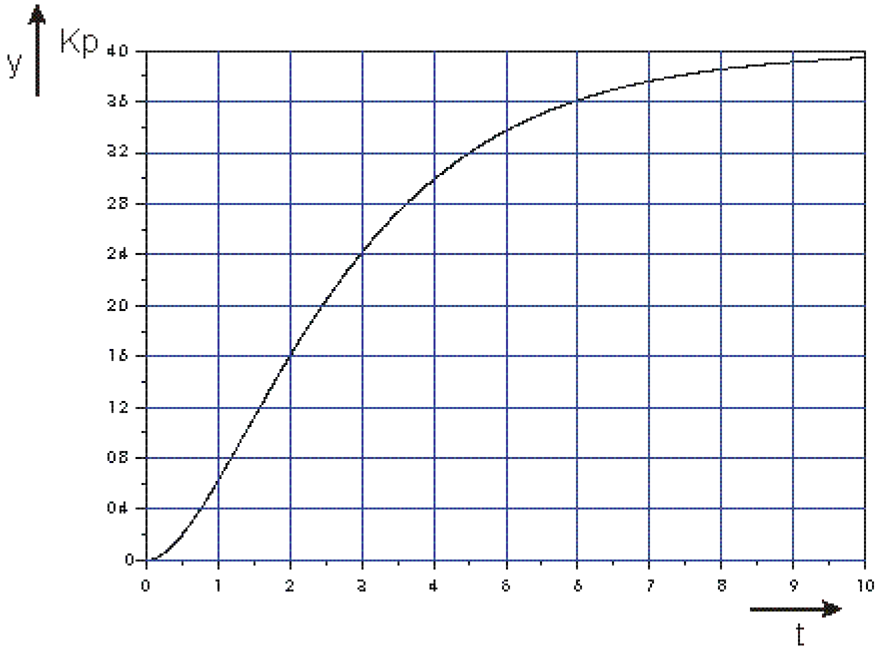
4.2.1.4.10 FB_CTRL_PT2

The function block provides a non-oscillating PT2 transfer element in the functional diagram.

Transfer function

$$G(s) = K_p \frac{1}{1+T_1 s} \frac{1}{1+T_2 s}$$

Step response



Inputs

```
VAR_INPUT
    fIn         : FLOAT;
    fManValue   : FLOAT;
    eMode       : E_CTRL_MODE;
    bHold       : BOOL;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input value of the PT2 element
fManValue	FLOAT	Input value that is output in manual mode.
eMode	E_CTRL_MODE	Input that specifies the operation mode [▸ 173] of the function block.
bHold	BOOL	A TRUE at this input will hold the internal state (and therefore also the output) constant at its current value, independently of the input value.

Outputs

```
VAR_OUTPUT
    fOut       : FLOAT;
    eState     : E_CTRL_STATE;
    eErrorId   : E_CTRL_ERRORCODES;
    bError     : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output of the PT2 element
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERRORCODES	Supplies the error number [▸ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams         : ST_CTRL_PT2_PARAMS;
END_VAR
```

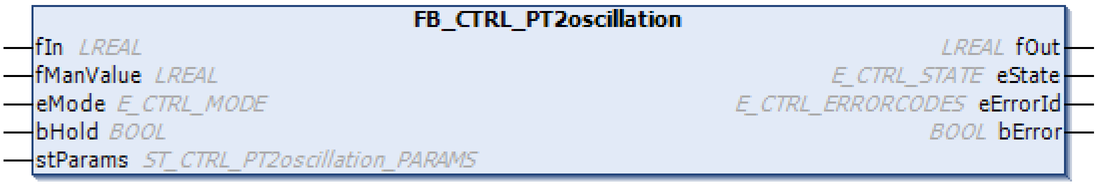
Name	Type	Description
stParams	ST_CTRL_PT2_ PARAMS	Parameter structure of the PT2 element

`stParams` consists of the following elements:

```
TYPE
ST_CTRL_PT2_PARAMS :
STRUCT
    tCtrlCycleTime  : TIME := T#0ms;
    tTaskCycleTime  : TIME := T#0ms;
    fKp             : FLOAT := 0;
    tT1             : TIME := T#0ms;
    tT2             : TIME := T#0ms;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
fKp	FLOAT	Controller amplification / transfer coefficient
tT1	TIME	Time constant T1
tT2	TIME	Time constant T2

4.2.1.4.11 FB_CTRL_PT2oscillation

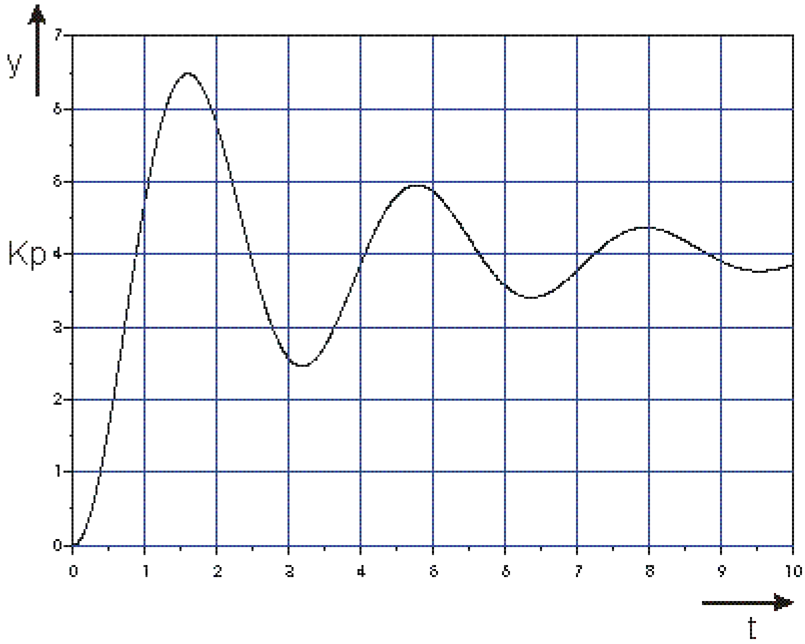


The function block provides an oscillating PT2 transfer element in the functional diagram.

Transfer function

$$G(s) = K_p \frac{1}{1 + 2\vartheta T_0 s + T_0^2 s^2}$$

Step response



Inputs

```
VAR_INPUT
    fIn         : FLOAT;
    fManValue   : FLOAT;
    eMode       : E_CTRL_MODE;
    bHold       : BOOL;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input value of the oscillating PT2 element
fManValue	FLOAT	Input value that is output in manual mode.
eMode	E_CTRL_MODE	Input that specifies the operation mode [▸ 173] of the function block.
bHold	BOOL	A TRUE at this input will hold the internal state (and therefore also the output) constant at its current value, independently of the input value.

Outputs

```
VAR_OUTPUT
    fOut       : FLOAT;
    eState     : E_CTRL_STATE;
    eErrorId   : E_CTRL_ERRORCODES;
    bError     : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output of the PT2 element
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERRORCODES	Supplies the error number [▸ 173] when the output bError is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams        : ST_CTRL_PT2oscillation_PARAMS;
END_VAR
```

Name	Type	Description
stParams	ST_CTRL_PT2 oscillation_ PARAMS	Parameter structure of the oscillating PT2 element

`stParams` consists of the following elements:

```
TYPE
ST_CTRL_PT2oscillation_PARAMS :
STRUCT
    tCtrlCycleTime   : TIME := T#0ms;
    tTaskCycleTime   : TIME := T#0ms;
    fKp              : FLOAT := 0;
    fTheta           : FLOAT := 0;
    tT0              : TIME := T#0ms;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
fKp	FLOAT	Proportional gain
fTheta	FLOAT	Damping ratio
tT0	TIME	Characteristic time

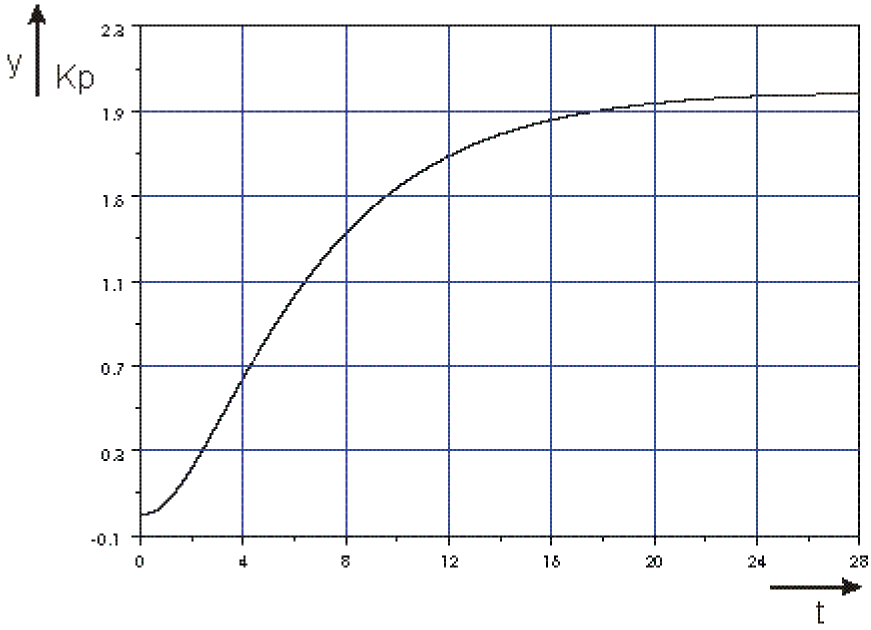
4.2.1.4.12 FB_CTRL_PT3

The function block provides a non-oscillating PT3 transfer element in the functional diagram.

Transfer function

$$G(s) = K_p \frac{1}{1+T_1 s} \frac{1}{1+T_2 s} \frac{1}{1+T_3 s}$$

Step response



Inputs

```
VAR_INPUT
    fIn         : FLOAT;
    fManValue   : FLOAT;
    eMode       : E_CTRL_MODE;
    bHold       : BOOL;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input value of the PT3 element
fManValue	FLOAT	Input value that is output in manual mode.
eMode	E_CTRL_MODE	Input that specifies the operation mode [▸ 173] of the function block.
bHold	BOOL	A TRUE at this input will hold the internal state (and therefore also the output) constant at its current value, independently of the input value.

Outputs

```
VAR_OUTPUT
    fOut       : FLOAT;
    eState     : E_CTRL_STATE;
    eErrorId   : E_CTRL_ERRORCODES;
    bError     : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output of the PT3 element
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERRORCODES	Supplies the error number [▸ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams         : ST_CTRL_PT3_PARAMS;
END_VAR
```

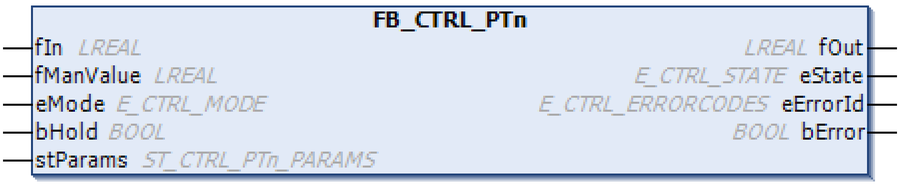
Name	Type	Description
stParams	ST_CTRL_PT3_PARAMS	Parameter structure of the PT3 element

stParams consists of the following elements:

```
TYPE
ST_CTRL_PT3_PARAMS :
STRUCT
    tCtrlCycleTime   : TIME := T#0ms;
    tTaskCycleTime   : TIME := T#0ms;
    fKp              : FLOAT := 0;
    tT1              : TIME := T#0ms;
    tT2              : TIME := T#0ms;
    tT3              : TIME := T#0ms;
END_STRUCT
END_TYPE>
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
fKp	FLOAT	Controller amplification / transfer coefficient
tT1	TIME	Time constant T1
tT2	TIME	Time constant T2p
tT3	TIME	Time constant T3

4.2.1.4.13 FB_CTRL_PTn

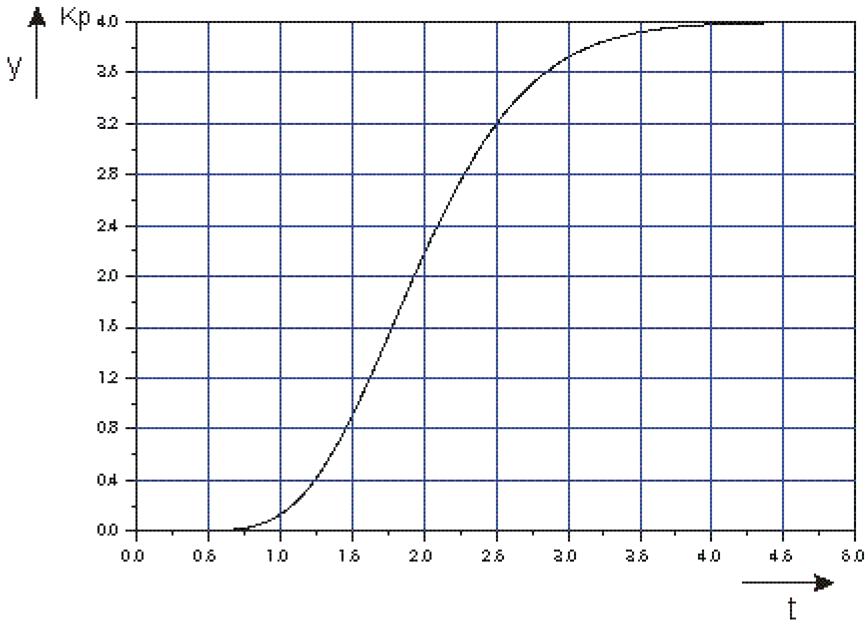


This function block provides a non-oscillating PTn transfer element with "n<= 10" and equal time constants in the functional diagram.

Transfer function

$$G(s) = K_p \frac{1}{(1+T_1 s)^n}$$

Step response with n=10



Inputs

```
VAR_INPUT
    fIn         : FLOAT;
    fManValue   : FLOAT;
    eMode       : E_CTRL_MODE;
    bHold       : BOOL;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input value of the PTn element
fManValue	FLOAT	Input value that is output in manual mode.
eMode	E_CTRL_MODE	Input that specifies the operation mode [▶ 173] of the function block.
bHold	BOOL	A TRUE at this input will hold the internal state (and therefore also the output) constant at its current value, independently of the input value.

Outputs

```
VAR_OUTPUT
    fOut         : FLOAT;
    eState       : E_CTRL_STATE;
    eErrorId     : E_CTRL_ERRORCODES;
    bError       : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output of the PTn element
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▶ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams         : ST_CTRL_PTn_PARAMS;
END_VAR
```

Name	Type	Description
stParams	ST_CTRL_PTn_ PARAMS	Parameter structure of the PTn element

`stParams` consists of the following elements:

```
TYPE
ST_CTRL_PTn_PARAMS :
STRUCT
    tCtrlCycleTime     : TIME := T#0ms;
    tTaskCycleTime     : TIME := T#0ms;
    fKp                : FLOAT := 0;
    tT1                : TIME := T#0ms;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
fKp	FLOAT	Controller amplification / transfer coefficient
tT1	TIME	Time constant T1

4.2.1.4.14 FB_CTRL_PTt

The function block provides an PTt transfer element in the functional diagram.

Transfer function

$$G(s) = K_p \cdot e^{-T_t s}$$

This function block contains internally an array of 500 elements with which the input values can be delayed. When using the `tCtrlCycleTime` this results in a maximum delay of "500 * tCtrlCycleTime". If this maximum delay is insufficient, the sampling time is extended internally to make it possible to reach the requested dead time. It should, however, be remembered that this process involves increasing the time between the discretization steps. If a new sampling time has been calculated, this is indicated by a TRUE on the `bSampleRateChanged` output.

Inputs

```
VAR_INPUT
    fIn            : FLOAT;
    fManSyncValue  : FLOAT;
    bSync          : BOOL;
    eMode          : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input value of the PTt element
fManSyncValue	FLOAT	Input value to which the PTt element can be set, or that is issued at the output in manual mode.
bSync	BOOL	A rising edge at this input sets the PTt element to the value "fManSyncValue".
eMode	E_CTRL_MODE	Input that specifies the operation mode [▸ 173] of the function block.

Outputs

```
VAR_OUTPUT
    fOut                  : FLOAT;
    bSampleRateChanged    : BOOL;
    eErrorId              : E_CTRL_ERRORCODES;
    bError                : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output of the PTt element
bSampleRate Changed	BOOL	Output that indicates whether the function block has internally reduced the sampling rate because of the array being used to delay the input signal not otherwise providing sufficient room.
eErrorId	E_CTRL_ ERRORCODES	Supplies the error number [▸ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams       : ST_CTRL_PTt_PARAMS;
END_VAR
```

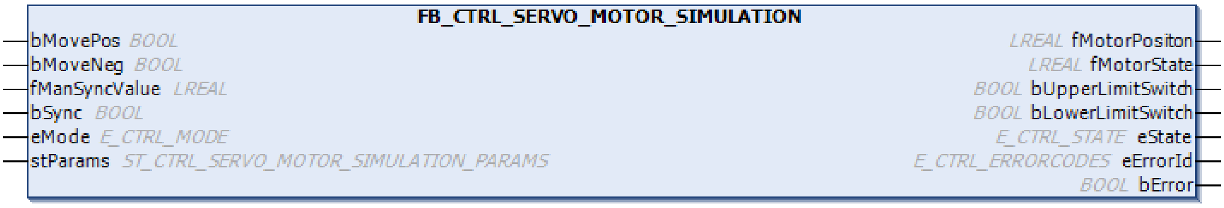
Name	Type	Description
stParams	ST_CTRL_PTt_ PARAMS	Parameter structure of the PTt element

`stParams` consists of the following elements:

```
TYPE ST_CTRL_PTt_PARAMS :
STRUCT
    tCtrlCycleTime   : TIME := T#0ms;
    tTaskCycleTime   : TIME := T#0ms;
    fKp              : FLOAT := 0;
    tTt              : TIME := T#0ms;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
fKp	FLOAT	Controller amplification / transfer coefficient
tTt	TIME	Dead time

4.2.1.4.15 FB_CTRL_SERVO_MOTOR_SIMULATION



The behavior of an actuator can be simulated with this function block.

Behavior of the output

Inputs

```
VAR_INPUT
    bMovePos        : BOOL;
    bMoveNeg        : BOOL;
    fManSyncValue   : FLOAT;
    bSync           : BOOL;
    eMode           : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
bMovePos	BOOL	Input that moves the simulated actuator in the positive direction.
bMoveNeg	BOOL	Input that moves the simulated actuator in the negative direction.
fManSyncValue	FLOAT	Input with which the simulated motor position can be set, or the value to which movement takes place in manual mode.
bSync	BOOL	A rising edge at this input sets the simulated motor position to the value "fManSyncValue".
eMode	E_CTRL_MODE	Input that specifies the operation mode [▶ 173] of the function block.

Outputs

```
VAR_OUTPUT
    fMotorPositon       : FLOAT;
    fMotorState         : FLOAT;
    bUpperLimitSwitch   : BOOL;
    bLowerLimitSwitch   : BOOL;
    eState              : E_CTRL_STATE;
    eErrorId            : E_CTRL_ERRORCODES;
    bError              : BOOL;
END_VAR
```

Name	Type	Description
fMotorPositon	FLOAT	Simulated motor position in the range [fMovingRangeMin ... fMovingRangeMax]
fMotorState	FLOAT	Simulated motor position in the range [0 … 100.0]
bUpperLimit Switch	BOOL	Simulated limit switch at the actuator's positive stop
bLowerLimit Switch	BOOL	Simulated limit switch at the actuator's negative stop.
eState	E_CTRL_ STATE	State of the function block
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▶ 173] when the output bError is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams        : ST_CTRL_SERVO_MOTOR_SIMULATION_PARAMS;
END_VAR
```

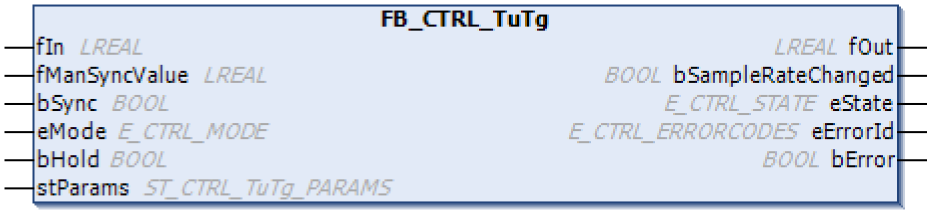
Name	Type	Description
stParams	ST_CTRL_ SERVO_MOTOR_ SIMULATION_ PARAMS	Parameter structure of the function block

stParams consists of the following elements:

```
TYPE ST_CTRL_SERVO_MOTOR_SIMULATION_PARAMS:
STRUCT
    tCtrlCycleTime    : TIME := T#0ms;
    tTaskCycleTime    : TIME := T#0ms;
    fMovingRangeMin   : FLOAT := 0;
    fMovingRangeMax   : FLOAT := 0;
    tMovingTime       : TIME := T#0ms;
    tDeadTime         : TIME := T#0ms;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
fMovingRange Min	FLOAT	Minimum position of the simulated actuator
fMovingRange Max	FLOAT	Maximum position of the simulated actuator
tMovingTime	TIME	The time required to move the simulated actuator from one stop to the other.
tDeadTime	TIME	Dead time of the simulated actuator

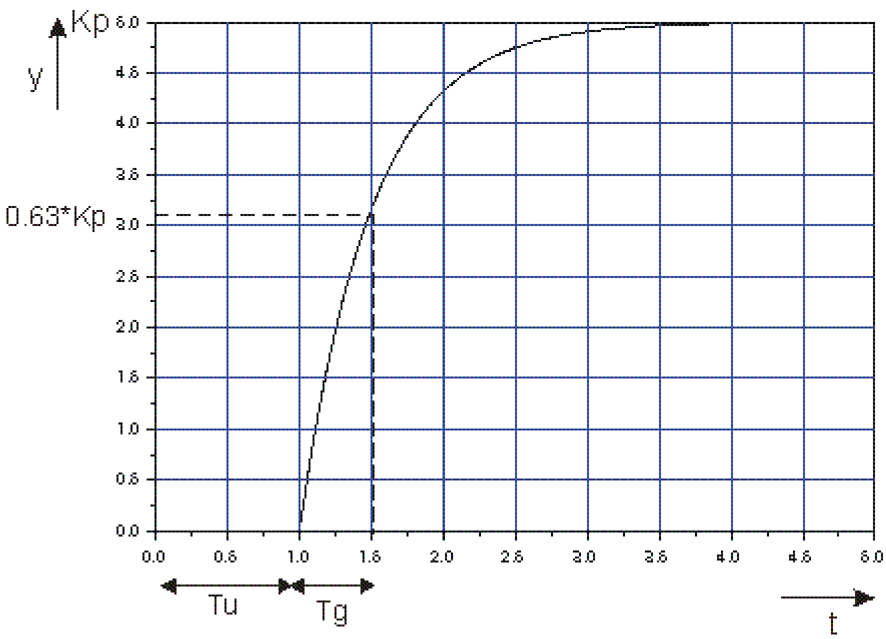
4.2.1.4.16 FB_CTRL_TuTg



The function block provides a TuTg transfer element (a dead time delay element) in the functional diagram.

Transfer function

$$G(s) = K_p \frac{1}{1+T_g s} \cdot e^{-T_u s}$$



Inputs

```
VAR_INPUT
    fIn                 : FLOAT;
    fManSyncValue       : FLOAT;
    bSync               : BOOL;
    eMode               : E_CTRL_MODE;
    bHold               : BOOL;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input value of the TuTg element
fManSyncValue	FLOAT	Input value to which the TuTg element can be set, or that is issued at the output in manual mode.
bSync	BOOL	A rising edge at this input sets the TuTg element to the value "fManSyncValue".
eMode	E_CTRL_MODE	Input that specifies the operation mode [▸ 173] of the function block.
bHold	FLOAT	A TRUE at this input will hold the internal state (and therefore also the output) constant at its current value, independently of the input value.

Outputs

```
VAR_OUTPUT
    fOut                    : FLOAT;
    bSampleRateChanged      : BOOL;
    eState                  : E_CTRL_STATE;
    eErrorId                : E_CTRL_ERRORCODES;
    bError                  : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output of the TuTg element
bSampleRate Changed	BOOL	Output that indicates whether the function block has internally reduced the sampling rate because of the array being used to delay the input signal not otherwise providing sufficient room.
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ ERRORCODES	Supplies the error number [▸ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams      : ST_CTRL_TuTg_PARAMS;
END_VAR
```

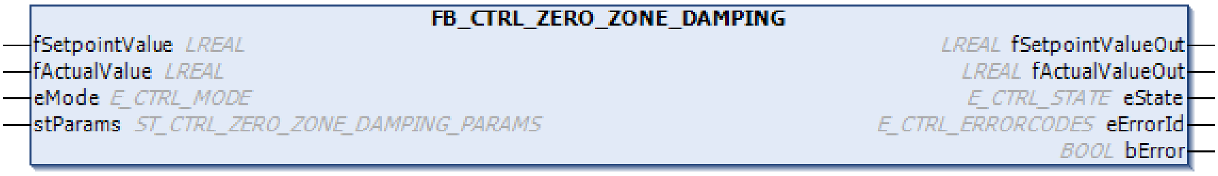
Name	Type	Description
stParams	ST_CTRL_TuTg_ PARAMS	Parameter structure of the TuTg element

`stParams` consists of the following elements:

```
TYPE ST_CTRL_TuTg_PARAMS :
STRUCT
    tCtrlCycleTime      : TIME  := T#0ms;
    tTaskCycleTime      : TIME  := T#0ms;
    fKp                 : FLOAT := 0;
    tTu                 : TIME  := T#0ms;
    tTg                 : TIME  := T#0ms;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
fKp	FLOAT	Controller amplification / transfer coefficient
tTu	TIME	Dead time
tTg	TIME	time constant

4.2.1.4.17 FB_CTRL_ZERO_ZONE_DAMPING

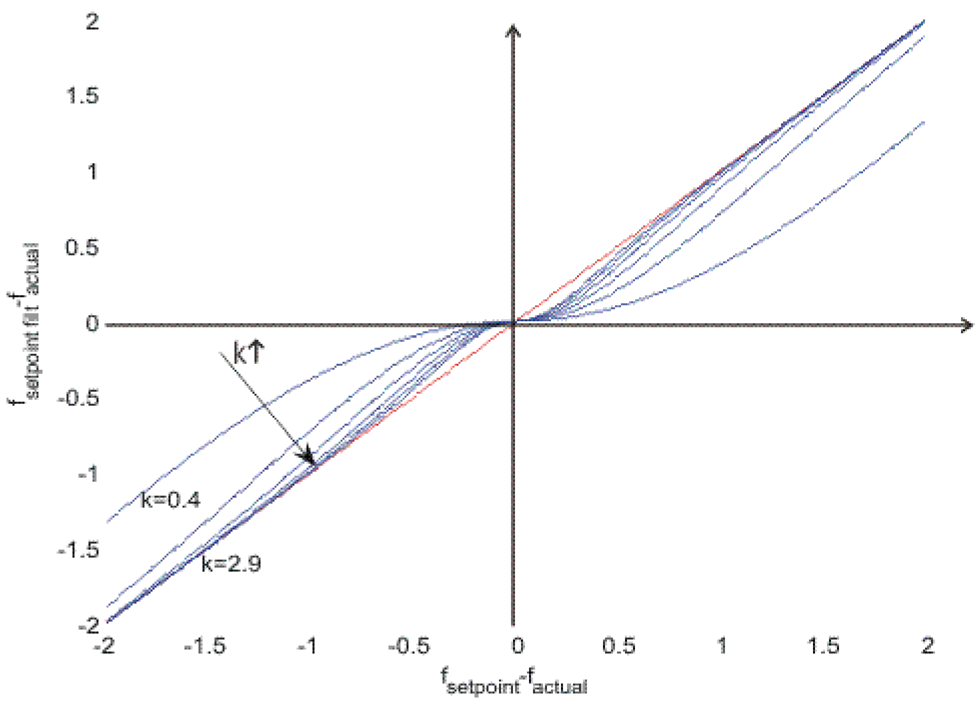


This function block enables zero point damping to be realized in order to minimize control interventions in the range | actual value - setpoint | < ε.

Transfer behavior in the time range

$$f_{setpoint\_out} = (f_{setpoint\_in} - f_{actual\_in}) \cdot \tanh(|f_{setpoint\_in} - f_{actual\_in}| \cdot k_{damping}) + f_{actual\_in}$$

$$f_{actual\_out} = f_{setpoint\_in}$$



Inputs

```
VAR_INPUT
    fSetpointValue      : FLOAT;
    fActualValue        : FLOAT;
    eMode               : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
fSetpointValue	FLOAT	Setpoint of the controlled variable
fActualValue	FLOAT	Actual value of the controlled variable
eMode	E_CTRL_MODE	Input that specifies the operation mode [▸ 173] of the function block.

Outputs

```
VAR_OUTPUT
    fSetpointValueOut     : FLOAT;
    fActualValueOut       : FLOAT;
    eState                : E_CTRL_STATE;
    eErrorId              : E_CTRL_ERRORCODES;
    bError                : BOOL;
END_VAR
```

Name	Type	Description
fSetpointValue Out	FLOAT	Filtered setpoint to controller
fActualValueOut	FLOAT	Actual value to controller
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▸ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams         : ST_CTRL_ZERO_ZONE_DAMPING_PARAMS;
END_VAR
```

Name	Type	Description
stParams	ST_CTRL_ZERO_ ZONE_ DAMPING_ PARAMS	Parameter structure of the transfer element

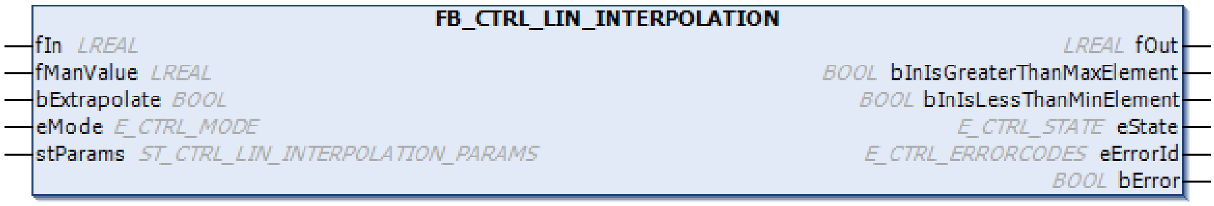
`stParams` consists of the following elements:

```
TYPE
ST_CTRL_PI_PST_CTRL_ZERO_ZONE_DAMPING_PARAMS :
STRUCT
    tCtrlCycleTime         : TIME  := T#0ms;
    tTaskCycleTime         : TIME  := T#0ms;
    fDampingCoefficient    : FLOAT := 0.0;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
fDampingCo efficient	FLOAT	The parameter corresponds to $k_{damping}$ in the transfer function.

4.2.1.5 Interpolation

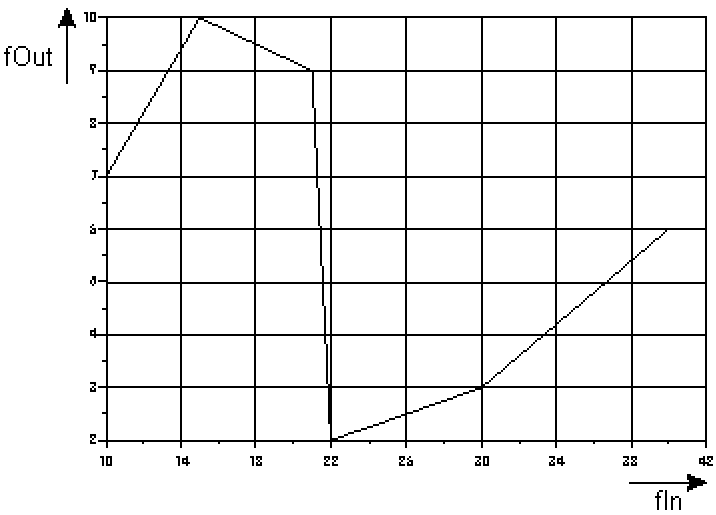
4.2.1.5.1 FB_CTRL_LIN_INTERPOLATION



This function block performs linear interpolation to obtain values on the basis of a sampling points table.

Behavior of the output

fIn	fOut
arrTable[1,1] := 10;	arrTable[1,2] := 7;
arrTable[2,1] := 15;	arrTable[2,2] := 10;
arrTable[3,1] := 21;	arrTable[3,2] := 9;
arrTable[4,1] := 22;	arrTable[4,2] := 2;
arrTable[5,1] := 30;	arrTable[5,2] := 3;
arrTable[6,1] := 40;	arrTable[6,2] := 6;



Inputs

```
VAR_INPUT
    fIn            : FLOAT;
    fManValue      : FLOAT;
    bExtrapolate   : BOOL;
    eMode          : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input value for the interpolation function block
fManValue	FLOAT	Input value that is output in manual mode.
bExtrapolate	BOOL	If this input is FALSE, then the value of the last interpolation point is output if the limits of the table are exceeded. If, however, it is TRUE, then extrapolation is performed on the basis of the last two interpolation points.
eMode	E_CTRL_MODE	Input that specifies the operation mode [▶ 173] of the function block.

Outputs

```
VAR_OUTPUT
    fOut                          : FLOAT;
    bInIsGreaterThanMaxElement    : BOOL;
    bInIsLessThanMinElement       : BOOL;
    eState                        : E_CTRL_STATE;
    eErrorId                      : E_CTRL_ERRORCODES;
    bError                        : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Table value from linear interpolation
bInIsGreater ThanMax Element	BOOL	A TRUE at this output indicates that the input value is greater than the largest interpolation point.
bInIsLessThan Min Element	BOOL	A TRUE at this output indicates that the input value is smaller than the smallest interpolation point.
eState	E_CTRL_ STATE	State of the function block
eErrorId	E_CTRL_ ERRORCODES	Supplies the error number [▶ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams         : ST_CTRL_LIN_INTERPOLATION_PARAMS;
END_VAR
```

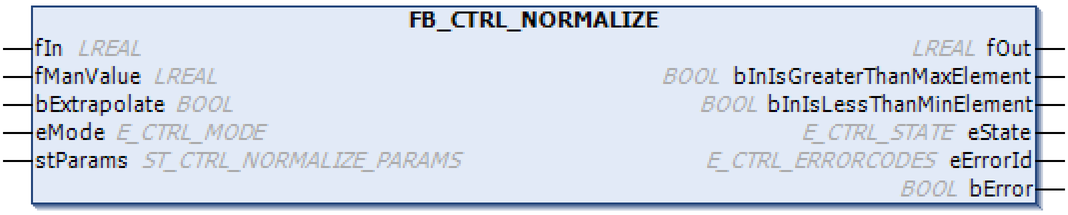
Name	Type	Description
stParams	ST_CTRL_LIN_IN TERPOLATION_ PARAMS	Parameter structure of the interpolation element

`stParams` consists of the following elements:

```
TYPE ST_CTRL_2POINT_PARAMS :
STRUCT
    tCtrlCycleTime            : TIME := T#0ms;
    tTaskCycleTime            : TIME := T#0ms;
    pDataTable_ADR            : POINTER TO FLOAT := 0;
    nDataTable_SIZEOF         : UINT     := 0;
    nDataTable_NumberOfRows   : UINT     := 0;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
pDataTable_ADR	POINTER TO FLOAT	Address of the n x 2 array on which linear interpolation is to be carried out.
nDataTable_SIZEOF	UINT	Size of the n x 2 array
nDataTable_NumberOfRows	UINT	Number of rows in the array

4.2.1.5.2 FB_CTRL_NORMALIZE

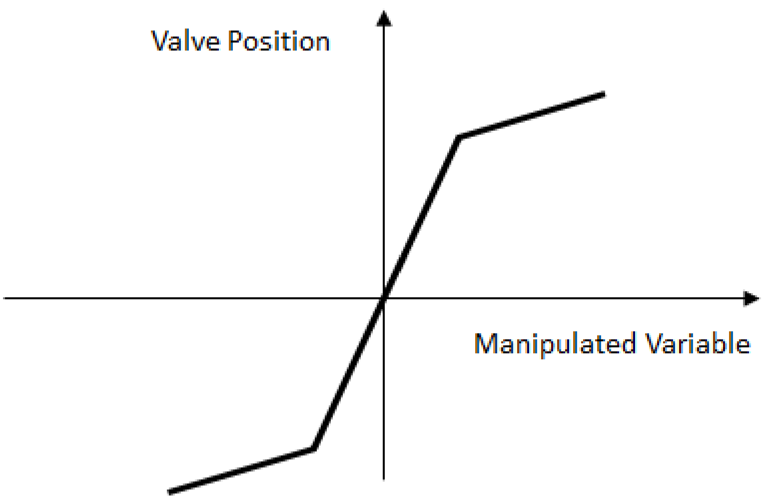


This function block can be used to linearize a non-linear transfer element, with the aid of an inverse characteristic curve.

The characteristic curve of the transfer element that is to be linearized is stored in the table associated with this function block. The function block uses this to calculate the inverse characteristic curve with which the linearization can be carried out.

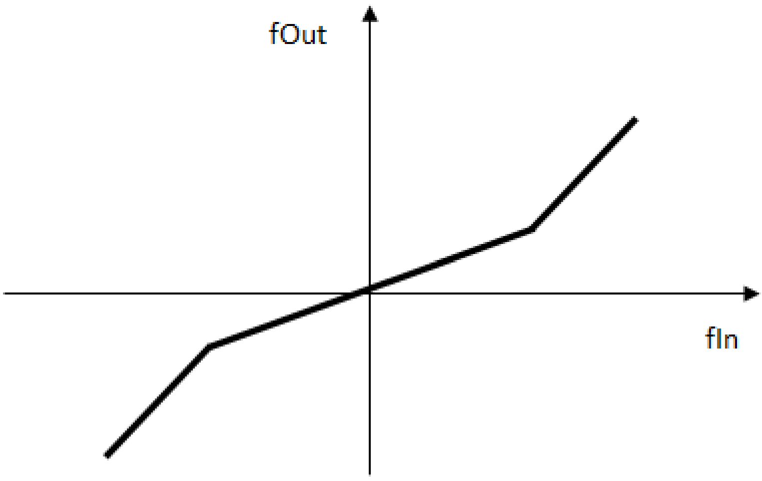
Sample

The following valve characteristic curve is stored in the table with 4 interpolation points.

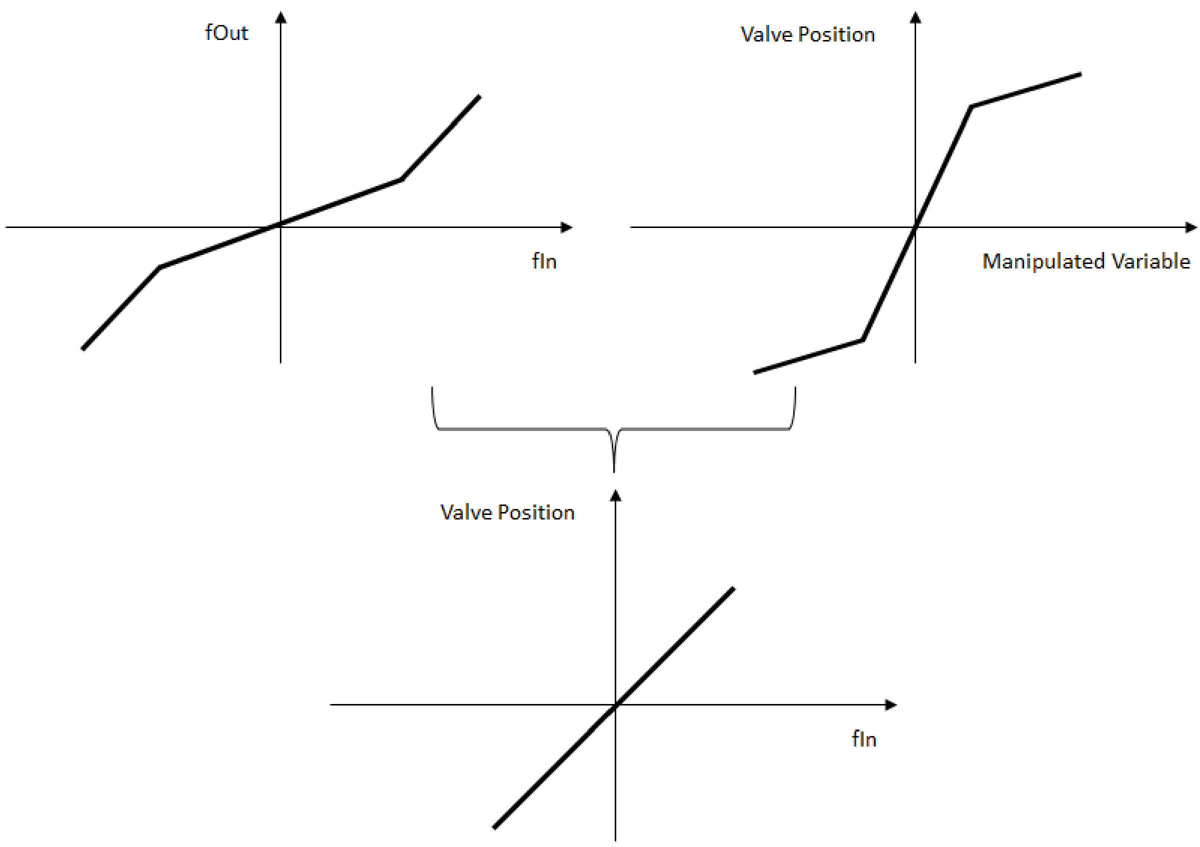


Control value	Valve position
arrTable[1,1] := -6;	arrTable[1,2] := -100;
arrTable[2,1] := -1;	arrTable[2,2] := -70;
arrTable[3,1] := 1;	arrTable[3,2] := 70;
arrTable[4,1] := 6;	arrTable[4,2] := 100;

The inverse characteristic curve is calculated from this characteristic curve:



In the ideal case, applying these two characteristic curves in series will result in a linear transfer behavior.



Inputs

```
VAR_INPUT
    fIn            : FLOAT;
    fManValue      : FLOAT;
    bExtrapolate   : BOOL;
    eMode          : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input value
fManValue	FLOAT	Input value that is output in manual mode.
bExtrapolate	BOOL	If this input is FALSE, then the value of the last interpolation point is output if the limits of the table are exceeded. If, however, it is TRUE, then extrapolation is performed on the basis of the last two interpolation points.
eMode	E_CTRL_MODE	Input that specifies the operation mode [▶ 173] of the function block.

Outputs

```
VAR_OUTPUT
    fOut                         : FLOAT;
    bInIsGreaterThanMaxElement   : BOOL;
    bInIsLessThanMinElement      : BOOL;
    eState                       : E_CTRL_STATE;
    eErrorId                     : E_CTRL_ERRORCODES;
    bError                       : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Table value from linear interpolation
bInIsGreaterThanMaxElement	BOOL	A TRUE at this output indicates that the input value is greater than the largest interpolation point.
bInIsLessThanMinElement	BOOL	A TRUE at this output indicates that the input value is smaller than the smallest interpolation point.
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERRORCODES	Supplies the error number [▶ 173] when the output bError is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams         : ST_CTRL_NORMALIZE_PARAMS;
END_VAR
```

Name	Type	Description
stParams	ST_CTRL_NORMALIZE_PARAMS	Parameter structure of the function block

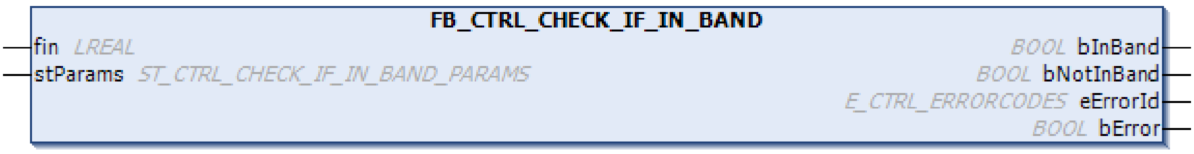
stParams consists of the following elements:

```
TYPE ST_CTRL_NORMALIZE_PARAMS:
STRUCT
    tCtrlCycleTime            : TIME := T#0ms;
    tTaskCycleTime            : TIME := T#0ms;
    pDataTable_ADR            : POINTER TO FLOAT := 0;
    nDataTable_SIZEOF         : UINT     := 0;
    nDataTable_NumberOfRows   : UINT     := 0;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
pDataTable_ ADR	POINTER TO FLOAT	Address of the n x 2 array on which linear interpolation is to be carried out.
nDataTable_ SIZEOF	UINT	Size of the n x 2 array
nDataTable_ NumberOfRows	UINT	Number of rows in the array

4.2.1.6 Monitoring / Alarming

4.2.1.6.1 FB_CTRL_CHECK_IF_IN_BAND



The function block monitors whether the input value is within the range [fMin ... fMax], i.e. whether the inequality

$$fMin \leq fIn \leq fMax$$

is fulfilled.

Inputs

```
VAR_INPUT
    fIn     : FLOAT;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input value that is monitored.

Outputs

```
VAR_OUTPUT
    bInBand      : BOOL;
    bNotInBand   : BOOL;
    eErrorId     : E_CTRL_ERRORCODES;
    bError       : BOOL;
END_VAR
```

Name	Type	Description
bInBand	BOOL	A TRUE at this output indicates that the input value is within the specified range.
bNotInBand	BOOL	A TRUE at this output indicates that the input value is **not** within the specified range.
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▸ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams          : ST_CTRL_CHECK_IF_IN_BAND_PARAMS;
END_VAR
```

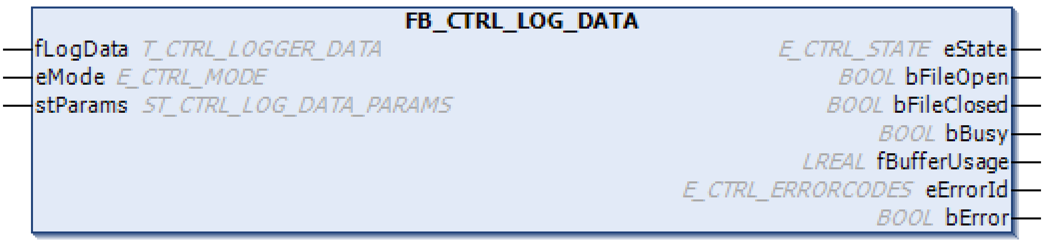
Name	Type	Description
stParams	ST_CTRL_ CHECK_IF_IN_ BAND_PARAMS	Parameter structure of the function block

stParams consists of the following elements:

```
TYPE ST_CTRL_CHECK_IF_IN_BAND_PARAMS:
STRUCT
    tCtrlCycleTime   : TIME := T#0ms;
    tTaskCycleTime   : TIME := T#0ms;
    fMin             : FLOAT;
    fMax             : FLOAT;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
fMin	FLOAT	Lower limit of the range
fMax	FLOAT	Upper limit of the range

4.2.1.6.2 FB_CTRL_LOG_DATA



This function block allows a log file to be created in *.csv format (comma separated values), in which a maximum of 10 variables may be recorded. The column headings specified by the user are written in the first row of this file. The input data is written at equal time intervals in the following lines. The individual entries are separated by a comma. The time interval between the entries is specified in the `tLogCycleTime` parameter. If, for instance, `tLogCycleTime := T\#2s` is chosen, then an entry is made in the file every 2s. The files that were generated can be analyzed with a spreadsheet program, for example.

The time stamp of the log entry, in s, is stored in the first column of the file. The other columns contain the data of the function block input `fLogData`.

When the mode is set to `eCTRL_MODE_ACTIVE` the log file is opened and entries are written into the file. The file remains open until the function block's mode is set to `eCTRL_MODE_PASSIVE`.

It is essential that the file is closed by switching into `eCTRL_MODE_PASSIVE` before attempting to analyze the log file. If this is not done, it is possible that not all the entries will be written into the file.

The function block makes it possible to work with or without an external buffer.

Operating without an external buffer:	The `bBusy` output is TRUE when the logging of a row has been started. The following data set will not be logged until the `bBusy` output is FALSE again. The `fBufferUsage` output indicates how full the internal buffer is.
Operating with an external buffer:	The user must create a buffer larger than 255 bytes and with the type `ARRAY OF BYTES`. The individual messages are temporarily stored in the external buffer, and this buffer is written into the file as quickly as possible. The `fBufferUsage` output indicates how full the buffer is. The function block is stopped and an error is output if the buffer overflows.

The external buffer is used if a buffer address and a buffer size greater than zero are parameterized. In the absence of an external buffer, an internal buffer with the size of 255 bytes is used.

Inputs

```
VAR_INPUT
    fLogData    : T_CTRL_LOGGER_DATA;
    eMode       : E_CTRL_MODE;
END_VAR

VAR_GLOBAL CONSTANT
    nCTRL_LOGGER_DATA_ARRAY_SIZE   : UINT := 10;
END_VAR

TYPE T_CTRL_LOGGER_DATA :
    ARRAY [1..nCTRL_LOGGER_DATA_ARRAY_SIZE] OF FLOAT;
END_TYPE
```

Name	Type	Description
fLogData	T_CTRL_LOGGER_DATA	Array containing the values that are to be written into the log file.
eMode	E_CTRL_MODE	Input that specifies the operation mode [▸ 173] of the function block.

Outputs

```
VAR_OUTPUT
    eState          : E_CTRL_STATE;
    bFileOpen       : BOOL
    bFileClosed     : BOOL
    fBufferUsage    : FLOAT
    bBusy           : BOOL
    eErrorId        : E_CTRL_ERRORCODES;
    bError          : BOOL;
END_VAR
```

Name	Type	Description
eState	E_CTRL_STATE	State of the function block
bFileOpen	BOOL	A TRUE at this output indicates that the file has successfully been opened.
bFileClosed	BOOL	A TRUE at this output indicates that the file has successfully been closed.
fBufferUsage	FLOAT	Current fill level of the external buffer as a percentage
bBusy	BOOL	A TRUE at this output indicates that logging a row is active.
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▶ 173] when the output bError is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams        : ST_CTRL_LOG_DATA_PARAMS;
END_VAR
```

Name	Type	Description
stParams	ST_CTRL_LOG_DATA_PARAMS	Parameter structure of the logging function block

stParams consists of the following elements:

```
TYPE ST_CTRL_LOG_DATA_PARAMS:
STRUCT
    tLogCycleTime              : TIME := T#0ms;
    tTaskCycleTime             : TIME := T#0ms;
    sFileName                  : STRING;
    sNetId                     : T_AmsNetId := '';
    tFileOperationTimeou       : TIME := T#3s;
    nNumberOfColumn            : INT(1..10);
    arColumnHeadings           : ARRAY [1..10] OF STRING;
    bAppendData                : BOOL := FALSE;
    bWriteTimeStamps           : BOOL := TRUE;
    bWriteColumnHeadings       : BOOL := TRUE;
    bWriteAbsoluteTimeStamps   : BOOL := FALSE
    pLogBuffer_ADR             : POINTER TO BYTE;
    nLogBuffer_SIZEOF          : UDINT;
END_STRUCT
END_TYPE
```

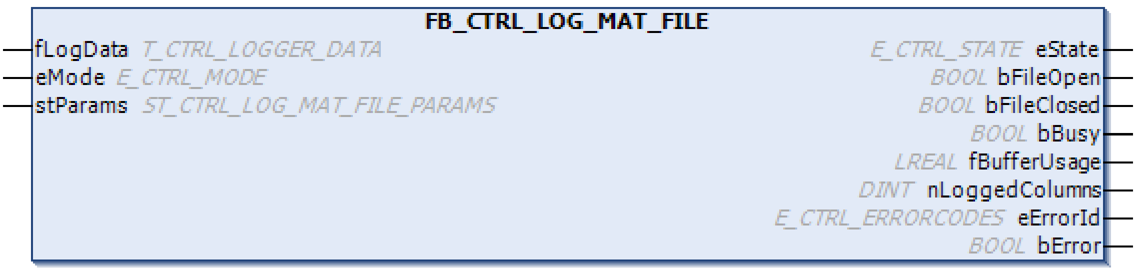
Name	Type	Description
tLogCycleTime	TIME	Cycle time with which entries are written into the log file. This must be greater than or equal to the **TaskCycleTime**.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
sFileName	STRING	Name and path of the log file, for example: d:\Logfile.csv
sNetId	T_AmsNetId	The log file is written to the system with the NetId specified here.
tFileOperation Timeou	TIME	Timeout for all file operations
nNumberOf Column	INT	Number of columns written into the file (maximum 10)
arColumn Headings	ARRAY	Array of strings that contain the column headings.
bAppendData	BOOL	If this parameter is TRUE, then new data sets are appended when a file is opened again. Otherwise the file is overwritten **without query**, and this will delete any content that already exists.
bWriteTime Stamps	BOOL	If this parameter is set to TRUE, the time stamp of the measurement is written into the first column of the file.
bWriteColumn Headings	BOOL	If this parameter is set to TRUE, the column headers are written into the first row of the file.
bWriteAbsolute TimeStamps	BOOL	If this parameter is set to TRUE, the local NT time is used as timestamp. In this case the minimum `LogCycleTime` is 5s!
pLogBuffer_ADR	POINTER TO BYTE	Address of the external `LogBuffer`. To detect a buffer, the address must not be 0.
nLogBuffer_ SIZEOF	UDINT	Size of the `LogBuffer`. The buffer must be an ARRAY OF BYTE with at least 256 elements. The size of the buffer can be optimized with the aid of the `fBufferUsage` output.

NOTICE

Error during file handling

The parameter set can only be changed when the file is closed (bFileClosed = TRUE). Otherwise errors can occur during file handling.

4.2.1.6.3 FB_CTRL_LOG_MAT_FILE



This function block allows a log file to be created in Matlab 5 (*.mat) format, in which a maximum of 10 magnitudes may be recorded.

Two variables are created in the file, a double array and a cell array. The recorded magnitudes are recorded, line-by-line, in the double array. The identifiers of the individual rows are stored in the cell array. The user can specify the name used for the double array in the function block's parameter structure. The name of the cell array is derived from the name of the double array by appending "_Info" to the variable name.

The input data is written at equal time intervals in the columns of the data array. A time stamp for the relevant entry, in s, can be stored in the first column. The time interval between the entries is specified in the `tLogCycleTime` parameter. If, for instance, "tLogCycleTime := T\#2s" is chosen, then an entry is made in the file every 2s.

When the mode is set to `eCTRL_MODE_ACTIVE` the log file is opened and entries are written into the file. The file remains open until the function block's mode is set to `eCTRL_MODE_PASSIVE`.

It is essential that the file is closed by switching into `eCTRL_MODE_PASSIVE` before attempting to analyze the log file. If this is not done, it is possible that not all the entries will be written into the file, which will then not be consistent.

The function block makes it possible to work with or without an external buffer.

Operating without an external buffer:	The bBusy output is TRUE when the logging of a row has been started. The following data set will not be logged until the bBusy output is FALSE again. The `fBufferUsage` output indicates how full the internal buffer is.
Operating with an external buffer:	The user must create a buffer larger than 1500 bytes and with the type `ARRAY OF BYTES`. The individual messages are temporarily stored in the external buffer, and this buffer is written into the file as quickly as possible. The `fBufferUsage` output indicates how full the buffer is. If a buffer overflow occurs, the function block is stopped and an error is output.

The external buffer is used if a buffer address and a buffer size greater than zero are parameterized. In the absence of an external buffer, an internal buffer with the size of 1500 bytes is used.

Inputs

```
VAR_INPUT
    fLogData    : T_CTRL_LOGGER_DATA;
    eMode       : E_CTRL_MODE;
END_VAR
VAR_GLOBAL CONSTANT
    nCTRL_LOGGER_DATA_ARRAY_SIZE :UINT := 10;
END_VAR
TYPE
    T_CTRL_LOGGER_DATA :ARRAY [1..nCTRL_LOGGER_DATA_ARRAY_SIZE]
OF FLOAT;
END_TYPE
```

Name	Type	Description
fLogData	T_CTRL_ LOGGER_DATA	Array containing the values that are to be written into the log file.
eMode	E_CTRL_MODE	Input that specifies the operation mode [▸ 173] of the function block.

Outputs

```
VAR_OUTPUT
    eState          : E_CTRL_STATE;
    bFileOpen       : BOOL
    bFileClosed     : BOOL
    fBufferUsage    : FLOAT
    nLoggedColumns  : DINT;
    bBusy           : BOOL
    eErrorId        : E_CTRL_ERRORCODES;
    bError          : BOOL;
END_VAR
```

Name	Type	Description
eState	E_CTRL_STATE	State of the function block
bFileOpen	BOOL	A TRUE at this output indicates that the file has successfully been opened.
bFileClosed	BOOL	A TRUE at this output indicates that the file has successfully been closed.
fBufferUsage	FLOAT	Current fill level of the external buffer as a percentage
nLogged Columns	DINT	Number of columns written in the file
bBusy	BOOL	A TRUE at this output indicates that logging a row is active.
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▶ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams         : ST_CTRL_LOG_MAT_FILE_PARAMS;
END_VAR
```

Name	Type	Description
stParams	ST_CTRL_LOG_MAT_FILE_PARAMS	Parameter structure of the logging function block

`stParams` consists of the following elements:

```
TYPE ST_CTRL_LOG_MAT_FILE_PARAMS:
STRUCT
    tLogCycleTime           : TIME := T#0ms;
    tTaskCycleTime          : TIME := T#0ms;
    sFileName               : STRING;
    nNumberOfRows           : INT(1..10);
    sMatrixName             : STRING;
    arRowDescription        : ARRAY [1..10] OF STRING(25);
    bWriteTimeStamps        : BOOL := TRUE;
    bWriteRowDescription    : BOOL := TRUE;
    pLogBuffer_ADR          : POINTER TO
     ST_CTRL_MULTIPLE_PWM_OUT_DATA;
    nLogBuffer_SIZEOF         : UDINT;
END_STRUCT
END_TYPE
```

Name	Type	Description
tLogCycleTime	TIME	Cycle time with which entries are written into the log file. This must be greater than or equal to the **TaskCycleTime**.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
sFileName	STRING	Name and path of the log file, for example: d:\Logfile.mat
nNumberOfRows	INT	Number of columns written into the file (maximum 10).
sMatrixName	STRING	Name of the data array
arRow Description	ARRAY	Array of strings that contain the line headings.
bWriteTime Stamps	BOOL	If this parameter is set to TRUE, the timestamp of the measurement is written into the first row of the data array.
bWriteRow Description	BOOL	If this parameter is set to TRUE a cell array is created into which the descriptions of the rows are written.
pLogBuffer_ADR	POINTER TO ST_CTRL_ MULTIPLE_PWM _OUT_DATA	Address of the external `LogBuffer`. To detect a buffer, the address must not be 0.
nLogBuffer_ SIZEOF	UDINT	Size of the `LogBuffer`. The buffer must be an `ARRAY OF BYTE` with at least 1501 elements. The size of the buffer can be optimized with the aid of the `fBufferUsage` output.

NOTICE

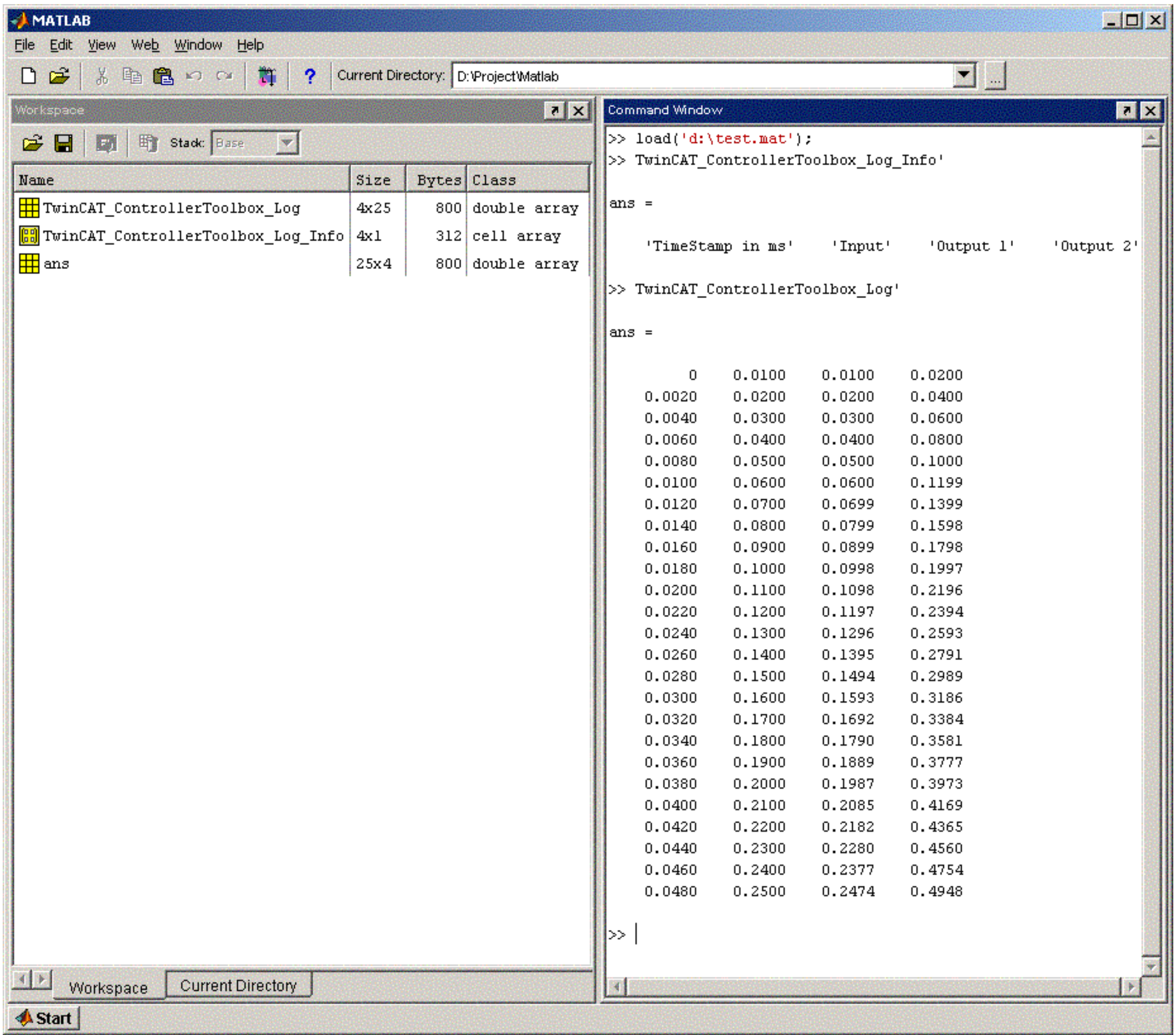
Error during file handling

The parameter set can only be changed when the file is closed (bFileClosed = TRUE). Otherwise errors can occur during file handling.

Sample

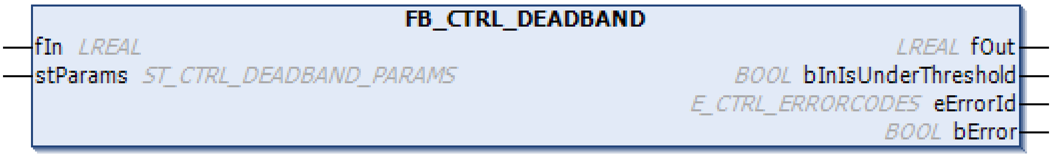
```
PROGRAM PRG_LOG_MAT_FILE_TEST_BUFFERED
VAR
    eMode               : E_CTRL_MODE;
    stParams            : ST_CTRL_LOG_MAT_FILE_PARAMS;
    LoggerData          : T_CTRL_LOGGER_DATA;
    eErrorId            : E_CTRL_ERRORCODES;
    bError              : BOOL;
    fbCtrlLogMatFile    : FB_CTRL_LOG_MAT_FILE;
    LogBuffer           : ARRAY[0..2000] OF BYTE;
    bInit               : BOOL := TRUE;
    fIn                 : LREAL;
    fOut                : LREAL;
    fMaxBufferUsage     : LREAL;
END_VAR

IF bInit THEN
    stCtrl_GLOBAL_CycleTimeInterpretation.bInterpretCycleTimeAsTicks := FALSE;
    stCtrl_GLOBAL_CycleTimeInterpretation.fBaseTime := 0;
    stParams.tLogCycleTime      := T#2ms;
    stParams.tTaskCycleTime         := T#2ms;
    stParams.nNumberOfRows      := 3;
    stParams.sFileName      := 'D:\test.mat';
    stParams.sMatrixName      := 'TwinCAT_ControllerToolbox_Log';
    stParams.arRowDescription[1]  := 'Input';
    stParams.arRowDescription[2]  := 'Output 1';
    stParams.arRowDescription[3]  := 'Output 2';
    stParams.bWriteRowDescription := TRUE;
    stParams.bWriteTimeStamps     := TRUE;
    eMode  := eCTRL_MODE_ACTIVE;
    bInit  := FALSE;
END_IF

stParams.nLogBuffer_SIZEOF := SIZEOF( LogBuffer );
stParams.pLogBuffer_ADR := ADR( LogBuffer );

fIn := fIn + 0.01;
fOut := SIN(fIn);
```

```
LoggerData[1]:= fIn;
LoggerData[2]:= fOut;
LoggerData[3]:= fOut * 2;

IF fbCtrlLogMatFile.nLoggedColumns >= 25 THEN
    eMode := eCTRL_MODE_Passive;
END_IF

fbCtrlLogMatFile( fLogData := LoggerData,
             eMode := eMode,
             stParams :=stParams,
             eErrorId => eErrorId,
             bError => bError);

fMaxBufferUsage := MAX(fbCtrlLogMatFile.fBufferUsage, fMaxBufferUsage);
```



4.2.1.7 Output To Controlling Equipment

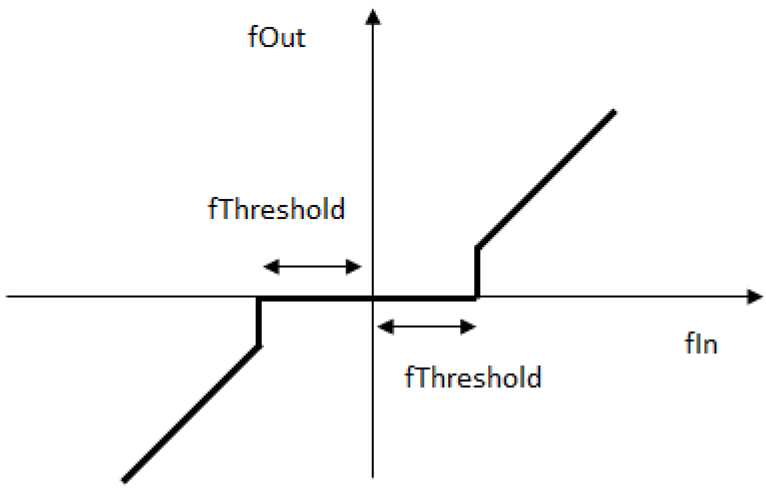
4.2.1.7.1 FB_CTRL_DEADBAND



This function block provides a dead band for the input signal. If the input signal is within the dead band, this is indicated by the `bInIsUnderThreshold` output.

Description of the output behavior

$$fout = \begin{cases} 0.0 : |fin| \leq fThreshold \\ fin : else \end{cases}$$



Inputs

```
VAR_INPUT
    fIn      : FLOAT;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input value

Outputs

```
VAR_OUTPUT
    fOut                  : FLOAT;
    bInIsUnderThreshold   : BOOL;
    eState                : E_CTRL_STATE;
    eErrorId              : E_CTRL_ERRORCODES;
    bError                : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output of the function block
bInIsUnder Threshold	BOOL	A TRUE at this output indicates that the input value is within the dead band.
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▸ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams          : ST_CTRL_DEADBAND_PARAMS;
END_VAR
```

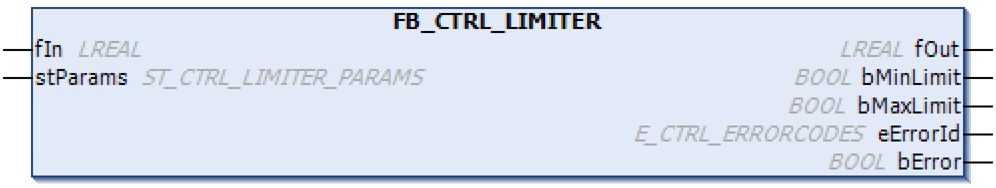
Name	Type	Description
stParams	ST_CTRL_DEAD BAND_PARAMS	Parameter structure of the function block

`stParams` consists of the following elements:

```
TYPE
ST_CTRL_DEADBAND_PARAMS:
STRUCT
    tCtrlCycleTime    : TIME := T#0ms;
    tTaskCycleTime    : TIME := T#0ms;
    fThreshold        : FLOAT;
END_STRUCT
END_TYPE
```

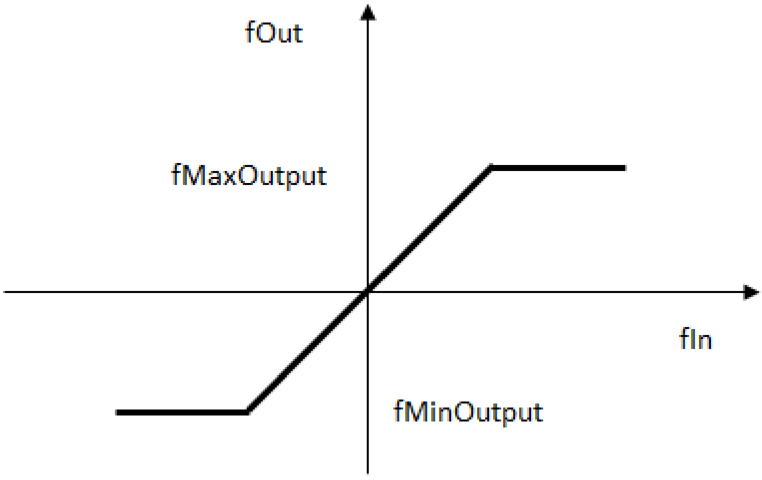
Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
fThreshold	FLOAT	The function block's dead band, see diagram.

4.2.1.7.2 FB_CTRL_LIMITER



This function block limits an input signal to a parameterizable interval.

Description of the output behavior



Inputs

```
VAR_INPUT
    fIn         : FLOAT;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input value for the function block

Outputs

```
AR_OUTPUT
    fOut        : FLOAT;
    bMinLimit   : BOOL;
    bMaxLimit   : BOOL;
    eErrorId    : E_CTRL_ERRORCODES;
    bError      : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output of the function block
bMinLimit	BOOL	A TRUE at this output indicates that the output has reached the lower limit.
bMaxLimit	BOOL	A TRUE at this output indicates that the output has reached the upper limit.
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▶ 173] when the output bError is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams        : ST_CTRL_LIMITER_PARAMS;
END_VAR
```

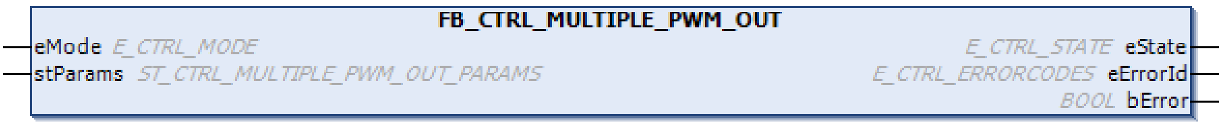
Name	Type	Description
stParams	ST_CTRL_ LIMITER_ PARAMS	Parameter structure of the function block

stParams consists of the following elements:

```
TYPE ST_CTRL_LIMITER_PARAMS:
STRUCT
    tCtrlCycleTime   : TIME := T#0ms;
    tTaskCycleTime   : TIME := T#0ms;
    fMinOutput       : FLOAT;
    fMaxOutput       : FLOAT;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
fMinOutput	FLOAT	Lower limit to which the output is restricted.
fMaxOutput	FLOAT	Upper limit to which the output is restricted.

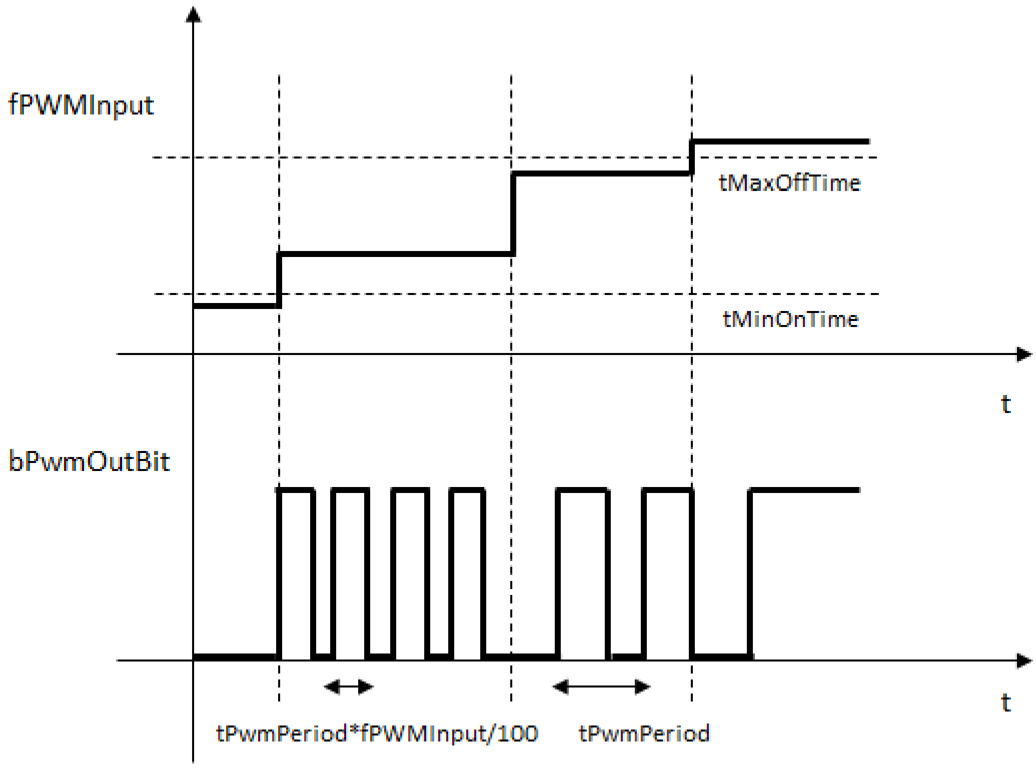
4.2.1.7.3 FB_CTRL_MULTIPLE_PWM_OUT



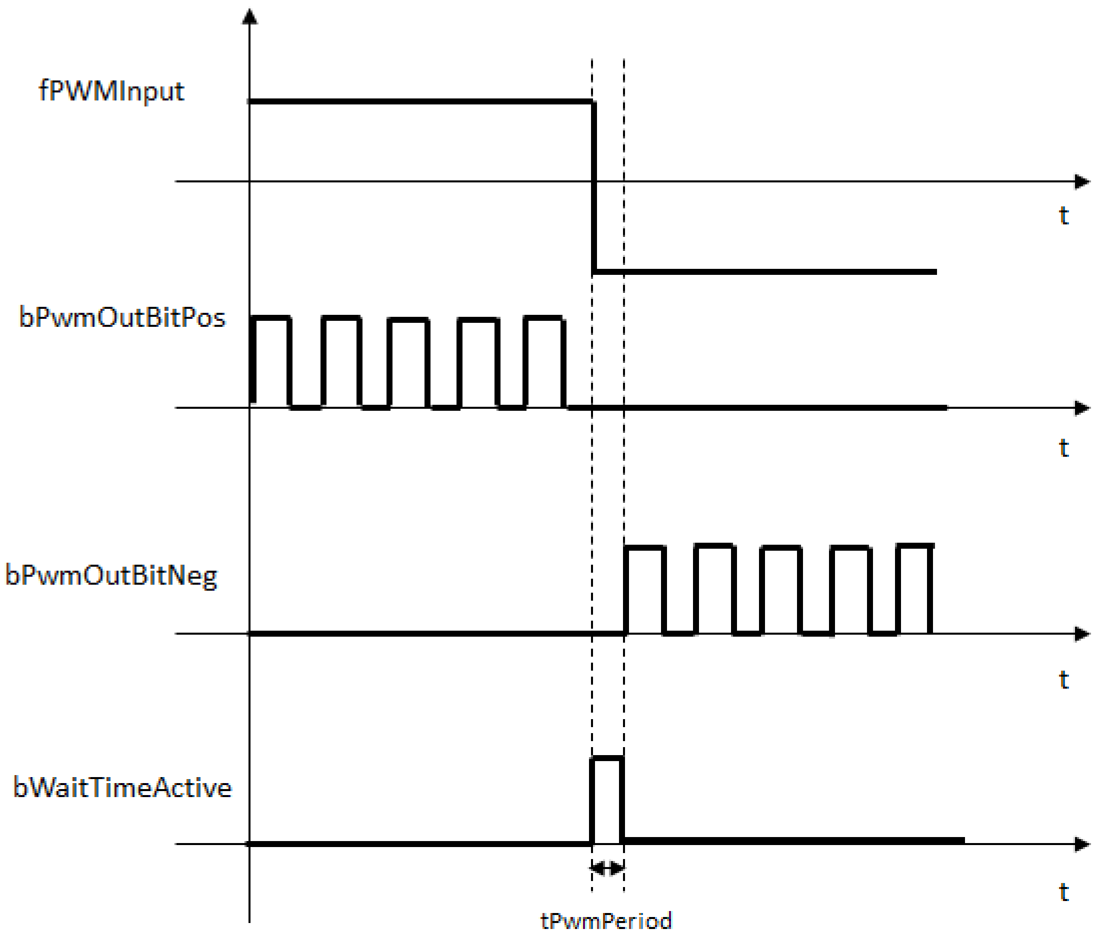
This function block creates PWM modulated output signals from a number of input signals in such a way that the temporal relationships between the output signals are arranged so that as few outputs as possible are switched on at any one time. This temporal arrangement reduces the maximum power necessary required for the actuators.

Both the minimum switch-on time and the minimum switch-off time can be parameterized, in addition to the mark-to-space ratio, in this extended function block.

Description of the output behavior 1



Description of the output behavior 2



The programmer must create the following arrays in the PLC if this function block is to be used:

```
aPwmInput            : ARRAY[1..nNumPwmOut] OF FLOAT;
aStWaitTimesConfig   : ARRAY[1..nNumPwmOut] OF ST_CTRL_MULTIPLE_PWM_OUT_TIMES;
aStPwmOutputs        : ARRAY[1.. nNumPwmOut] OF ST_CTRL_MULTIPLE_PWM_OUT_OUTPUTS;
aStPwmDataVars       : ARRAY[1.. nNumPwmOut] OF ST_CTRL_MULTIPLE_PWM_OUT_DATA;
```

The input values for the individual channels of the PWM function block are written into the **ar_fPwmInput** array. The programmer can specify the parameterizable times for the individual channels in the **ar_stWaitTimesConfig** array. The function block writes the output bits into the **ar_stPwmOutputs** array. The internal data required by the function block is stored in the **ar_stPwmDataVars** array. Under no circumstances should the structures contained in the array **ar_stPwmDataVars** be changed within the PLC program. This procedure is also illustrated in the sample program listed below.

Inputs

```
VAR_INPUT
    eMode     : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
eMode	E_CTRL_MODE	Input that specifies the operation mode of the function block.

Outputs

```
VAR_OUTPUT
    eState    : E_CTRL_STATE;
    bError    : BOOL;
    eErrorId  : E_CTRL_ERRORCODES;
END_VAR
```

Name	Type	Description
eState	E_CTRL_STATE	State of the function block
bError	BOOL	Supplies the error number [▶ 173] when the output bError is set.
eErrorId	E_CTRL_ERROR CODES	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams     : ST_CTRL_PWM_OUT_EXT_PARAMS;
END_VAR
```

Name	Type	Description
stParams	ST_CTRL_PWM_ OUT_EXT_ PARAMS	Parameter structure of the PWM element

stParams consists of the following elements:

```
TYPE ST_CTRL_MULTIPLE_PWM_OUT_PARAMS :
STRUCT
    tTaskCycleTime              : TIME;
    tPWMPeriod                  : TIME;
    nNumberOfPwmOutputs         : USINT;
    pPwmInputArray_ADR          : POINTER TO FLOAT := 0;
    nPwmInputArray_SIZEOF       : UINT;
    pPwmTimesConfig_ADR         : POINTER TO ST_CTRL_MULTIPLE_PWM_OUT_TIMES;
    nPwmTimesConfig_SIZEOF      : UINT;
    pPwmOutputArray_ADR         : POINTER TO ST_CTRL_MULTIPLE_PWM_OUT_OUTPUTS;
    nPwmOutputArray_SIZEOF      : UINT;
    pPwmData_ADR                : POINTER TO ST_CTRL_MULTIPLE_PWM_OUT_DATA;
    nPwmData_SIZEOF             : UINT;
END_STRUCT
END_TYPE
```

Name	Type	Description
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
tPWMPeriod	TIME	Period of the PWM signal
nNumberOfPwmOutputs	USINT	Number of PWM outputs. [1...n]
pPwmInputArray_ADR	POINTER TO FLOAT	Address of the PWM input array
nPwmInputArray_SIZEOF	UINT	Size of the PWM input array in bytes
pPwmTimesConfig_ADR	POINTER TO ST_CTRL_MULTIPLE_PWM_OUT_TIMES	Address of the PWM times array
nPwmTimesConfig_SIZEOF	UINT	Size of the PWM times array in bytes
pPwmOutputArray_ADR	POINTER TO ST_CTRL_MULTIPLE_PWM_OUT_OUTPUTS	Address of the PWM output array
nPwmOutputArray_SIZEOF	UINT	Size of the PWM output array in bytes
pPwmData_ADR	POINTER TO ST_CTRL_MULTIPLE_PWM_OUT_DATA	Address of the internal PWM data array
nPwmData_SIZEOF	UINT	Size of the internal PWM data array in bytes

```
TYPE ST_CTRL_MULTIPLE_PWM_OUT_TIMES :
STRUCT
    tMinOnTime     : TIME;
    tMinOffTime    : TIME;
    tMinWaitTime   : TIME;
END_STRUCT
END_TYPE
```

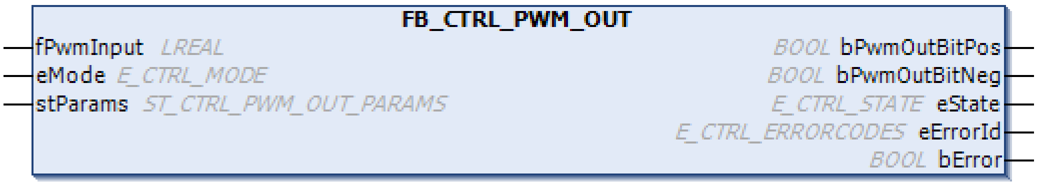
Name	Type	Description
tMinOnTime	TIME	Minimum switch-on time of the PWM output
tMinOffTime	TIME	Minimum switch-off time of the PWM output
tMinWaitTime	TIME	Waiting time between the switching actions between a positive and negative output signal

```
TYPE ST_CTRL_MULTIPLE_PWM_OUT_OUTPUTS :
STRUCT
    bPwmOutBitPos     : BOOL;
    bPwmOutBitNeg     : BOOL;
    bWaitTimeActive   : BOOL;
END_STRUCT
END_TYPE
```

Name	Type	Description
bPwmOutBitPos	BOOL	PWM signal, when fPwmInput > 0.0
bPwmOutBitNeg	BOOL	PWM signal, when fPwmInput < 0.0
bWaitTimeActive	BOOL	A TRUE at this output indicates that the waiting time between the switching actions of the output signals is active. This output can be used to hold an I component that may be present in the prior controller constant during this time.

```
TYPE ST_CTRL_MULTIPLE_PWM_OUT_DATA :
STRUCT
Internal structure. This must not be written to.
END_STRUCT
END_TYPE
```

4.2.1.7.4 FB_CTRL_PWM_OUT

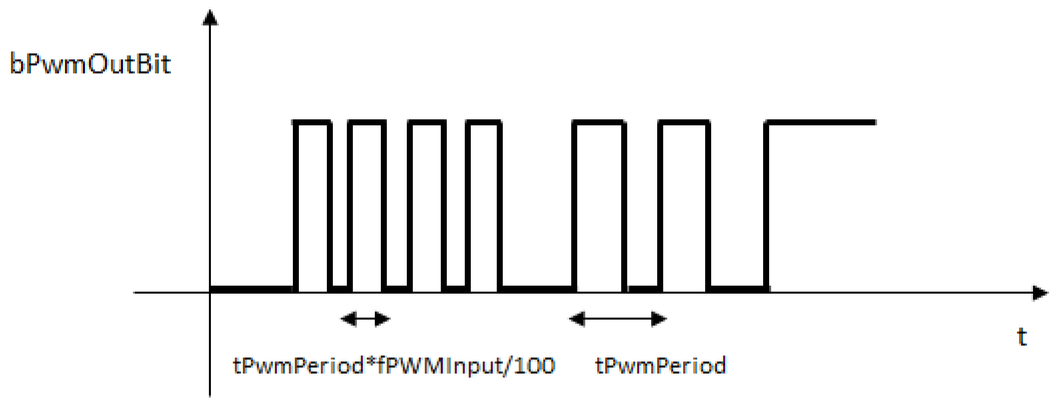


This function block creates a PWM modulated signal on the basis of the input signal.

Description of the output behavior

This function block creates a PWM signal at the outputs with a mark-to-space ratio corresponding to the **fPwmInput** input. The mark-to-space ratio is specified at the input in %; the range from -100% to 100% is available. If a positive value is specified, the pulse width modulated signal is made available at the **bPwmOutBitPos** output. If the specified mark-to-space ratio is negative, the signal is output at **bPwmOutBitNeg**. These two signals therefore make it possible to control two different actuators, depending on the arithmetic sign.

If the **bInstantPWMUpdate** is set to TRUE it is possible to adopt a new input value instantly. The new input value, in other words, is effective even in the current PWM cycle. If this parameter is FALSE, then a new input value is only adopted at the start of a new PWM cycle.



Inputs

```
VAR_INPUT
    fPwmInput   : FLOAT;
    eMode       : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
fPwmInput	FLOAT	Input value
eMode	E_CTRL_MODE	Input that specifies the operation mode of the function block.

Outputs

```
VAR_OUTPUT
    bPwmOutBitPos   : BOOL;
    bPwmOutBitNeg   : BOOL;
    eState          : E_CTRL_STATE;
    bError          : BOOL;
    eErrorId        : E_CTRL_ERRORCODES;
END_VAR
```

Name	Type	Description
bPwmOutBitPos	BOOL	PWM signal, when fPwmInput > 0.0.
bPwmOutBitNeg	BOOL	PWM signal, when fPwmInput < 0.0.
eState	E_CTRL_STATE	State of the function block
bError	BOOL	Supplies the error number [▶ 173] when the output `bError` is set.
eErrorId	E_CTRL_ERRORCODES	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams    : ST_CTRL_PWM_OUT_PARAMS;
END_VAR
```

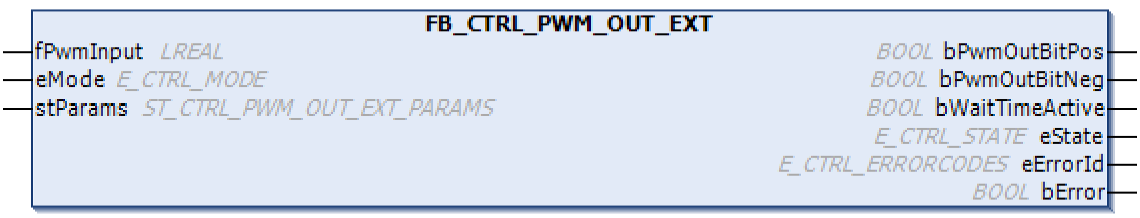
Name	Type	Description
stParams	ST_CTRL_PWM_OUT_PARAMS	Parameter structure of the PWM element

`stParams` consists of the following elements:

```
TYPE ST_CTRL_PWM_OUT_PARAMS:
STRUCT
    tTaskCycleTime      : TIME
    tPWMPeriod          : TIME;
    bInstantPWMUpdate   : BOOL;
END_STRUCT
END_TYPE
```

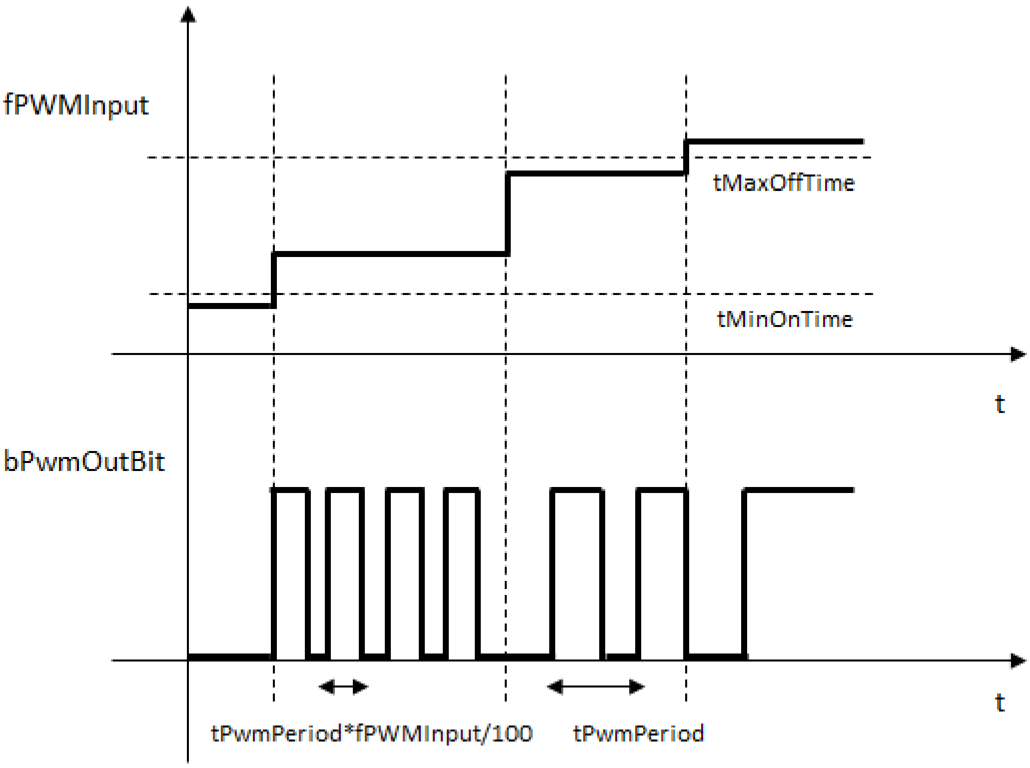
Name	Type	Description
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
tPWMPeriod	TIME	Period of the PWM signal
bInstantPWMUpdate	BOOL	If this flag is TRUE, then a new input value is immediately adopted, even in the present PWM cycle.

4.2.1.7.5 FB_CTRL_PWM_OUT_EXT

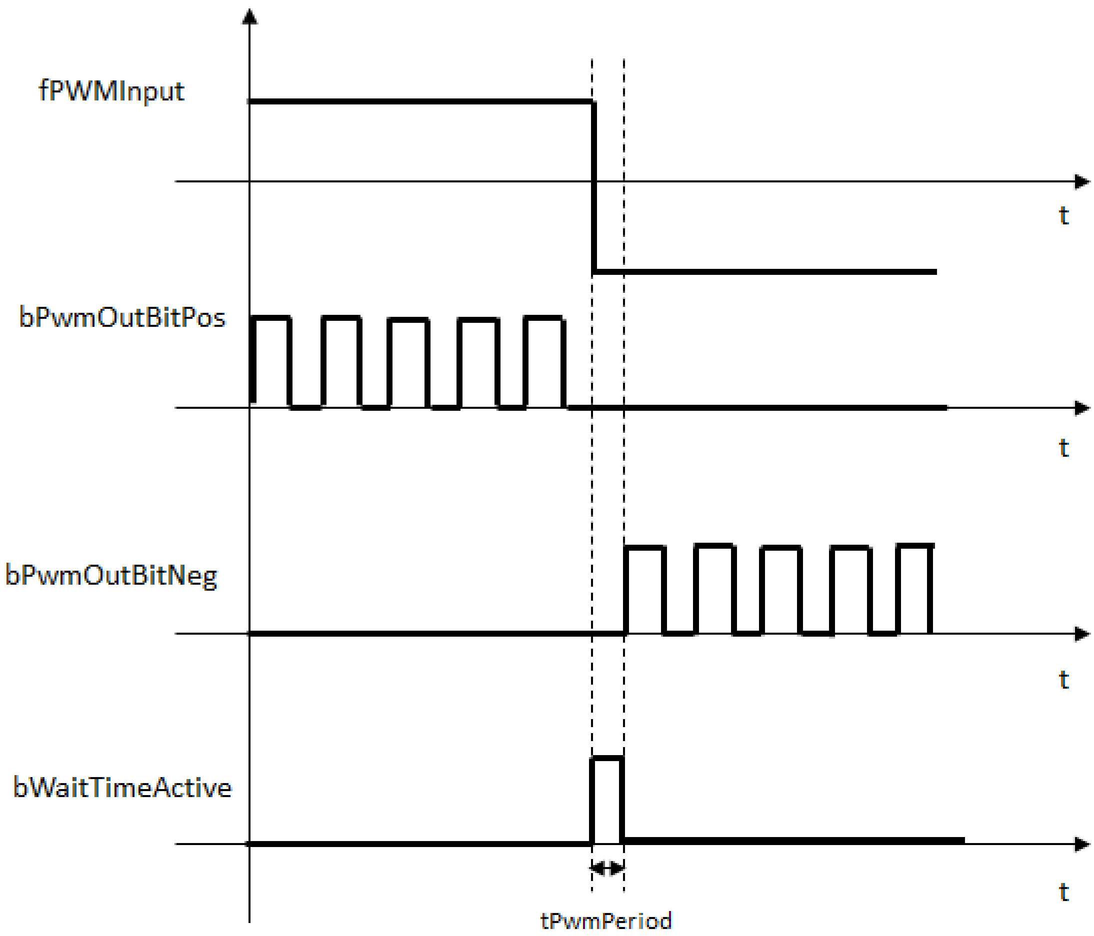


This function block creates a PWM modulated signal on the basis of the input signal. Both the minimum switch-on time and the minimum switch-off time can be parameterized, in addition to the mark-to-space ratio, in this extended function block.

Description of the output behavior 1



Description of the output behavior 2



Inputs

```
VAR_INPUT
    fPwmInput   : FLOAT;
    eMode       : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
fPwmInput	FLOAT	Input value for the function block
eMode	E_CTRL_MODE	Input that specifies the operation mode of the function block.

Outputs

```
VAR_OUTPUT
    bPwmOutBitPos   : BOOL;
    bPwmOutBitNeg   : BOOL;
    eState          : E_CTRL_STATE;
    bError          : BOOL;
    eErrorId        : E_CTRL_ERRORCODES;
END_VAR
```

Name	Type	Description
bPwmOutBitPos	BOOL	PWM signal, when fPwmInput > 0.0
bPwmOutBitNeg	BOOL	PWM signal, when fPwmInput < 0.0
eState	E_CTRL_STATE	State of the function block
bError	BOOL	Supplies the error number [▶ 173] when the output `bError` is set.
eErrorId	E_CTRL_ERROR CODES	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams     : ST_CTRL_PWM_OUT_EXT_PARAMS;
END_VAR
```

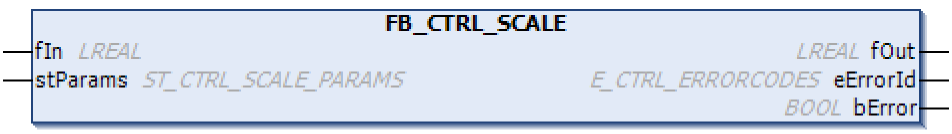
Name	Type	Description
stParams	ST_CTRL_PWM_OUT_EXT_PARAMS	Parameter structure of the PWM element

`stParams` consists of the following elements:

```
TYPE ST_CTRL_PWM_OUT_EXT_PARAMS :
STRUCT
    tTaskCycleTime   : TIME;
    tPWMPeriod       : TIME;
    tMinOnTime       : TIME;
    tMinOffTime      : TIME;
    tMinWaitTime     : TIME;
END_STRUCT
END_TYPE
```

Name	Type	Description
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
tPWMPeriod	TIME	Period of the PWM signal
tMinOnTime	TIME	Minimum switch-on time
tMinOffTime	TIME	Minimum switch-off time
tMinWaitTime	TIME	Waiting time between the switching actions between a positive and negative output signal

4.2.1.7.6 FB_CTRL_SCALE



This function block makes it possible to adjust the signal in a linear manner between two value ranges.

Inputs

```
VAR_INPUT
    fIn     : FLOAT;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input value for the function block

Outputs

```
VAR_OUTPUT
    fOut       : FLOAT;
    eErrorId   : E_CTRL_ERRORCODES;
    bError     : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Scaled output value
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▶ 173] when the output bError is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams     : ST_CTRL_SCALE_PARAMS;
END_VAR
```

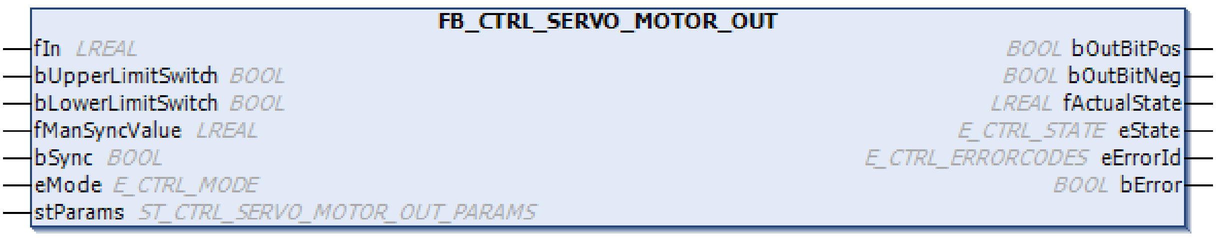
Name	Type	Description
stParams	ST_CTRL_ SCALE_PARAMS	Parameter structure of the function block

stParams consists of the following elements:

```
TYPE ST_CTRL_SCALE_PARAMS:
STRUCT
    tCtrlCycleTime   : TIME := T#0ms;
    tTaskCycleTime   : TIME := T#0ms;
    fInMin           : FLOAT;
    fInMax           : FLOAT;
    fOutMin          : FLOAT;
    fOutMax          : FLOAT;
END_STRUCT
END_TYPE
```

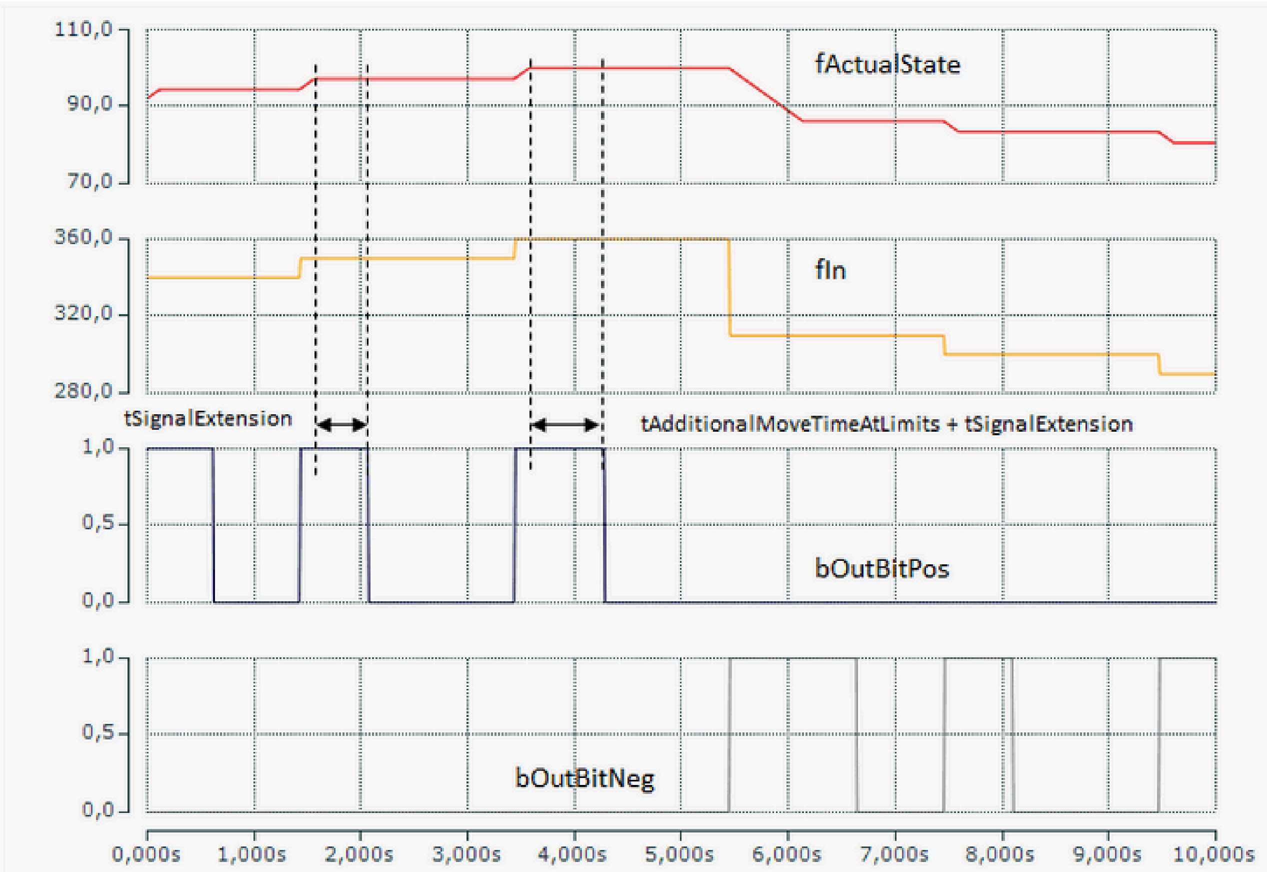
Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
fInMin	FLOAT	Minimum of the input value
fInMax	FLOAT	Maximum of the input value
fOutMin	FLOAT	Minimum of the output value
fOutMax	FLOAT	Maximum of the output value

4.2.1.7.7 FB_CTRL_SERVO_MOTOR_OUT



This function block generates pulses with which a servomotor can be driven to a defined position.

Behavior of the output



Inputs

```
VAR_INPUT
    fIn                 : FLOAT;
    bUpperLimitSwitch   : BOOL;
    bLowerLimitSwitch   : BOOL;
    fManSyncValue       : FLOAT;
    bSync               : BOOL;
    eMode               : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Control value of the controller over the range fCtrlOutMin ... fCtrlOutMax] (controller output).
bUpperLimit Switch	BOOL	Limit switch: TRUE if the upper stop has been reached.
bLowerLimit Switch	BOOL	Limit switch: TRUE if the lower stop has been reached.
fManSyncValue	FLOAT	Input with which the internal state of the current motor setting can be adjusted, or whose value is adopted in manual mode.
bSync	BOOL	With a rising edge at this input the internal motor position, which must correspond to the actual motor position, is set to the value "fManSyncValue".
eMode	E_CTRL_ MODE	Input that specifies the operation mode [▸ 173] of the function block.

For synchronization with the current valve position, this function block has the mode **eCTRL_MODE_SYNC_MOVEMENT**. In this mode, the output for closing the valve is set until the valve is closed safely. The function block is then synchronized with this valve position.

Once the synchronization process is completed, **eCTRL_STATE_ACTIVE** mode is automatically activated.

Outputs

```
VAR_OUTPUT
    bOutBitPos      : BOOL;
    bOutBitNeg      : BOOL;
    fActualState    : FLOAT;
    eState          : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError          : BOOL;
END_VAR
```

Name	Type	Description
bOutBitPos	BOOL	Output required to drive the motor in a positive direction.
bOutBitNeg	BOOL	Output required to drive the motor in a negative direction.
fActualState	FLOAT	Current motor position in the range [fCtrlOutMin ... fCtrlOutMax] in which the motor is currently located.
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▶ 173] when the output bError is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams     : ST_CTRL_SERVO_MOTOR_OUT_PARAMS;
END_VAR
```

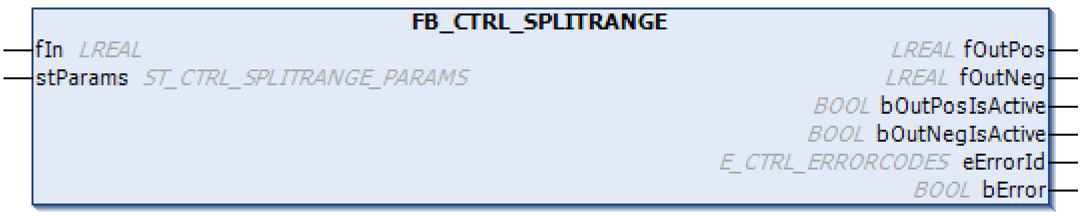
Name	Type	Description
stParams	ST_CTRL_ SERVO_MOTOR_ OUT_PARAMS	Parameter structure of the function block

stParams consists of the following elements:

```
TYPE ST_CTRL_SERVO_MOTOR_OUT_PARAMS:
STRUCT
    tCtrlCycleTime                              : TIME := T#0ms;
    tTaskCycleTime                              : TIME := T#0ms;
    tMovingTime                                 : TIME;
    tSignalExtension                            : TIME;
    tAdditionalMoveTimeAtLimits                 : TIME;
    tMinWaitTimeBetweenDirectionChange          : TIME;
    tMinimumPulseTime                           : TIME;
    bMoveOnLimitSwitch                          : BOOL;
    bStopAdditionalMoveTimeIfInputValueIsChanged : BOOL;
    fCtrlOutMax                                 : FLOAT := 100.0;
    fCtrlOutMin                                 : FLOAT := 0.0;
END_STRUCT
END_TYPE
```

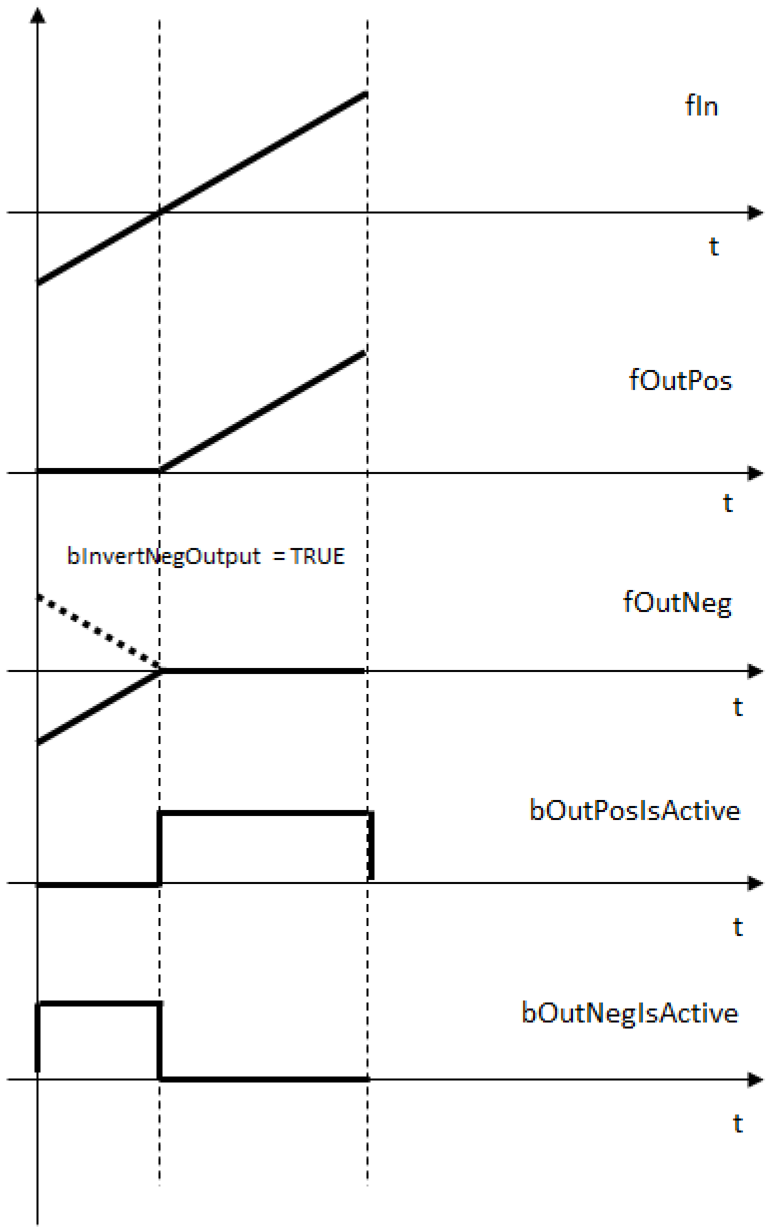
Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
tMovingTime	TIME	The time required to move the actuator from one stop to the other.
tSignalExtension	TIME	Signal extension by which each output pulse is extended in order to compensate for the dead time.
tAdditionalMoveTimeAtLimits	TIME	Supplementary signal extension, output to reliably reach the limits when the actuator is to be driven to +/-100%. Only effective if `bMoveOnLimitSwitch` is FALSE.
tMinWaitTimeBetweenDirectionChange	TIME	Minimum waiting time between positive and negative output pulses
tMinimumPulseTime	TIME	Minimum length of an output pulse
bMoveOnLimitSwitch	BOOL	If TRUE, then when the control value is either `fCtrlOutMin` or `fCtrlOutMax` a signal will continue to be output until the corresponding limit switch is reached.
bStopAdditionalMoveTimeIfInputValueIsChanged	BOOL	If this flag is TRUE, movement of the valve to the end position, which is triggered by "Additional Moving Time At Limits", is stopped, if an input value is specified that does not match the end position. If this flag is FALSE and the input value matches an end position, the valve always safely moves to the end position first, before it can move to a different valve position.
fCtrlOutMax	FLOAT	Control value for which the valve will be driven to 100%.
fCtrlOutMin	FLOAT	Control value for which the valve will be driven to 0%.

4.2.1.7.8 FB_CTRL_SPLITRANGE



This function block divides an input signal into positive and negative components. The parameters `bDisablePosOut` and `bDisableNegOut` can be used to deactivate the positive or negative output → heating mode only during the winter, cooling mode only during the summer. The `bInvertNegOutput` parameter allows the negative output to be inverted.

Description of the output behavior



Inputs

```
VAR_INPUT
    fIn     : FLOAT;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Input value for the function block

Outputs

```
VAR_OUTPUT
    fOutPos            : FLOAT;
    fOutNeg            : FLOAT;
    bOutPosIsActive    : BOOL;
    bOutNegIsActive    : BOOL;
    eErrorId           : E_CTRL_ERRORCODES;
    bError             : BOOL;
END_VAR
```

Name	Type	Description
fOutPos	FLOAT	Positive part of `fIn`
fOutNeg	FLOAT	Negative part of `fIn`
bOutPosIsActive	BOOL	TRUE indicates "fIn > 0.0"
bOutNegIsActive	BOOL	TRUE indicates "fIn < 0.0"
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▸ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams        : ST_CTRL_SPLITRANGE_PARAMS;
END_VAR
```

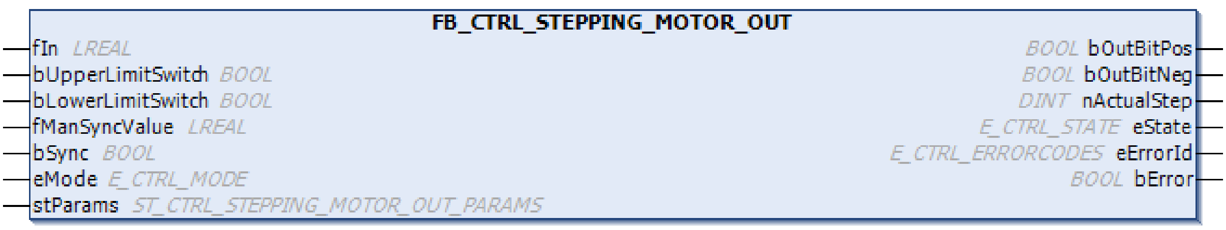
Name	Type	Description
stParams	ST_CTRL_SPLIT RANGE_ PARAMS	Parameter structure of the function block

`stParams` consists of the following elements:

```
TYPE
ST_CTRL_SPLITRANGE_PARAMS:
STRUCT
    tCtrlCycleTime    : TIME := T#0ms;
    tTaskCycleTime    : TIME := T#0ms;
    bInvertNegOutput  : BOOL;
    bDisablePosOut    : BOOL;
    bDisableNegOut    : BOOL;
END_STRUCT
END_TYPE
```

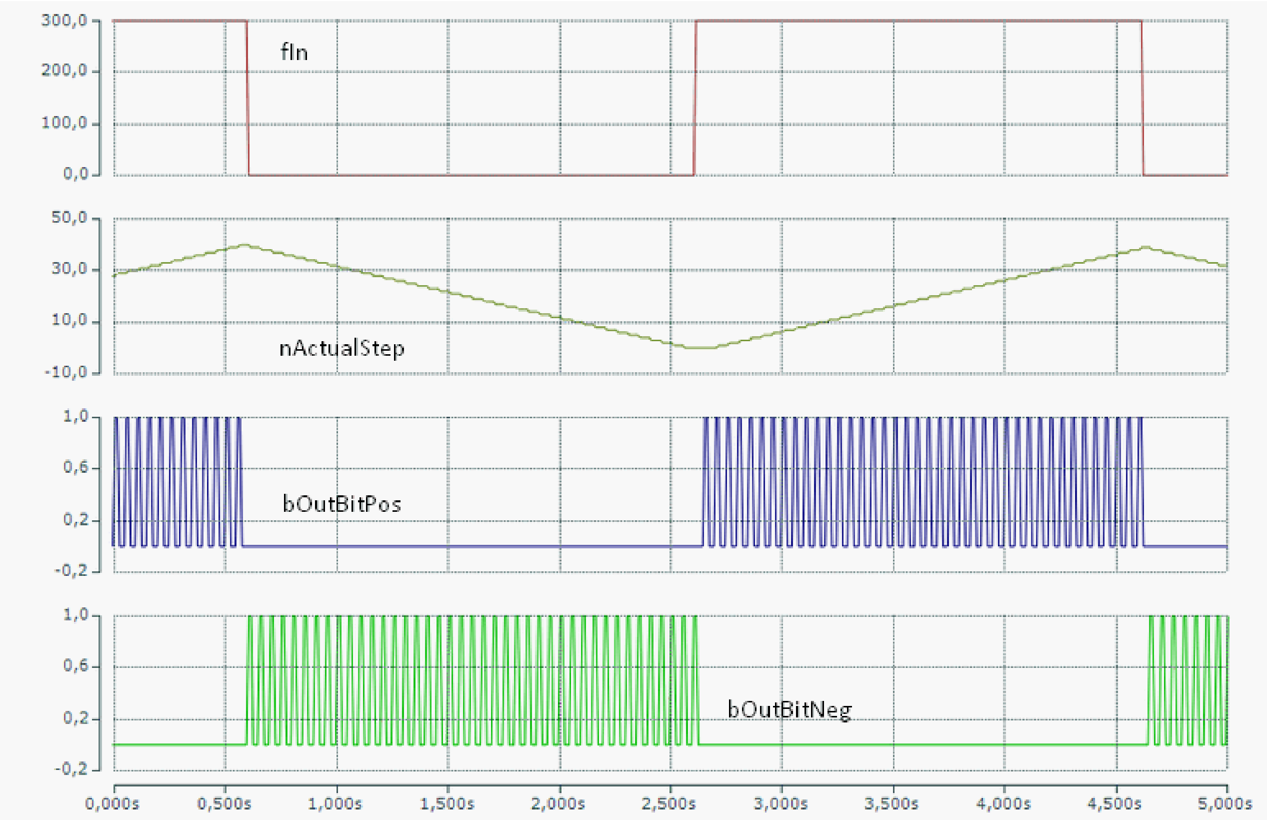
Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
bInvertNegOutput	BOOL	`fOutNeg` is inverted when this parameter is TRUE.
bDisablePosOut	BOOL	The output `fOutPos` is disabled and always "0.0“.
bDisableNegOut	BOOL	The output `fOutNeg` is disabled and always "0.0“.

4.2.1.7.9 FB_CTRL_STEPPING_MOTOR_OUT



This function block generates a control value for a stepper motor.

Behavior of the output



Inputs

```
VAR_INPUT
    fIn                   : FLOAT;
    bUpperLimitSwitch     : BOOL;
    bLowerLimitSwitch     : BOOL;
    fManSyncValue         : FLOAT;
    bSync                 : BOOL;
    eMode                 : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
fIn	FLOAT	Controller's control value (controller output)
bUpperLimit Switch	BOOL	Limit switch, becomes TRUE when the upper stop is reached.
bLowerLimit Switch	BOOL	Limit switch, becomes TRUE when the lower stop is reached.
fManSyncValue	FLOAT	Input with which the internal state of the motor setting can be adjusted, or whose value is adopted in manual mode.
bSync	BOOL	With a rising edge at this input the internal step counter is set to the step that corresponds to the value "fManSyncValue".
eMode	E_CTRL_MODE	Input that specifies the operation mode [▸ 173] of the function block.

For synchronization with the current valve position, this function block has the mode **eCTRL_MODE_SYNC_MOVEMENT**. In this mode, pulses for closing the valve are output until the valve is safely closed. The function block is then synchronized with this valve position.

Once the synchronization process is completed, **eCTRL_STATE_ACTIVE** mode is automatically activated.

Outputs

```
VAR_OUTPUT
    bOutBitPos    : BOOL;
    bOutBitNeg    : BOOL;
    nActualStep   : DINT;
    eState        : E_CTRL_STATE;
    eErrorId      : E_CTRL_ERRORCODES;
    bError        : BOOL;
END_VAR
```

Name	Type	Description
bOutBitPos	BOOL	Output required to drive the motor in a positive direction.
bOutBitNeg	BOOL	Output required to drive the motor in a negative direction.
nActualStep	DINT	Actual step at which the motor is positioned.
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▸ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams    : ST_CTRL_STEPPING_MOTOR_OUT_PARAMS;
END_VAR
```

Name	Type	Description
stParams	ST_CTRL_STEPP ING_MOTOR_ OUT_PARAMS	Parameter structure of the function block

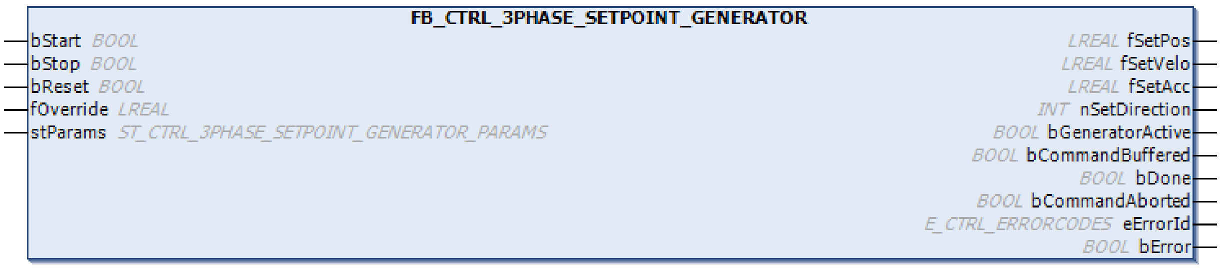
`stParams` consists of the following elements:

```
TYPE
ST_CTRL_STEPPING_MOTOR_OUT_PARAMS:
STRUCT
    tCtrlCycleTime              : TIME  := T#0ms;
    tTaskCycleTime              : TIME  := T#0ms;
    tOnTime                     : TIME;
    tOffTime                    : TIME;
    nMaxMovingPulses            : DINT;
    nMinMovingPulses            : UINT  := 0;
    bHoldWithOutputOn           : BOOL;
    nAdditionalPulsesAtLimits   : DINT;
    bMoveOnLimitSwitch          : BOOL;
    fCtrlOutMax                 : FLOAT  := 100.0;
    fCtrlOutMin                 : FLOAT  := 0.0;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
tOnTime	TIME	Pulse length
tOffTime	TIME	Pause length
nMaxMoving Pulses	DINT	Max. number of pulses required to move from one limit to another.
nMinMoving Pulses	UINT	Min. number of pulses required to move from one limit to another.
bHoldWith OutputOn	BOOL	If this parameter is set to TRUE, then an output remains set when the drive is stationary. This will result in braking.
nAdditionalPulse sAtLimits	DINT	Number of supplementary pulses that are output to reliably ensure that limits are reached.
bMoveOnLimit Switch	BOOL	If this is TRUE, then pulses are output when the control value is either 0% or 100% until the limit switch is reached.
fCtrlOutMax	FLOAT	Control value for which the valve will be driven to 100%.
fCtrlOutMin	FLOAT	Control value for which the valve will be driven to 0%.

4.2.1.8 Setpointgeneration

4.2.1.8.1 FB_CTRL_3PHASE_SETPOINT_GENERATOR



This function block represents a 3-phase setpoint generator.

Description

This function block generates a 3-phase setpoint profile in which the acceleration has a rectangular curve.

It is possible to specify a new parameter set while the generator is active. Depending on the specified type of parameter set, it is activated immediately: `eNewParameterType := eCTRL_NEW_PARAMETER_TYPE_Instant`

Alternatively, the current movement is first completed and then a new movement is started with the new parameter set:
`eNewPosType := eCTRL_NEW_PARAMETER_TYPE_NotInstant`

NOTICE

Overrunning the end position

Specifying a new parameter set may mean that the previous end position is exceeded. See the sample. It is generally recommended that end position monitoring is included after the setpoint generator.

Inputs

```
VAR_INPUT
    bStart        : BOOL;
    bStop         : BOOL;
    bReset        : BOOL;
    fOverride     : LREAL;
END_VAR
```

Name	Type	Description
bStart	BOOL	Setpoint generation begins when a positive edge appears at the `bStart` input, provided the generator is not already active and that the `bStop` and `bReset` inputs are FALSE.
bStop	BOOL	Generation of the setpoints is stopped by a positive edge at the `bStop` input. Braking is carried out using the deceleration specified in the current parameter set, and any subsequent task that may be pending is deleted.
bReset	BOOL	Reset of the setpoint generator; any positioning operation that may be active is immediately aborted, the outputs `fSetVelo` and `fSetAcc` become 0.0, the set position is set to the start position and the internal states are deleted.
fOverride	LREAL	The set velocity can be scaled through an override in the range [0 .. 100.0 %]. If the override is 100%, then the generated profile uses the set velocity as specified in the parameter set. The override implemented here does not scale the current runtime table when the override is changed. Instead, an internal restart instruction is generated having a different set velocity. The override has a resolution of 0.1%.

Outputs

```
VAR_OUTPUT
    fSetPos             : LREAL;
    fSetVelo            : LREAL;
    fSetAcc             : LREAL;
    nSetDirection       : INT;
    bCommandBuffered    : BOOL;
    bDone               : BOOL;
    bCommandAborted     : BOOL;
    eErrorId            : E_CTRL_ERRORCODES;
    bError              : BOOL;
END_VAR
```

Name	Type	Description
fSetPos	LREAL	Set position
fSetVelo	LREAL	Set velocity
fSetAcc	LREAL	Set acceleration
nSetDirection	INT	Direction of movement [-1, 0, 1], 1 --> movement direction is positive 0 --> generator is inactive 1 --> movement direction is negative

bCommandBuffered	BOOL	When this output is TRUE it indicates that a motion command is stored that will begin when the current order has completed.
		A saved order is deleted if the following special case is specified as a parameter set.
		`fAcceleration      := 0.0;`
		`fDeceleration      := 0.0;`
		`fStartPos          := 0.0;`
		`fStartVelo         := 0.0;`
		`fTargetPos         := 0.0;`
		`fTargetVelo        := 0.0;`
		`fVelocity          := 0.0;`
		`tCtrlCycleTime     := T#0s;`
		`tTaskCycleTime     := T#0s;`
		`eNewParameterType  := eCTRL_NEW_PARAMETER_TYPE_NotInstant;`

bDone	BOOL	This output becomes TRUE when the movement has been completed and the target position has been reached.
bCommandAborted	BOOL	This output becomes TRUE when the current movement is interrupted. This can be caused, for instance, through a rising edge at the `bStop` input.
eErrorId	E_CTRL_ERRORCODES	Supplies the error number [▶ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE as soon as an error situation occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams : ST_CTRL_3PHASE_SETPOINT_GENERATOR_PARAMS;
END_VAR
```

Name	Type	Description
stParams	ST_CTRL_3PHASE_SETPOINT_GENERATOR_PARAMS	Parameter structure of the setpoint generator

`stParams` consists of the following elements:

```
TYPE ST_CTRL_RAMP_GENERATOR_PARAMS :
STRUCT
    tTaskCycleTime       : TIME;
    tCtrlCycleTime       : TIME;
    fStartPos            : LREAL;
    fStartVelo           : LREAL;
    fVelocity            : LREAL;  (* >= 0.0 *)
    fTargetPos           : LREAL;
    fTargetVelo          : LREAL;
    fAcceleration        : LREAL;  (* > 0.0 *)
    fDeceleration        : LREAL;  (* > 0.0 *)
    eNewParameterType    : E_CTRL_NEW_PARAMETER_TYPE;
EN_STRUCT
END_TYPE
```

Name	Type	Description
tTaskCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tCtrlCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every task cycle this corresponds to the task cycle time of the calling task.
fStartPos	LREAL	Start position for the motion profile
fStartVelo	LREAL	Start velocity for the motion profile
fVelocity	LREAL	Velocity in units/second
fTargetPos	LREAL	Target position of the motion profile
fTargetVelo	LREAL	Target velocity of the motion profile. **Note:** The target velocity is retained when the destination position has been reached (the *bDone* flag is set), but after this point the position is no longer calculated (constant position at velocity ≠ 0.0).
fAcceleration	LREAL	Acceleration in units / second^2
fDeceleration	LREAL	Deceleration in units / second^2

eNewParameterType	E_CTRL_NEW_PARAMETER_TYPE	`eCTRL_NEW_PARAMETER_TYPE_Instant`
		and
		`eCTRL_NEW_PARAMETER_TYPE_NotInstant`

Special characteristics for eNewParameterType

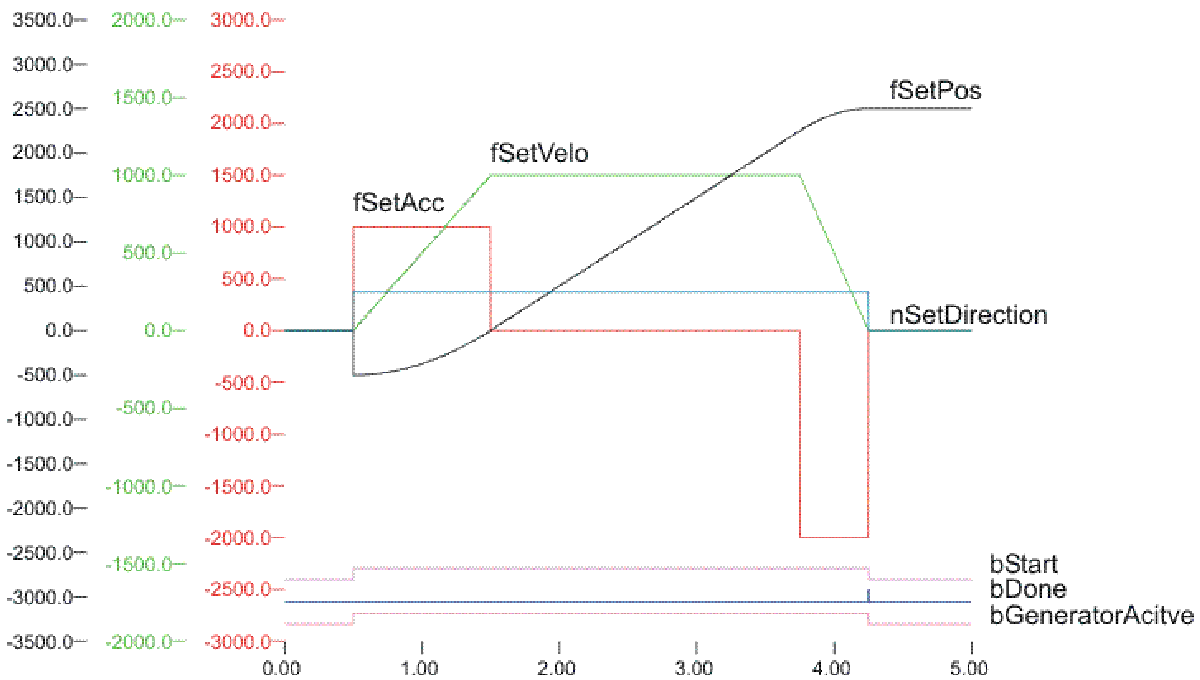
```
TYPE E_CTRL_NEW_PARAMETER_TYPE :
(    eCTRL_NEW_PARAMETER_TYPE_NotInstant := 0,
    eCTRL_NEW_PARAMETER_TYPE_Instant := 1);
END_TYPE
```

eCTRL_NEW_PARAMETER_TYPE_Instant: When a restart instruction is issued with a new parameter set, this set is adopted immediately. In other words a transition from the current state of movement to the data represented by the new parameter set is calculated, and the old parameters are discarded.

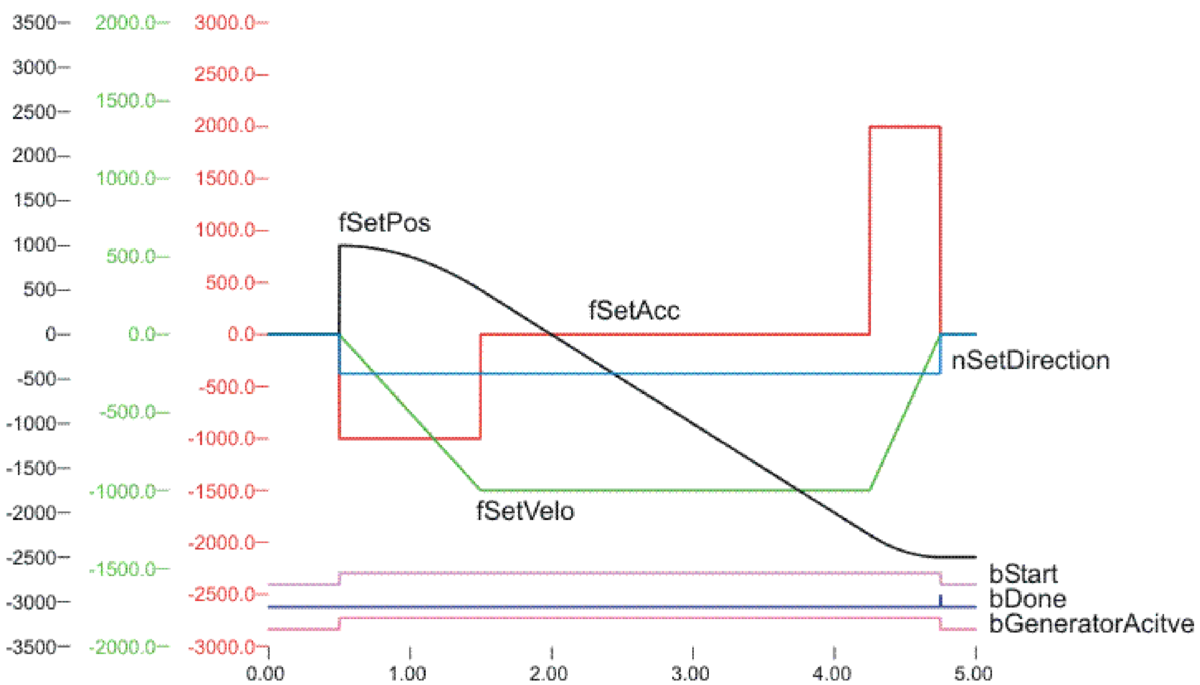
eCTRL_NEW_PARAMETER_TYPE_NotInstant: When a restart instruction is issued with a new parameter set, the new set is not adopted immediately. In other words the current movement is first executed to completion, after which positioning to the new destination is carried out with the new parameters. A TRUE at the **bCommandBuffered** output indicates that a non-instantaneous restart instruction is stored. An order that has already been stored can be overwritten or deleted by a further new, non-instantaneous parameter set.

Positioning examples

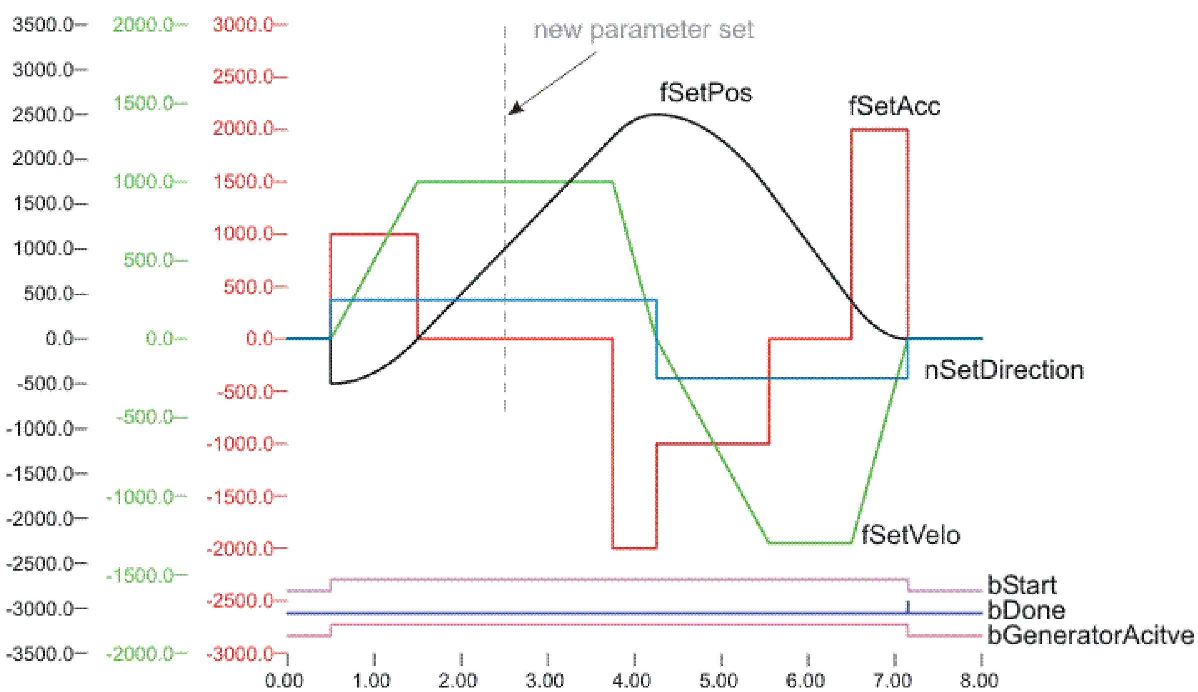
Positioning example 1:	stParams.fStartPos := -500.0; stParams.fTargetPos := 2500.0; stParams.fStartVelo := 0.0; stParams.fVelocity := 1000.0; stParams.fTargetVelo := 0.0; stParams.fAcceleration := 1000.0; stParams.fDeceleration := 2000.0; fOverride := 100.0;



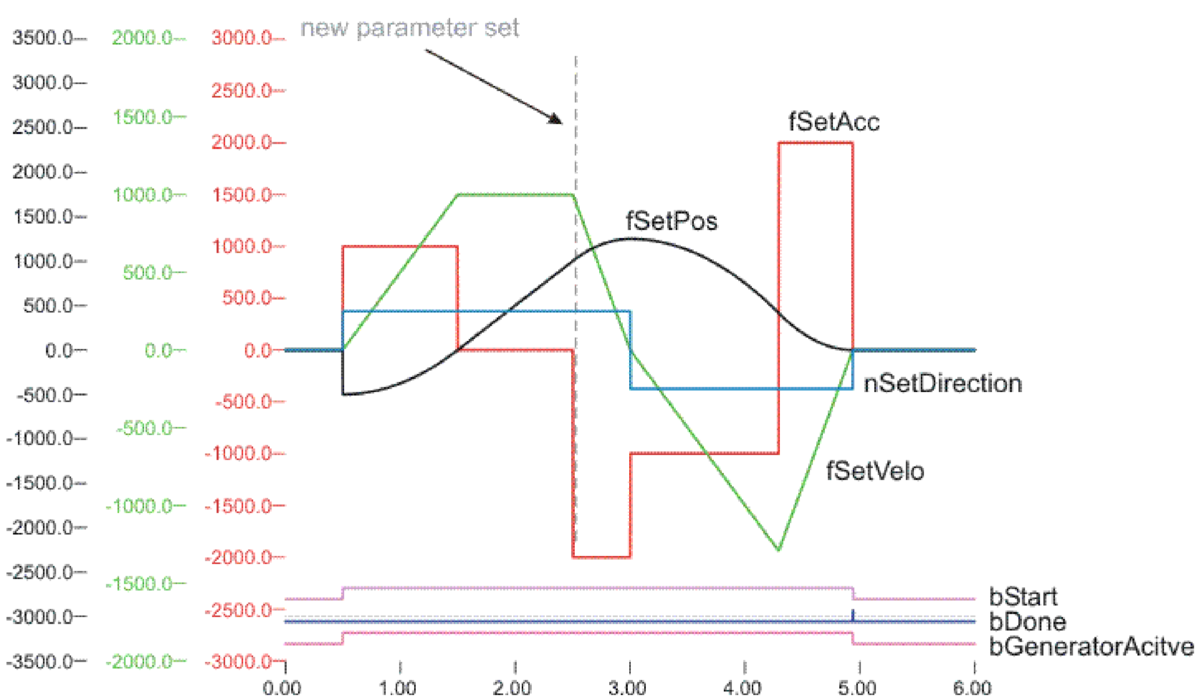
Positioning example 2:	stParams.fStartPos := 1000.0; stParams.fTargetPos := -2500.0; stParams.fStartVelo := 0.0; stParams.fVelocity := 1000.0; stParams.fTargetVelo := 0.0; stParams.fAcceleration := 1000.0; stParams.fDeceleration := 2000.0; fOverride := 100.0;

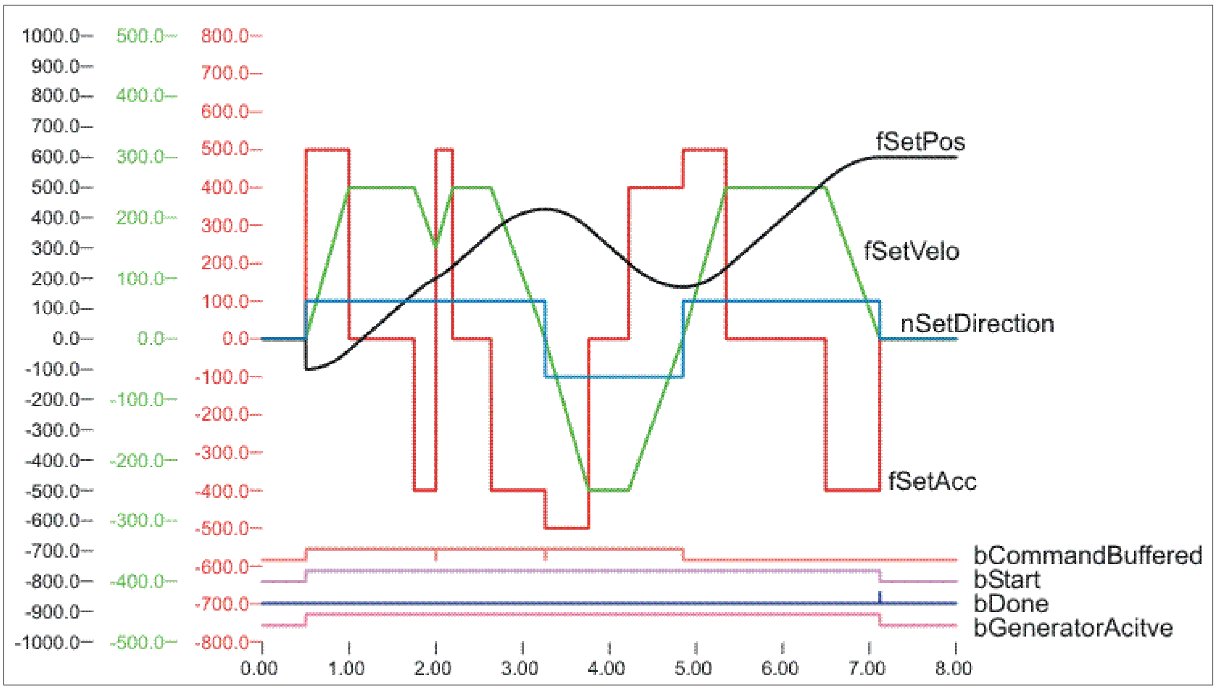


Positioning example 3:	stParams.fStartPos := -500.0; stParams.fTargetPos := 2500.0; stParams.fStartVelo := 0.0; stParams.fVelocity := 1000.0; stParams.fTargetVelo := 0.0; stParams.fAcceleration := 1000.0; stParams.fDeceleration := 2000.0; stParams.eNewParameterType := eCTRL_NEW_PARAMETER_TYPE_NotInstant; fOverride := 100.0; **parameter change if fSetPos > 1000.0, eNewPosType := eCTRL_NEW_POS_TYPE_NotInstant**stParams.fTargetPos := 0.0; stParams.fStartVelo := 0.0; stParams.fVelocity := 1300.0; stParams.fTargetVelo := 0.0; stParams.fAcceleration := 1000.0; stParams.fDeceleration := 2000.0; fOverride := 100.0;



Positioning example 4:	stParams.fStartPos := -500.0; stParams.fTargetPos := 2500.0; stParams.fStartVelo := 0.0; stParams.fVelocity := 1000.0; stParams.fTargetVelo := 0.0; stParams.fAcceleration := 1000.0; stParams.fDeceleration := 2000.0; stParams.eNewParameterType := eCTRL_NEW_PARAMETER_TYPE_NotInstant; fOverride := 100.0;**parameter change if fSetPos > 1000.0, eNewPosType := eCTRL_NEW_POS_TYPE_Instant**stParams.fTargetPos := 0.0; stParams.fStartVelo := 0.0; stParams.fVelocity := 1300.0; stParams.fTargetVelo := 0.0; stParams.fAcceleration := 1000.0; stParams.fDeceleration := 2000.0; fOverride := 100.0;

	
Positioning example 5:	**Start at point 1:** stParams.fStartPos := -100.0; stParams.fTargetPos := 200.0; stParams.fStartVelo := 0.0; stParams.fVelocity := 250.0; stParams.fTargetVelo := 150.0; stParams.fAcceleration := 500.0; stParams.fDeceleration := 400.0; stParams.eNewParameterType := eCTRL_NEW_PARAMETER_TYPE_Instant**Restart at point 2:** stParams.fTargetPos := 400.0; stParams.eNewParameterType := eCTRL_NEW_PARAMETER_TYPE_NotInstant;**Restart at point 3:** stParams.fTargetPos := 200.0; stParams.eNewParameterType := eCTRL_NEW_PARAMETER_TYPE_NotInstant;**Restart at point 4:** stParams.fTargetPos := 600.0; stParams.fTargetVelo := 0.0; stParams.eNewParameterType := eCTRL_NEW_PARAMETER_TYPE_NotInstant;



NOTICE

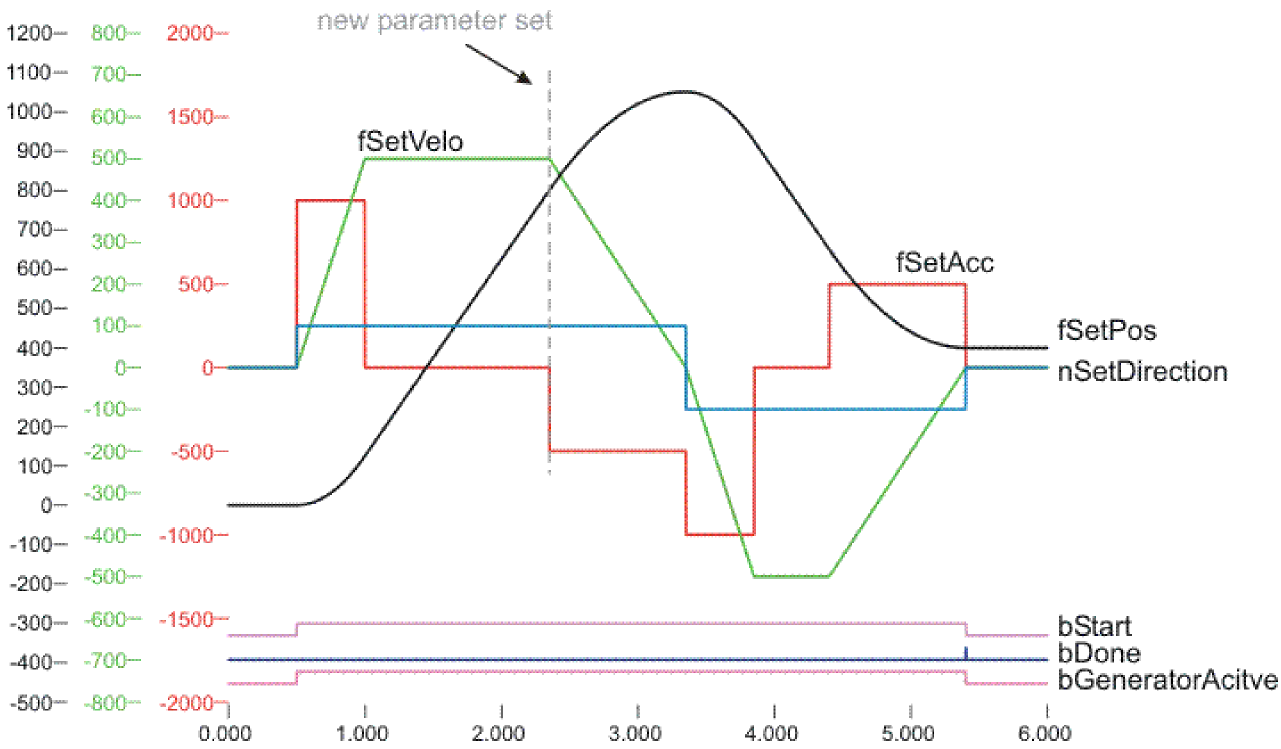
Target position overshoot!

If a new parameter set of type "eCTRL_NEW_POS_TYPE_Instant" in which the deceleration is reduced is given to the function block it is possible that the old target position will be exceeded.

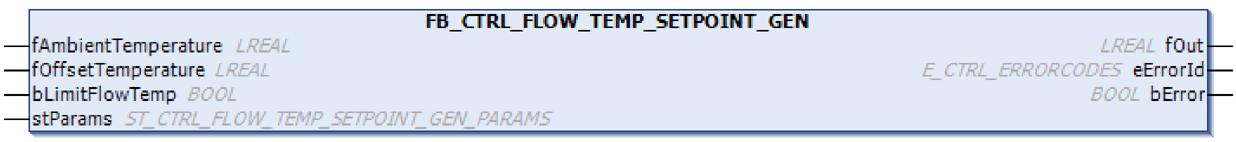
Sample:

```
stParams.fTargetPos := 1000.0;
stParams.fStartPos := 0.0;
stParams.fVelocity := 500.0;
stParams.fAcceleration := 1000.0;
stParams.fDeceleration := 1000.0;
IF fSetPos > 800.0 THEN
stParams.fTargetPos := 400.0;
stParams.fVelocity := 500.0;
stParams.fAcceleration := 1_000.0;
stParams.fDeceleration := 500.0;
stParams.eNewPosType := eCTRL_NEW_POS_TYPE_Instant;
END_IF
```

It can clearly be seen in the following scope trace that the original target position of 1000 mm is exceeded, the reason being that the new parameter set had a reduced deceleration.



4.2.1.8.2 FB_CTRL_FLOW_TEMP_SETPOINT_GEN

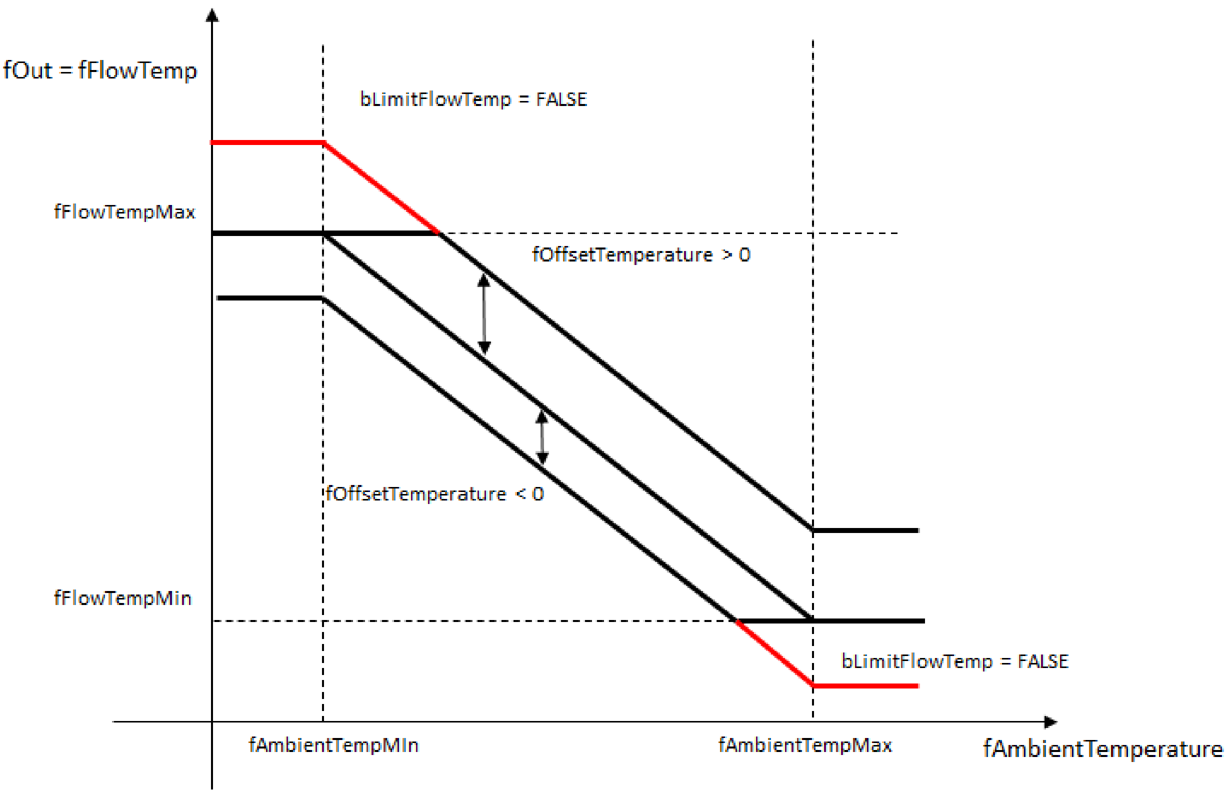


This function block enables specification of a flow temperature depending on the outdoor temperature.

Description

The setpoint for the flow temperature (`fOut`) is determined from the ambient temperature (`fAmbientTemperature`). A straight line that can be moved via an offset (`fOffsetTemperature`) is used for this purpose. The slope is determined based on the specified ambient and flow temperature corners. A flag (`bLimitFlowTemp`) is used to specify whether or not the flow temperature is restricted to its limit values. The offset temperature can be used to realize a night setback or pre-control.

Behavior of the output value



Inputs

```
VAR_INPUT
    fAmbientTemperature   : FLOAT;
    fOffsetTemperature    : FLOAT;
    bLimitFlowTemp        : BOOL;
END_VAR
```

Name	Type	Description
fAmbient Temperature	FLOAT	Start of ramp generation
fOffset Temperature	FLOAT	Start value of the ramp
bLimitFlowTemp	BOOL	Target value of the ramp

Outputs

```
VAR_OUTPUT
    fOut      : FLOAT;
    eErrorId  : E_CTRL_ERRORCODES;
    bError    : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Setpoint of the flow temperature
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▸ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE as soon as an error situation occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams     : ST_CTRL_FLOW_TEMP_SETPOINT_GEN_PARAMS;
END_VAR
```

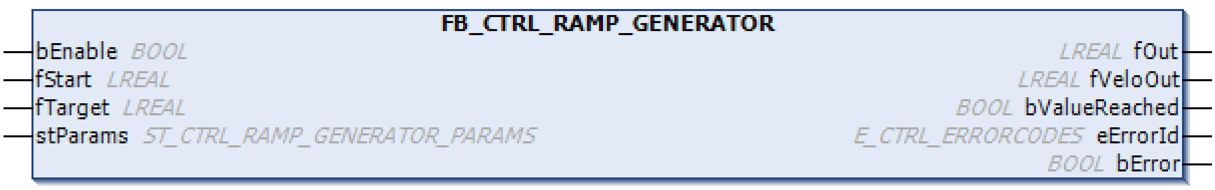
Name	Type	Description
stParams	ST_CTRL_FLOW_TEMP_SETPOINT_GEN_PARAMS	Parameter structure of the ramp generator

`stParams` consists of the following elements:

```
TYPE ST_CTRL_FLOW_TEMP_SETPOINT_GEN_PARAMS:
STRUCT
    tTaskCycleTime     : TIME;
    tCtrlCycleTime     : TIME;
    fForeRunTempMax    : FLOAT;
    fForeRunTempMin    : FLOAT;
    fAmbientTempMax    : FLOAT;
    fAmbientTempMin    : FLOAT;
END_STRUCT
END_TYPE
```

Name	Type	Description
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every task cycle this corresponds to the task cycle time of the calling task.
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
fForeRunTempMax	FLOAT	Maximum flow temperature (see diagram)
fForeRunTempMin	FLOAT	Minimum flow temperature (see diagram)
fAmbientTempMax	FLOAT	outdoor temperature for which the minimum flow temperature is specified.
fAmbientTempMin	FLOAT	outdoor temperature for which the maximum flow temperature is specified.

4.2.1.8.3 FB_CTRL_RAMP_GENERATOR

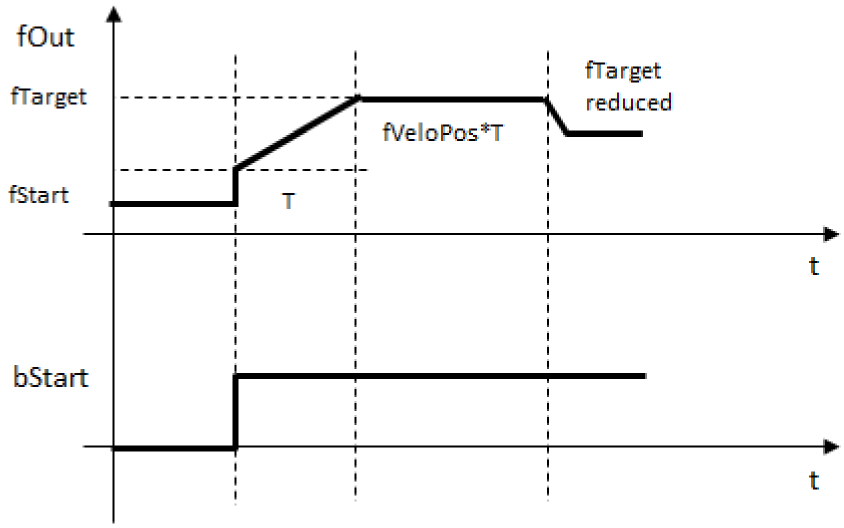


The function block provides a parameterizable ramp generator.

Description

This function block generates a ramp connecting the starting value `fStart` and the target value `fTarget`. The slope of the ramp (i.e. the velocity) is given in units/s by means of the `fVeloPos` and `fVeloNeg` parameters. The starting value is adopted when a rising edge appears at `bEnable`; calculation of the ramp then begins. As long as the signal `bEnable` remains TRUE the target value can be changed, and the output value changes, taking the form of a ramp as it moves from the current value to the presently active target value.

Behavior of the output value



Inputs

```
VAR_INPUT
    bEnable  : BOOL;
    fStart   : FLOAT;
    fTarget  : FLOAT;
END_VAR
```

Name	Type	Description
bEnable	BOOL	Start of ramp generation
fStart	FLOAT	Start value of the ramp
fTarget	FLOAT	Target value of the ramp

Outputs

```
VAR_OUTPUT
    fOut            : FLOAT;
    fVeloOut        : FLOAT;
    bValueReached   : BOOL;
    eState          : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError          : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output of the ramp generator
fVeloOut	FLOAT	Current velocity of the ramp generator
bValueReached	BOOL	This output indicates by going TRUE that the output `fOut` has reached the value `fTarget`.
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERRORCODES	Supplies the error number [▶ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE as soon as an error situation occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams     : ST_CTRL_RAMP_GENERATOR_PARAMS
END_VAR
```

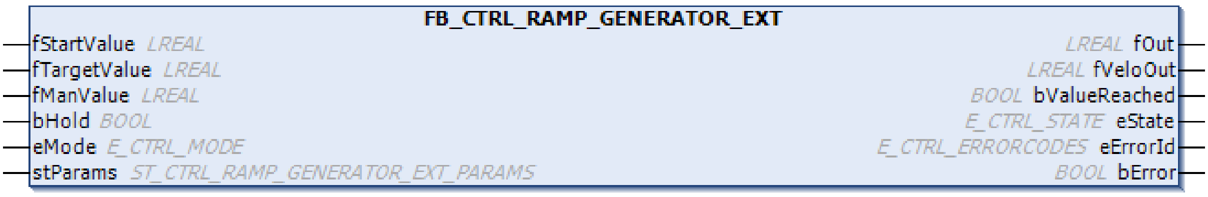
Name	Type	Description
stParams	ST_CTRL_RAMP_GENERATOR_PARAMS	Parameter structure of the ramp generator

`stParams` consists of the following elements:

```
TYPE ST_CTRL_RAMP_GENERATOR_PARAMS :
STRUCT
    tTaskCycleTime   : TIME;
    tCtrlCycleTime   : TIME;
    fVeloPos         : FLOAT;
    fVeloNeg         : FLOAT;
END_STRUCT
END_TYPE
```

Name	Type	Description
tTaskCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tCtrlCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every task cycle this corresponds to the task cycle time of the calling task.
fVeloPos	FLOAT	Velocity in unit/s, with which the output is changed from a lower value to a higher one.
fVeloNeg	FLOAT	Velocity in unit/s, with which the output is changed from a higher value to a lower one.

4.2.1.8.4 FB_CTRL_RAMP_GENERATOR_EXT

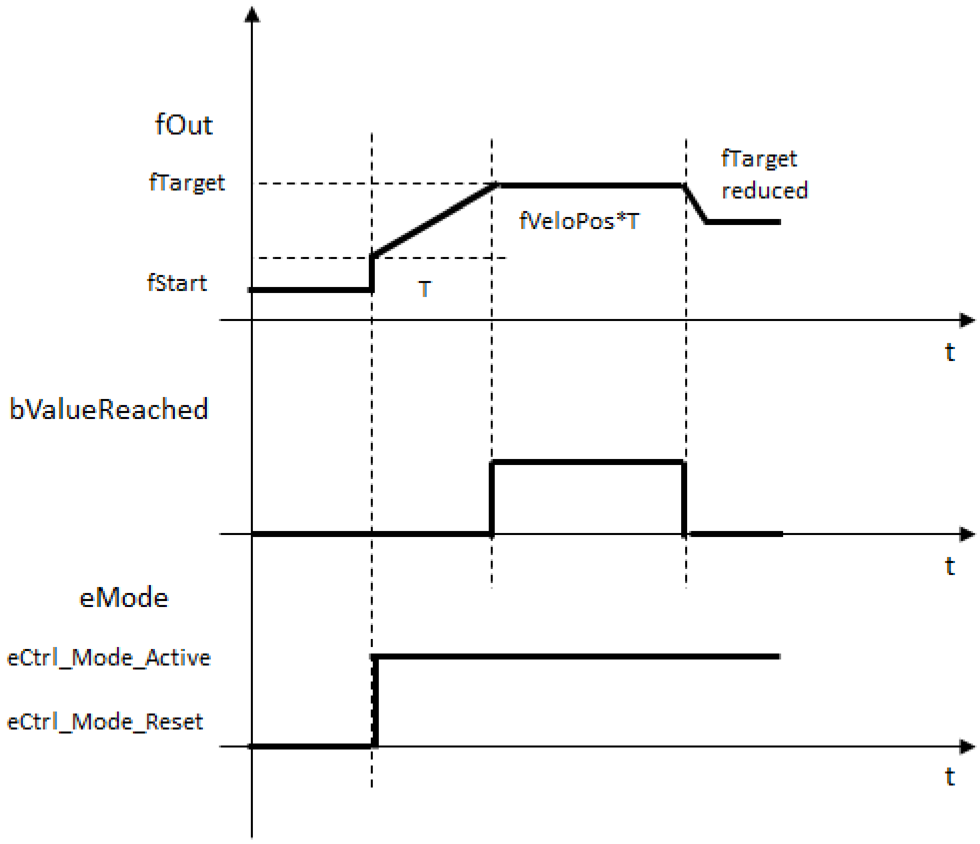


This function block represents a parameterizable ramp generator. In contrast to **FB_CTRL_RAMP_GENERATOR** it supports **E_CTRL_MODE**.

Description:

This function block generates a ramp connecting the starting value `fStartValue` and the target value `fTargetValue`. The slope of the ramp (i.e. the velocity) is given in units/s by means of the `fVeloPos` and `fVeloNeg` parameters. The start value is adopted when **eCTRL_MODE_RESET** changes to **eCTRL_MODE_ACTIVE**, and calculation of the ramp begins. As long as the function block remains in **eCTRL_MODE_ACTIVE** the target value can be changed, and the output value changes, taking the form of a ramp as it moves from the current value to the presently active target value. The current velocity is output at `fVeloOut`. It is possible to use this for feed forward in the control loop.

Behavior of the output value



Inputs

```
VAR_INPUT
    fStartValue    : FLOAT;
    fTargetValue   : FLOAT;
    fManValue      : FLOAT;
    bHold          : BOOL;
    eMode          : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
fStartValue	FLOAT	Start value of the ramp
fTargetValue	FLOAT	Target value of the ramp
fManValue	FLOAT	Input value to which the output in `eCTRL_MODE_MANUAL` is set.
bHold	BOOL	Calculation of the ramp is halted at the current value.
eMode	E_CTRL_MODE	Input that specifies the operation mode [▸ 173] of the function block.

Outputs

```
VAR_OUTPUT
    fOut            : FLOAT;
    fVeloOut        : FLOAT;
    bValueReached   : BOOL;
    eState          : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError          : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output of the ramp generator
fVeloOut	FLOAT	Current velocity of the ramp generator
bValueReached	BOOL	This output indicates by going TRUE that the output `fOut` has reached the value "fTargetValue".
eState	E_CTRL_ STATE	State of the function block
eErrorId	E_CTRL_ ERRORCODES	Supplies the error number [▸ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE as soon as an error situation occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams     : ST_CTRL_RAMP_GENERATOR_EXT_PARAMS;
END_VAR
```

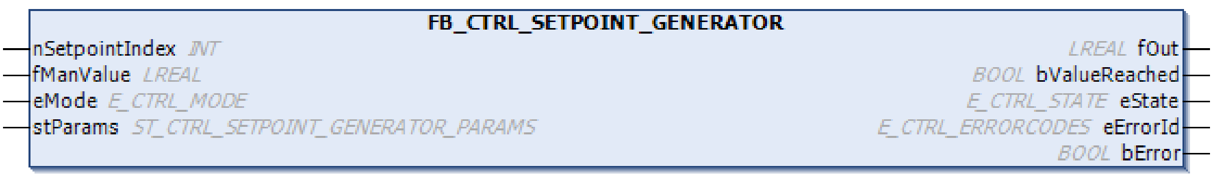
Name	Type	Description
stParams	ST_CTRL_RAMP _GENERATOR_E XT_ PARAMS	Parameter structure of the ramp generator

`stParams` consists of the following elements:

```
TYPE ST_CTRL_RAMP_GENERATOR_EXT_PARAMS :
STRUCT
    tTaskCycleTime   : TIME;
    tCtrlCycleTime   : TIME;
    fVeloPos         : FLOAT;
    fVeloNeg         : FLOAT;
END_STRUCT
END_TYPE
```

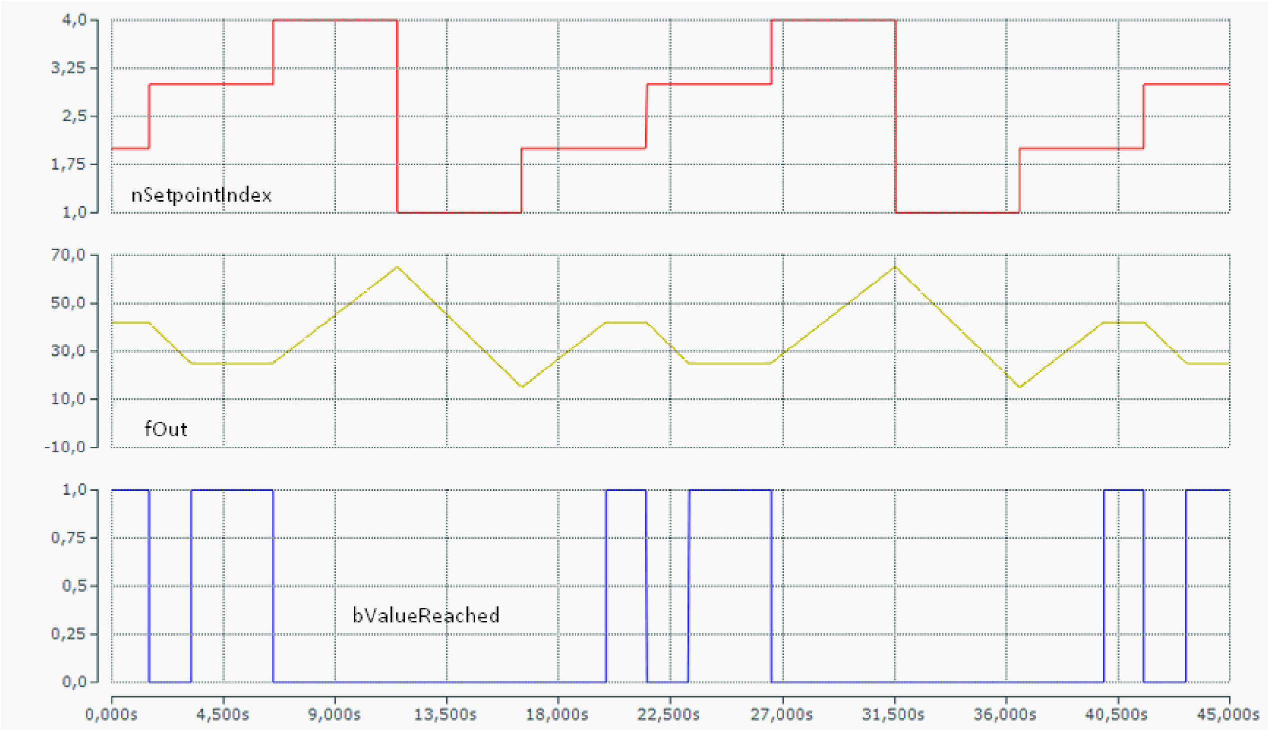
Name	Type	Description
tTaskCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tCtrlCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every task cycle this corresponds to the task cycle time of the calling task.
fVeloPos	FLOAT	Velocity (>0.0) in unit/s, with which the output is changed from a lower value to a higher one.
fVeloNeg	FLOAT	Velocity (>0.0) in unit/s, with which the output is changed from a higher value to a lower one.

4.2.1.8.5 FB_CTRL_SETPOINT_GENERATOR



The function block provides a setpoint generator that outputs the setpoint selected from a table. Changing from one setpoint to another can be implemented continuously or discontinuously.

Behavior of the output value



Example table:

Index	
1	12
2	42
3	25
…	73

Description

The individual setpoints are stored in the array. The array must be made known to the function block through the appropriate parameters. One of the setpoints stored in the table is selected by means of the `nSetpointIndex` input. This is then made available at the output, and can be used as the setpoint for the controller. Changing from one value to another can be implemented in a linear fashion or as a jump. The velocity of a continuous transition is specified by the fVeloPos and fVeloNeg parameters. The `bValueReached` output indicates that the chosen setpoint has been reached.

Inputs

```
VAR_INPUT
    nSetpointIndex   : INT;
    fManValue        : FLOAT;
    eMode            : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
nSetpointIndex	INT	Index of the selected setpoint
fManValue	FLOAT	Input whose value is output in manual mode.
eMode	E_CTRL_MODE	Input that specifies the operation mode [▶ 173] of the function block.

Outputs

```
VAR_OUTPUT
    fOut            : SETPOINT_TABLE_ELEMENT;
    bValueReached   : BOOL;
    eState          : E_CTRL_STATE;
    eErrorId        : E_CTRL_ERRORCODES;
    bError          : BOOL;
END_VAR
```

Name	Type	Description
fOut	SETPOINT_ TABLE_ELEMEN T	Output of the setpoint generator
bValueReached	BOOL	The output is TRUE when the selected setpoint has been reached, i.e. the ramp leading to the selected value has finished.
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▶ 173] when the output `bError` is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams    : ST_CTRL_SETPOINT_GENERATOR_PARAMS;
END_VAR
```

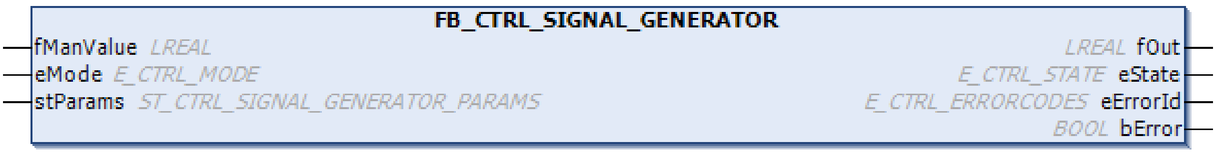
Name	Type	Description
stParams	ST_CTRL_SETP OINT_ GENERATOR_ PARAMS	Parameter structure of the ramp generator

`stParams` consists of the following elements:

```
TYPE ST_CTRL_SETPOINT_GENERATOR_PARAMS:
STRUCT
    tCtrlCycleTime              : TIME := T#0ms;
    tTaskCycleTime              : TIME := T#0ms;
    pDataTable_ADR              : POINTER TO
     INTERPOLATION_TABLE_ELEMENT := 0;
    nDataTable_SIZEOF           : UINT := 0;
    nDataTable_NumberOfRows     : UINT := 0;
    fVeloPos                    : FLOAT;
    fVeloNeg                    : FLOAT;
    bDisableRamping             : BOOL := FALSE;
END_STRUCT
END_TYPE
```

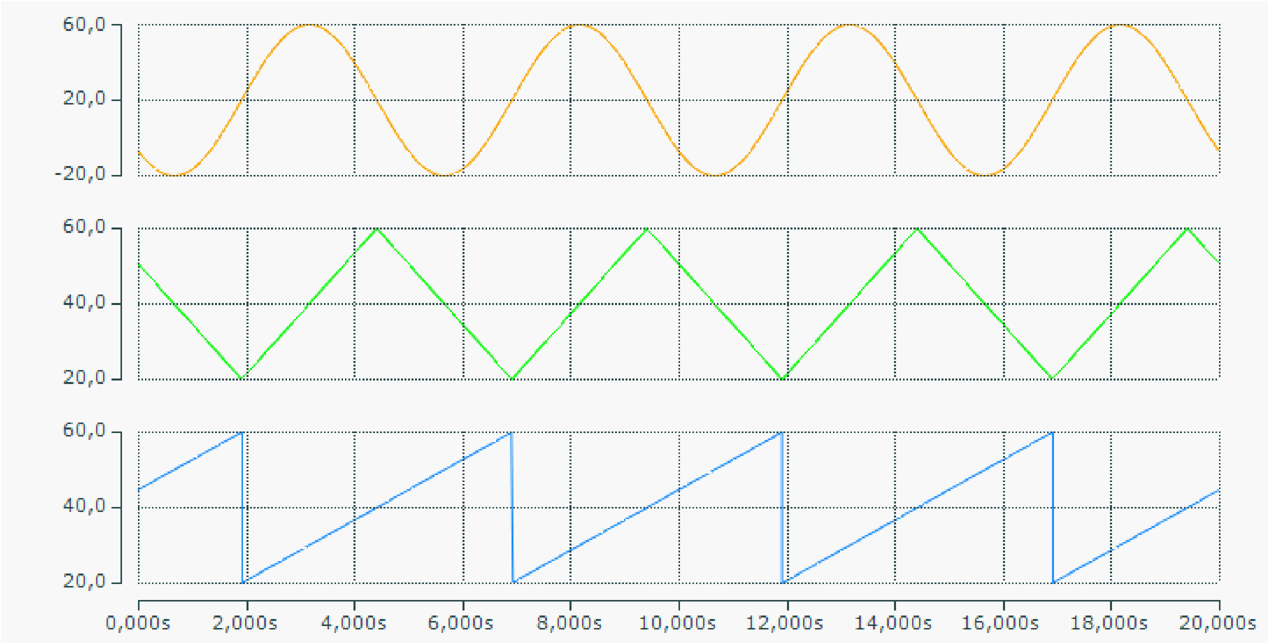
Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle. Indent: -103; margin-left: 108">
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.
pDataTable_ADR	POINTER TO INTERPOLATION_TABLE_ELEMENT	Address of the data array
nDataTable_SIZEOF	UINT	Size of the data array
nDataTable_NumberOfRows	UINT	Number of rows in the data array
fVeloPos	FLOAT	Velocity in unit/s, with which the output is changed from a lower value to a higher one.
fVeloNeg	FLOAT	Velocity in unit/s, with which the output is changed from a higher value to a lower one.
bDisableRamping	BOOL	A continuous output value is not calculated if this parameter is TRUE. There is a step change between the output values.

4.2.1.8.6 FB_CTRL_SIGNAL_GENERATOR



The function block provides a signal generator offering **triangular, sine** and **sawtooth** signal forms.

Output signals



Inputs

```
VAR_INPUT
    fManValue   : FLOAT;
    eMode       : E_CTRL_MODE;
END_VAR
```

Name	Type	Description
fManValue	FLOAT	Input whose value is present at the output in manual mode.
eMode	E_CTRL_MODE	Input that specifies the operation mode [▶ 173] of the function block.

Outputs

```
VAR_OUTPUT
    fOut       : FLOAT;
    eState     : E_CTRL_STATE;
    eErrorId   : E_CTRL_ERRORCODES;
    bError     : BOOL;
END_VAR
```

Name	Type	Description
fOut	FLOAT	Output of the signal generator
eState	E_CTRL_STATE	State of the function block
eErrorId	E_CTRL_ERROR CODES	Supplies the error number [▶ 173] when the output bError is set.
bError	BOOL	Becomes TRUE, as soon as an error occurs.

Inputs/ outputs

```
VAR_IN_OUT
    stParams    : ST_CTRL_SIGNAL_GENERATOR_PARAMS;
END_VAR
```

Name	Type	Description
stParams	ST_CTRL_SIGNAL_GENERATOR_PARAMS	Parameter structure of the function block

`stParams` consists of the following elements:

```
TYPE ST_CTRL_SIGNAL_GENERATOR_PARAMS:
STRUCT
    tCtrlCycleTime   : TIME := T#0ms;
    tTaskCycleTime   : TIME := T#0ms;
    eSignalType      : E_CTRL_SIGNAL_TYPE;
    tCylceDuration   : TIME;
    fAmplitude       : FLOAT;
    fOffset          : FLOAT := 0.0;
    tStart           : TIME := T#0s;
END_STRUCT
END_TYPE
```

Name	Type	Description
tCtrlCycleTime	TIME	Cycle time with which the control loop is processed. This must be greater than or equal to the **TaskCycleTime**. The function block uses this input value to calculate internally whether the state and the output values have to be updated in the current cycle.
tTaskCycleTime	TIME	Cycle time with which the function block is called. If the function block is called in every cycle this corresponds to the task cycle time of the calling task.

eSignalType	E_CTRL_ SIGNAL_TYPE	Selection of the type of signal.
		`TYPE E_CTRL_SIGNAL_TYPE :`
		`( eCTRL_TRIANGLE := 0,`
		`eCTRL_SINUS := 1,`
		`eCTRL_SAWTOOTH := 2 );`
		`END_TYPE`

tCylceDuration	TIME	Period of the generated signal curve
fAmplitude	FLOAT	Amplitude of the generated signal curve
fOffset	FLOAT	Offset to be added to the signal curve
tStart	TIME	The moment of time within a period at which following the signal curve will begin when switching into **eCTRL_MODE_ACTIVE** occurs.

4.2.2 Global Constants

4.2.2.1 Library version

All libraries have a specific version. This version is shown in the PLC library repository too.
A global constant contains the library version information:

Global_Version

```
VAR_GLOBAL CONSTANT
    stLibVersion_Tc2_ControllerToolbox : ST_LibVersion;
END_VAR
```

ST_LibVersion

To compare the existing version to a required version the function F_CmpLibVersion (defined in Tc2_System library) is offered.

Obsolete Functions

All other possibilities known from TwinCAT2 libraries to query a library version are obsolete!

4.2.3 Data Structures

4.2.3.1 Definition of the structures and enums

This appendix describes all the structures and enums used in the TwinCAT controller toolbox.

FLOAT:

The function blocks of the library only work with the data type FLOAT. This data type is defined as LREAL or as REAL in a supplementary library.

On a **PC system** the additional library "**TcFloatPC.lib**" is automatically integrated.

```
TYPE
FLOAT : LREAL;
END_TYPE
```

E_CTRL_MODE

```
TYPE E_CTRL_MODE :
(
    eCTRL_MODE_IDLE      := 0,
    eCTRL_MODE_PASSIVE      := 1,
    eCTRL_MODE_ACTIVE      := 2,
    eCTRL_MODE_RESET      := 3,
    eCTRL_MODE_MANUAL      := 4,
    eCTRL_MODE_TUNE      := 5,
    eCTRL_MODE_SELFTEST     := 6,
    eCTRL_MODE_SYNC_MOVEMENT := 7
)
END_TYPE
```

E_CTRL_STATE

```
TYPE E_CTRL_STATE :
(
    eCTRL_STATE_IDLE        := 0,
    eCTRL_STATE_PASSIVE := 1,
    eCTRL_STATE_ACTIVE := 2,
    eCTRL_STATE_RESET       := 3,
    eCTRL_STATE_MANUAL := 4,
    eCTRL_STATE_TUNING := 5,
    eCTRL_STATE_TUNED       := 6,
    eCTRL_STATE_SELFTEST := 7,
    eCTRL_STATE_ERROR       := 8,
    eCTRL_STATE_SYNC_MOVEMENT := 9
);
END_TYPE
```

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
```

E_CTRL_SIGNAL_TYPE

```
TYPE E_CTRL_SIGNAL_TYPE :
(
eCTRL_TRIANGLE := 0,
eCTRL_SINUS := 1,
eCTRL_SAWTOOTH := 2
);
END_TYPE
```

```
TYPE E_CTRL_STEP_SENSORTYPE
(
eSENSOR_NONE := 0,
eSENSOR_PT100 := 1,
eSENSOR_THERMO_J := 2,
eSENSOR_THERMO_K := 3
);
END_TYPE
```

5 Example project

The behavoiur of the TF4100 Controller Toolbox function blocks are shown with an example. The base functionalities are included. All closed loop controlled systems of the programs are simulated, no hardware is necessary for the project.

5.1 Example Installation

The TwinCAT controller toolbox offers the programmer transfer elements with which a very wide variety of controllers can be implemented.

1. Save the sample program https://infosys.beckhoff.com/content/1033/TF4100_TC3_Controller_Toolbox/Resources/1461955979.zip and unpack it.
   ⇨ The sample program *TcControllerToolbox_Examples.*sln is loaded into the TwinCAT XAE, compiled and started.
2. Load the solution file.
3. Compile the PLC project via the menu option *Project - Rebuild All*.
4. Load the PLC project into the runtime system via the menu option *Online - Login*.
5. Start the program via the menu option *Online - Run*.

Using TwinCAT ScopeView examples

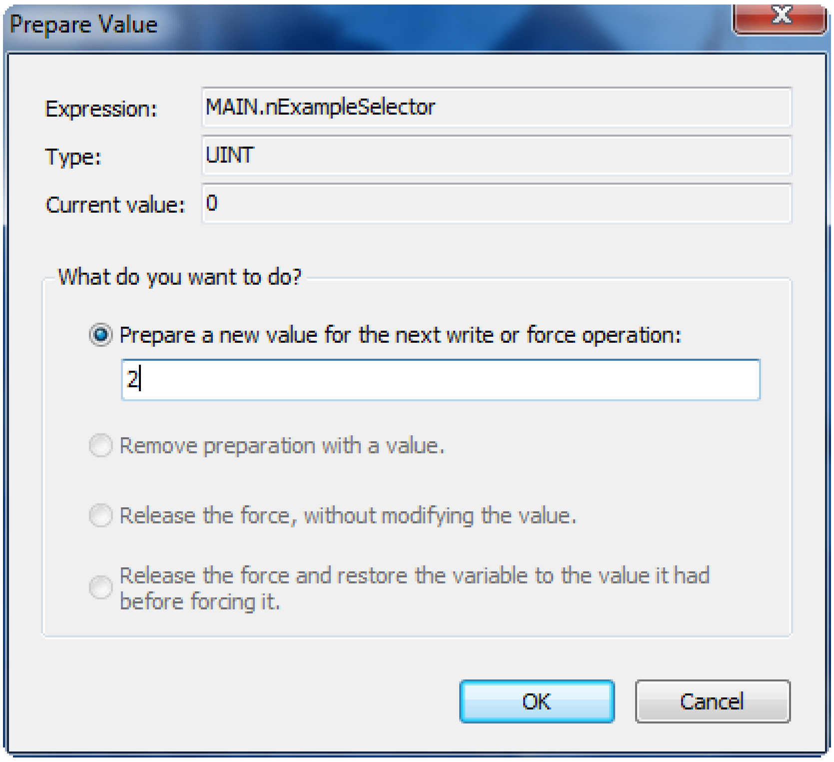
TwinCAT ScopeView allows the signal curves from the individual examples to be displayed graphically. The settings supplied in *TcControllerToolboxExamples_Scope_x.sv* can be used for this purpose. This specification of the Scope file to be used for each particular example can be found in the comments in the project's **MAIN** program.

6. Load the Scope configuration file into the TwinCAT ScopeView.
7. Start the recording via the menu or by pressing the **Record** key.

Select the desired example project

Because the example project contains a number of different examples, it is necessary to set the variable `nExampleSelector` to the appropriate example number in the **MAIN** program.

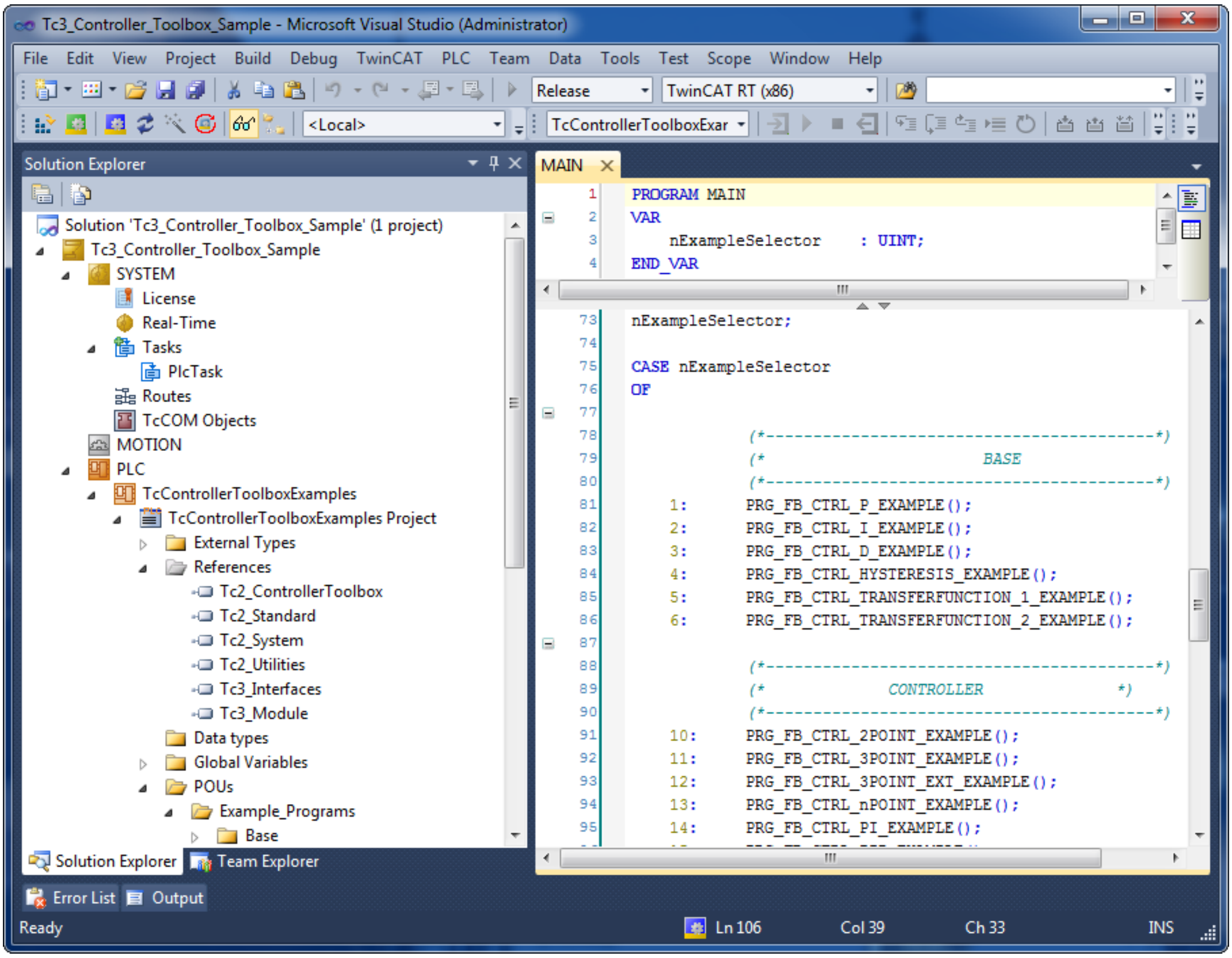
8. Double-click the variable `nExampleSelector`.



9. Enter the number of the example.
10. Click **OK**.
11. Press the **[F7]** key or click **Force Values** in the menu **Online**.

5.2 Example Structure

The **MAIN** module calls up the corresponding sample program in accordance with the variable `nExampleSelector`.



The individual sample programs contain comments for clarity and traceability.

6 Appendix

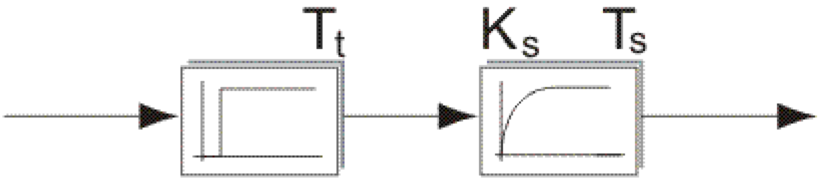
6.1 Setting rules for the P, PI and PID controllers

This page summarizes some of the setting rules found in the relevant literature. The setting rule to be used for a particular case has to be determined depending on the controlled system.

Ziegler and Nichols

The Ziegler and Nichols setting rules can be used if the controlled system can be approximated by a dead time element and 1st order delay element.

$$G_s(s) = K_s \cdot \frac{e^{-T_t s}}{1 + s \cdot T_s}$$



T_t = dead time of the controlled system

K_s = gain factor of the controlled system

T_s = time constant of the controlled system

Regler	K_r	T_n	T_v
P-Regler	$\frac{T_s}{K_s \cdot T_t}$	-	-
PI-Regler	$0.9 \cdot \frac{T_s}{K_s \cdot T_t}$	$3.33 \cdot T_t$	-
PID-Regler	$1.2 \cdot \frac{T_s}{K_s \cdot T_t}$	$2.0 \cdot T_t$	$0.5 \cdot T_t$

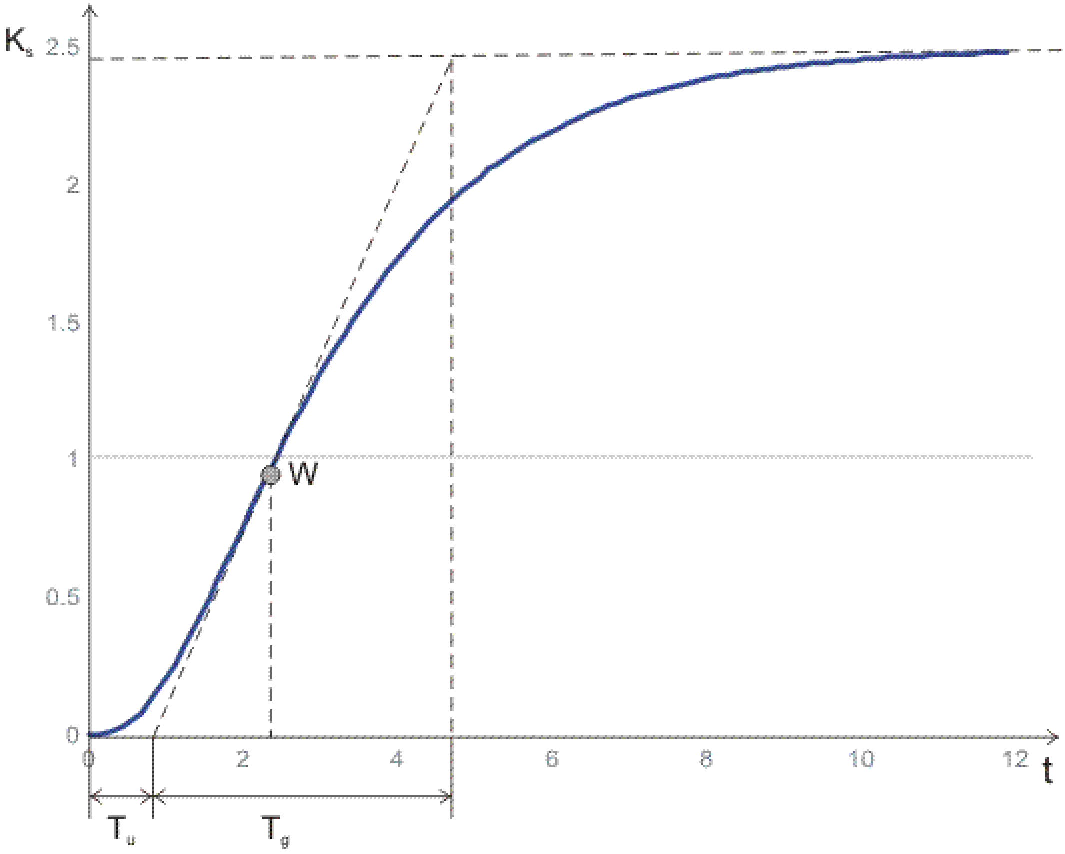
Chien, Hrones and Reswick

If this procedure is to be used, the step response of the system must show delay characteristics and be free from overshoot effects.

From the step response the delay time T_U, the compensation time T_g and the system amplification K_s are determined.

The system amplification is calculated from the following quotient:

$$K_s = \frac{\text{Step Response Height}}{\text{Controller Output Value}}$$



Optimization of the interference behavior:

Regler		Aperiodischer Regelverlauf	20% Überschwingen
P-Regler	K_r	$0.30 \cdot \frac{T_g}{T_u \cdot K_s}$	$0.70 \cdot \frac{T_g}{T_u \cdot K_s}$
PI-Regler	K_r	$0.60 \cdot \frac{T_g}{T_u \cdot K_s}$	$0.70 \cdot \frac{T_g}{T_u \cdot K_s}$
	T_N	$4.00 \cdot T_u$	$2.3 \cdot T_u$
PID-Regler	K_r	$0.95 \cdot \frac{T_g}{T_u \cdot K_s}$	$1.20 \cdot \frac{T_g}{T_u \cdot K_s}$
	T_N	$2.40 \cdot T_u$	$2.00 \cdot T_u$
	T_V	$0.42 \cdot T_u$	$0.42 \cdot T_u$

Optimization of the control behavior

Regler		Aperiodischer Regelverlauf	20% Überschwingen
P-Regler	K_r	$0.30 \cdot \frac{T_g}{T_u \cdot K_s}$	$0.70 \cdot \frac{T_g}{T_u \cdot K_s}$
PI-Regler	K_r	$0.35 \cdot \frac{T_g}{T_u \cdot K_s}$	$0.60 \cdot \frac{T_g}{T_u \cdot K_s}$
	T_N	$1.20 \cdot T_g$	$1.0 \cdot T_g$
PID-Regler	K_r	$0.60 \cdot \frac{T_g}{T_u \cdot K_s}$	$0.95 \cdot \frac{T_g}{T_u \cdot K_s}$
	T_N	$1.00 \cdot T_g$	$1.35 \cdot T_g$
	T_V	$0.50 \cdot T_u$	$0.47 \cdot T_u$

Trademark statements

Third-party trademark statements

More Information:
www.beckhoff.com/tf4100

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