Concept 2.6
Block Library IEC
Part: CONT_CTL

01/2007

33002211.04

a brand of Schneider Electric

Telemecanique

www.telemecanique.com

Table of Contents

Safety Information

Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

The addition of this symbol to a Danger or Warning safety label indicates that an electrical hazard exists, which will result in personal injury if the instructions are not followed.

This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

DANGER

DANGER indicates an imminently hazardous situation, which, if not avoided, **will result** in death or serious injury.

WARNING

WARNING indicates a potentially hazardous situation, which, if not avoided, **can result** in death, serious injury, or equipment damage.

CAUTION

CAUTION indicates a potentially hazardous situation, which, if not avoided, **can result** in injury or equipment damage.

PLEASE NOTE Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

About the Book

At a Glance

Document Scope

This documentation will assist you when configuring functions and Function blocks.

Validity Note

This document applies to Concept 2.6 under Microsoft Windows 98, Microsoft Windows 2000, Microsoft Windows XP and Microsoft Windows NT 4.x.

Note: Additional up-to-date tips can be found in the README data file in Concept.

Related Documents

Title of Documentation	Reference Number
Concept Installation Instructions	840 USE 502 00
Concept User Manual	840 USE 503 00
Concept-EFB User Manual	840 USE 505 00
Concept LL984 Block Library	840 USE 506 00

You can download these technical publications and other technical information from our website at *www.telemecanique.com*

User Comments

We welcome your comments about this document. You can reach us by e-mail at techpub@schneider-electric.com

General information about the block library CONT_CTL

I

Overview

At a glance

This section contains general information about the block library CONT_CTL.

What's in this Part?

This part contains the following chapters:

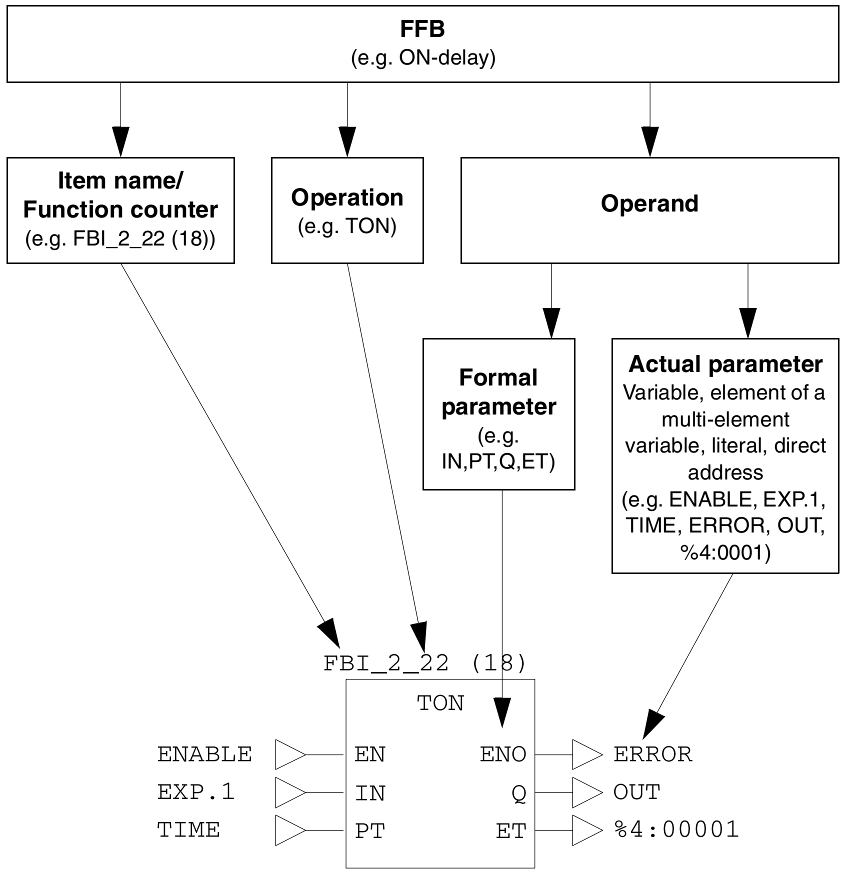
Parameterizing functions and function blocks

1

Parameterizing functions and function blocks

General

Each FFB consists of an operation, the operands needed for the operation and an instance name or function counter.



Operation

The operation determines which function is to be executed with the FFB, e.g. shift register, conversion operations.

Operand

The operand specifies what the operation is to be executed with. With FFBs, this consists of formal and actual parameters.

Formal/actual parameters

The formal parameter holds the place for an operand. During parameterization, an actual parameter is assigned to the formal parameter.

The actual parameter can be a variable, a multi-element variable, an element of a multi-element variable, a literal or a direct address.

Conditional/ unconditional calls

"Unconditional" or "conditional" calls are possible with each FFB. The condition is realized by pre-linking the input EN.
- Displayed EN
  conditional calls (the FFB is only processed if EN = 1)
- EN not displayed
  unconditional calls (FFB is always processed)

Note: If the EN input is not parameterized, it must be disabled. Any input pin that is not parameterized is automatically assigned a "0" value. Therefore, the FFB should never be processed.

Note: For disabled function blocks (EN = 0) with an internal time function (e.g. DELAY), time seems to keep running, since it is calculated with the help of a system clock and is therefore independent of the program cycle and the release of the block.

Calling functions and function blocks in IL and ST

Information on calling functions and function blocks in IL (Instruction List) and ST (Structured Text) can be found in the relevant chapters of the user manual.

General information on the CONT_CTL block library

2

Introduction

At a glance

This section contains general information on the CONT_CTL block library.

What's in this Chapter?

This chapter contains the following topics:

Groups in the CONT_CTL block library

Overview of the groups

The "Continuous Control"(CONT-CTL) library consists of 7 groups with Elementary function blocks (EFBs):

Groups	Contents
CLC	Contains closed loop control function blocks such as filters, controllers, integrators and Deadtime devices
CLC_PRO	Contains a further selection of closed loop control function blocks
Conditioning	EFBs for processing the measurement or another discrete variable
Controller	Controller EFBs and automatic closed control loop blocks
Mathematics	EFBs for mathematical control functions
Output Processing	EFBs for controlling the various actuator types
Setpoint Management	EFBs for generating and selecting the setpoint

"CLC" group

This group contains the following EFBs:

Block	Meaning
DELAY	Deadtime device
INTEGRATOR1	Integrator with limit (Operating modes, Manual, Halt, Automatic)
LAG1	Time lag device: 1st order
LEAD_LAG1	PD device with smoothing
LIMV	Velocity limiter: 1st order
PI1	PI controller
PID1	PID controller
PIDP1	PID controller with parallel structure
SMOOTH_RATE	Differentiator with smoothing
THREEPOINT_CON1	Three point controller
THREE_STEP_CON1	Three-step step-action controller
TWOPOINT_CON1	Two-step controller

"CLC_PRO" group

This group contains the following EFBs:

Block	Meaning
ALIM	Velocity limiter: 2nd order
COMP_PID	Complex PID controller
DEADTIME	Deadtime device
DERIV	Differentiator with smoothing
FGEN	Function generator
INTEG	Integrator with limit
LAG	Time lag device: 1st order
LAG2	Time lag device: 2nd order
LEAD_LAG	PD device with smoothing
PCON2	Two-step controller
PCON3	Three point controller
PD_or_PI	Algorithm-adaptive PD/PI controller
PDM	Pulse duration modulation
PI	PI controller
PID	PID controller
PID_P	PID controller with parallel structure
PIP	PIP cascade controller
PPI	PPI cascade controller
PWM	Pulse width modulation
QPWM	Pulse width modulation (simple)
SCON3	Three-step step-action controller
VLIM	Velocity limiter: 1st order

"Conditioning" group

This group contains the EFBs for processing procedures which come before the controllers in general, such as the processing of the measurements of the controlled variables, the disturbance variables or other discrete variables.

This group also contains delay and summation functions beyond filters and other classic functions.

This group contains the following EFBs:

Block	**Meaning**
DTIME	Delay function, for increased precision or for dynamic (online) modification of the delay value
INTEGRATOR	Integrator with limit (Tracking and automatic operating modes)
LAG_FILTER	Time lag device: 1st order
LDLG	PD device with smoothing (phase advance/delay)
LEAD	Differentiator with smoothing
MFLOW	Controller for mass flow, e.g. for processing the differential pressure measurement of a throttle device
QDTIME	Deadtime device, delay function for quick parametering (Q = Quick)
SCALING	Scaling of all discrete variables
TOTALIZER	An integrator for integrating a flow and thereby calculating a flow volume. Very small values can be taken into account with this EFB, even if the total volume is large. It has a partial amount and a total amount counter.
VEL_LIM	Limiting the input or intermediate variable velocity

"Controller" group

The contents of this group a block for autotuning (AUTOTUNE). This block is standardized with the PI_B and PIDFF controller blocks. Self-tuning controller applications can be programmed with this.

This group contains the following EFBs:

Block	**Meaning**
AUTOTUNE	Autotuning
PI_B	Simple PI controller
PIDFF	Complete PID controller
STEP2	Two-step controller
STEP3	Three point controller

"Mathematics" group

Arithmetic functions are often used in connection with dead zones and weightings in the regulation zone.

This group covers directly applicable arithmetic functions on the basis of this principle.

- Multiplication / division with weighting: MULDIV_W
- Summation with weighting: SUM_W
- Comparison with dead zone and hysteresis: COMP_DB
- Square root with division and weighting K_SQRT

This group contains the following EFBs:

Block	Meaning
COMP_DB	Comparison
K_SQRT	Square root
MULDIV_W	Multiplication / division
SUM_W	Summer

"Output processing" group

It is often not possible to use the controller output directly to control the actuator.

If for example, as in the case of many processes, electric server motors are in use, a SERVO function block must be switched to the controller.

If two actuators are affecting the same variable, the SPLRG function block should be used. This function block functions both as a three step controller (when the actuators have an opposing effect) and in the "Split range" operating mode (when the actuators have an equal effect).

The PWM1 block enables pulse width modulation, for example of a setting variable of a pre-enabled continuous controller (PI, PID).

Although all the controller blocks can work in manual operating mode, it is often necessary to used the MS function block for this purpose.

This block enables extended control of manual operation mode

- The variable to be controlled is not the control output directly
- The output is not controlled via a servo loop
- The servo loop has a long sampling interval (1s and over)

This group contains the following EFBs:

Block	Meaning
MS	Manual control of an output
PWM1	Pulse width modulation
SERVO	Control for electric server motors
SPLRG	Controlling two actuators

Setpoint Management group

The classic ‘Select Setpoint’ function is integrated into the SP_SEL function rather than the control elements. This modular structure enables greater flexibility and improved user comfort without losing extended functions.

This includes the following:
- Tracking the process value if the servo loop is set to manual mode
- Bumpless switchover internal/external
- Bumpless extern/intern changeover (with setpoint tracking)

Two other function blocks make it possible to generate the setpoint to be switched to the controller: the RATIO function block, which is used to control a variable depending on a different variable (relationship control) and the RAMP block, which makes it possible to generate a setpoint in ramp form.

This group contains the following EFBs:

Block	Meaning
RAMP	Ramp generator
RATIO	Ratio controller
SP_SEL	Setpoint switch

Operating mode

Operating mode

Several function blocks have integrated operating mode control available.

A choice can be made between the following operating mode:
- Tracking
- Manual/Automatic

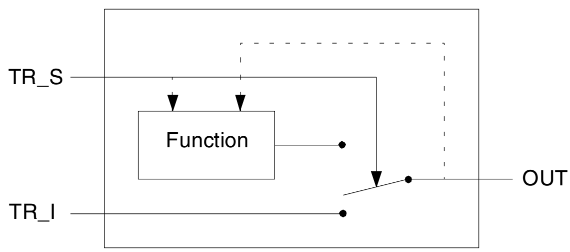
The Order of priorities of the operating mode is explained further.

Tracking

This operating mode makes it possible to set a function block to the ‘Sub Controller’ operating mode. Two inputs make it possible to control this operating mode: a binary input TR_S (TRacking Switch), and a signal input TR_I (TRacking Input). If a function block is in tracking mode (TR_S = 1), its main output (e.g. OUT with a PIDFF controller) is assigned the input value TR_I and the internal variables of the different algorithms are updated. In this way a bumpless changeover is guaranteed when the function block is switched to manual or automatic mode.

The OUT output of the FFB is controlled with the TR_I input in tracking mode.

Tracking operating mode



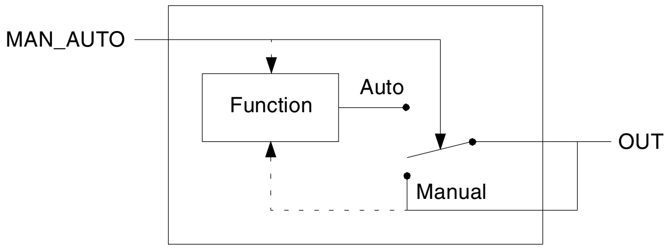
This operating mode can be used in various situations:
- Initializing during the start phase,
- Tracking operating mode with a redundant PLC, to guarantee a bumpless start for the Standby device,
- Controlling the operating mode using a program, for example to avoid direct control of the manipulated variable, when an automatic controller setting is in progress, etc.

A limit can be assigned to the function block’s output if it is in tracking operating mode: this should be decided separately for the individual function blocks.

Manual/ Automatic

If a function block is in automatic mode, its algorithm calculates the value to be assigned to the output. Manual mode can be used to bar the adjustment of the main output (OUT) of a function block, to permit control via a user dialog, for example. The MAN_AUTO input permits control of this operating mode (0 : Manual, 1: Automatic).

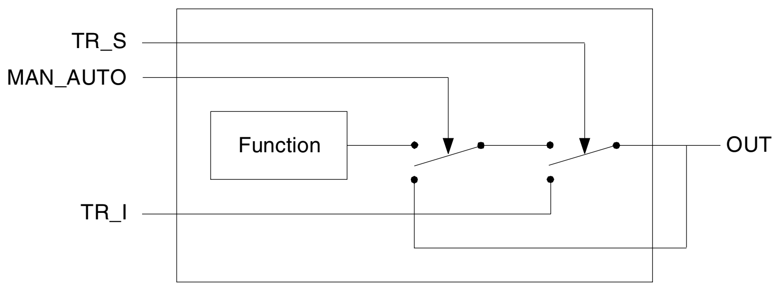
Manual/Automatic mode



The function block reads this output, however, and thus permits a bumpless changeover between the Manual <-> Automatic modes. A limit can be assigned to the function block's output if it is in manual or automatic mode: this should be decided individually for each function block.

Order of priorities of the operating mode

If a function block has both operating mode available, the tracking operating mode has priority over the manual/automatic mode:



The connections between the function and the operating mode of the function block are not displayed to ensure a better overview. The same applies to the effectively assigned setpoint.

Scanning

Scanning

The control algorithms are based on scan values where the time interval between two consecutive cycles should be taken into account. The function blocks calculate the value of this interval automatically, which means they can be placed anywhere in the Concept section without any need to take the time management into account.

The following control functions can be done with a fixed time interval :
- Run time optimization of the PLC program by dividing the control operations into several cycles,
- improved control quality, where scanning the servoloop too frequently is prevented
- Minimizing the demands on the tuning device

For example, the SAMPLETM function block can be used, which should be attached to the input EN of the function block to be scanned.

If the scan interval of the servoloop exceeds 1 second, the function block *MS: Manual control of an output, p. 215* should be switched to the function blocks *PIDFF: Complete PID controller, p. 341* and *PI_B: Simple PI controller, p. 283* so that the servoloops can be controlled manually independently of the scan interval.

Error management

Principle

Most of the function blocks of the groups "Conditioning", "Controller", "Output Processing" and "Setpoint Management" have a STATUS output word available. The error recording and notification procedures used by these function blocks are described in this chapter.

Each bit of the STATUS parameter can be used for notifying an error, an alarm or some information. The meaning of the first 8 bits of the STATUS word is the same for all modules. The meaning of the subsequent bits (bits 8 to 15) is different for each function block.

Status word

The following table shows the meaning of the bits common to all the function blocks in the first byte of the STATUS word. Further information can be found in the description of each function block.

Bit	Meaning	Type
Bit 0 = 1	Error in a calculation with floating point values (e.g. calculation of the square root of a negative number)	Error
Bit 1 = 1	An unauthorized value being recorded on a floating point input can be caused by the following: • the value is not a floating point value • the value is infinite (e.g. the result of a calculation previously enabled to the function block)	Error
Bit 2 = 1	Division by zero with calculation in floating point values	Error
Bit 3 = 1	Capacity overflow with calculation in floating point values	Error
Bit 4 = 1	An input parameter is outside the zone. The value internally used by the function block is capped.	Warning or information (Note 1)
Bit 5 = 1 (Note 2)	The main output of the function block has reached the lower threshold	Information
Bit 6 = 1 (Note 2)	The main output of the function block has reached the upper threshold	Information
Bit 7 = 1	The lower and upper threshold of the input parameter zone are identical	Error

Note 1 (input parameter)

> **Note:** If the value originates from a parameter zone with derived data types (typically the PARA parameter), a warning is given because of the capping and bit 4 is set to 1. If the value originates from a simple type of inputs, no warning is given, but bit 4 of the STATUS word is set to 1.

Note 2 (thresholds)

> **Note:** If the upper and lower threshold parameters of an output have been invented (e.g.. out_min >= out_max), the function block switches the output to the lowest value (i.e. to out_max).

Convention

Specifying the convention

If a Boolean parameter is used to differentiate between 2 operating mode or 2 states of a function block, its name often has the following form: mode1_mode2 (Example: MANU_AUTO, SP_RSP). It is usually specified that the mode1 corresponding value is 0 and the mode2 corresponding value is 1. If for example the MANU_AUTO parameter of a function block is 0, the function block is in manual mode. It is in automatic mode when MANU_AUTO is equal to 1.

EFB Descriptions (A to PH)

II

Overview

Introduction

The EFB descriptions are arranged in alphabetical order.

Note: The number of inputs of some EFBs can be increased (up to a maximum of 32) by vertically resizing the FFB symbol. For information on which EFBs have this capability, please see the descriptions of the individual EFBs.

What's in this Part?

This part contains the following chapters:

ALIM: Velocity limiter: 2nd order

3

Overview

At a glance

This chapter describes the ALIM block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block produces velocity limiter: 2nd order.

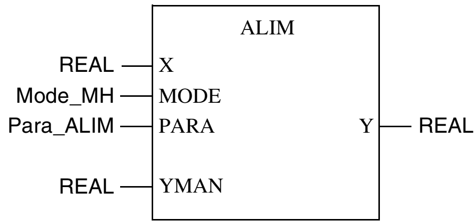
The function block individually contains the following properties:
- Operating mode, Manual, Halt, Automatic
- Output limiting

EN and ENO can be projected as additional parameters.

Presentation

Symbol

Block display:



Parameter description ALIM

Block parameter description:

Parameter	Data type	Meaning
X	REAL	Input
MODE	Mode_MH	Operating mode
PARA	Para_ALIM	Parameter
YMAN	REAL	Manual value for output Y
Y	REAL	Output

Parameter description Mode_MH

Data structure description:

Element	Data type	Meaning
man	BOOL	"1" = Operating mode Hand
halt	BOOL	"1" = Halt mode

Parameter description Para_ALIM

Data structure description:

Element	Data type	Meaning
max_v	REAL	Maximum upper speed (maximum x') Unit: 1/[s]
max_a	REAL	Maximum speed increase (maximum x') Unit: $1/s^2$

Detailed description

Parametering

The parametering of the function block appears through determination of the maximum upper speed max_v as well as the maximum speed increase max_a. The maximum upper speed specifies to which value the output Y can change within one second. The maximum speed increase specifies the maximum value the output Y can change speed at.

The value of Y follows the value of X, but is limited by the maximum permitted speed and speed increase.

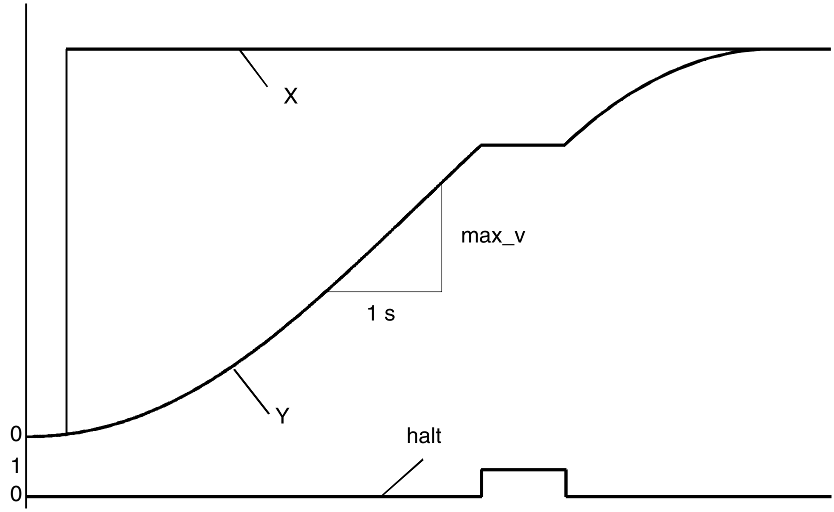
Operating mode

There are three operating mode selectable through the man and halt parameter inputs:

Operating mode	man	halt	Meaning
Automatic	0	0	A new value for Y will be constantly calculated and issued.
Manual mode	1	0 or 1	The manual value YMAN will be transmitted fixed to the output Y.
Halt	0	1	The output Y will be held at the last calculated value. The output will no longer be changed, but can be overwritten by the user.

Example

In the diagram the dynamic behavior of the function block is displayed as well as the reaction during HALY operating mode.



The jump at input X causes the function block to react with an accelerated increase of output Y. Output Y is accelerated with an acceleration increase determined by parameter max_a. Should the slew rate reach the max_v value, acceleration stops, but output Y continues to follow input X with the maximum slew rate max_v (see the straight section in the middle of the figure).

If the value of output Y is close enough to input signal value, the output is reversed to brake at a negative speed increase of –max_a, so that the output does not come to an abrupt stop, but slowly approximates the terminal point.

Runtime error

Error message

There is an Error message, if

- an invalid floating point number lies at input YMAN or X,
- max_a or max_v is ≤ 0.

AUTOTUNE: Automatic regulator setting

4

Overview

At a glance

This chapter describes the AUTOTUNE block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This Function block enables the autotuning of the PID controller (*PIDFF: Complete PID controller, p. 341*, *PI_B: Simple PI controller, p. 283*).

Autotuning stabilizes the control when starting the system and, in so doing, saves time.

EN and ENO can be configured as additional parameters.

Algorithm

The algorithm is based upon heuristic controls, as with the Ziegler Nichols method. Initially, an analysis corresponding to approximately 2.5 times the reaction time of the open loop is performed. Through this, the process can be identified as a process of the first order with delay.

Building on this model, a control parameter set based on heuristic controls and historical data is created.

The parameter range is determined by the "perf." criteria. In this individual case, this factor gives the highest rank to the reaction time to disturbances or stability.

The algorithm is applied to the following process types :
- Processes with only one input / output
- Processes with natural stability or integral components
- Asymmetric processes within the limits authorized by the algorithm of the PID controller
- Processes controlled via pulse width modulation output (PWM).

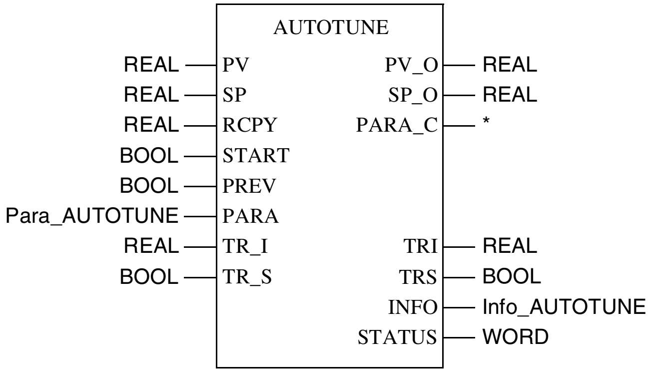
Important characteristics

The block has the following characteristics
- Pre-estimation of the control for the types PIDFF and/or PI_B
- Diagnostic function
- Parametering of the control dynamic
- Recovery of previous control settings

Representation

Symbol

Block representation



* Parameters of the autotuned controller (Para_PIDFF, Para_PI_B,…etc.)

AUTOTUNE parameter description

Block parameter description

Parameter	Data type	Meaning
PV	REAL	Process value
SP	REAL	Setpoint
RCPY	REAL	Copy of the actual manipulated variable
START	BOOL	"0 → 1" : Starting the autotuning
PREV	BOOL	Reverting to the previous controller settings
PARA	Para_AUTOTUNE	Parameter
TR_I	REAL	Start input
TR_S	BOOL	Start command
PV_O	REAL	Copy of the process value PV
SP_O	REAL	Copy of the SP input
PARA_C	Parameters of the autotunable controller (Para_PIDFF or. Para_PI_B)	Control parameters
TRI	REAL	Copy of the TR_I input
TRS	BOOL	Copy of the TR_S input
INFO	Info_AUTOTUNE	Information
STATUS	WORD	Status word

Parameter description Para_ AUTOTUNE

Data structure description

Element	Data type	Meaning
step_ampl	REAL	Value of the output actuating pulse (expressed in output scale values out_inf, out_sup)
tmax	TIME	Duration of the actuating pulse in autom. Tuning
perf	REAL	Performance index between 0 and 1
plant_type	WORD	Reserved word

Info_AUTOTUNE parameter description

Data structure description

Element	Data type	Meaning
diag	UDINT	Double word used for diagnosis
p1_prev	REAL	Previous value of parameter 1
p2_prev	REAL	Previous value of parameter 2
p3_prev	REAL	Previous value of parameter 3
p4_prev	REAL	Previous value of parameter 4
p5_prev	REAL	Previous value of parameter 5
p6_prev	REAL	Previous value of parameter 6

Principle of the autotuning

Two kinds of autotuning

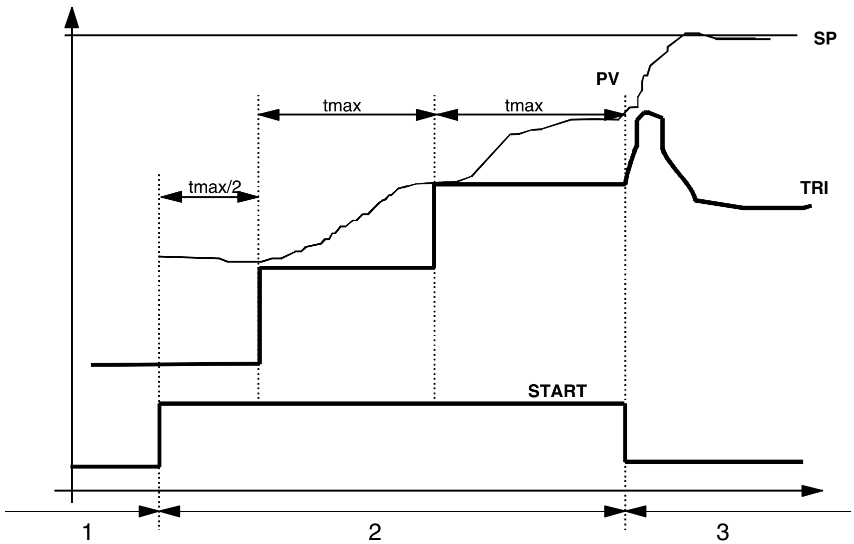
Two kinds of autotuning are possible autotuning at a warm and cold system start

The first phase of autotuning applies for both kinds of tuning: this involves a sound and stability test of the control process lasting 0.5 * tmax with constant outputs. Subsequent phases depend on the kind of tuning.

Autotuning at a cold start

Autotuning at a cold start is referred to when the deviation between the process and setpoint values exceeds 40% and the process value is less than 30%. In this case the TRI output of the function block is admitted with two actuator pulses of the same kind. Each actuator pulse has duration tmax. When autotuning ends, there is a smooth return to the previous operating mode for the servo loop:

Autotuning at a cold start

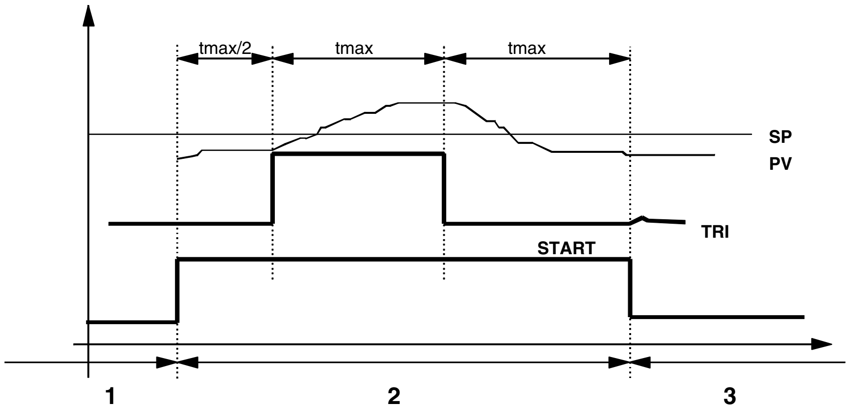


1 Automatic or manual mode
2 Autotune mode
3 Automatic or manual mode

Autotuning at a warm start

If the conditions for autotuning at a cold start are not fulfilled, tuning at a warm start takes place: the output is admitted with an actuator pulse, followed by an actuator pulse in the opposite direction. Each stage has duration tmax. When autotuning ends, there is a smooth return to the previous operating mode for the servo loop:

Autotuning at a warm start



1 Automatic or manual mode
2 Autotune mode
3 Automatic or manual mode

Identification principle

Identification process

The identification process consists of 3 stages:
- a sound and stability analysis of the control process
- an initial analysis of the reaction to an actuator pulse, which is shown as the first identification model: a filter is created on the basis of this first estimate; this is used during the last phase
- a second analysis of the reaction to a second actuator pulse gives more precise information because of the data filter

Finally, a complete process model is created. If the results of the two previous phases are two far apart, the estimate is abandoned and autotuning fails.

Control principle

After both phases a parameter set is created for the controller being tuned. The resulting control parameters are based on the gain and on the ratio between reaction time and process delay.

The algorithm must be able to withstand the modification of the gain and the time constants in ratio 2 without losing stability. The asymmetrical processes are supported if they fulfil these conditions. If not, an error is displayed during diagnosis diag.

Parametering

Parametering actuating pulse

During autotuning, the output TRI is turned up two actuating pulses. An actuating impulse is identified by two parameters: its time duration (tmax) and its amplitude (step_ampl.).

The following value ranges are valid for these parameters: tmax greater than 4 seconds and step_ampl greater than 1 % of the output scale (out_inf, out_sup). The function also monitors even if the TRI output exceeds the threshold for the output scale.

The check occurs when autotune is started.

The following table contains parameter values for some of the typical control methods:

Diagram	tmax (s)	step_ampl (%)
Vol. flow or pressure from liquids	5-30	10-20
Gas pressure	60-300	10-20
Level	120-600	20
Steam temperature or pressure	600-3600	30-50
Module	600-3600	30-50

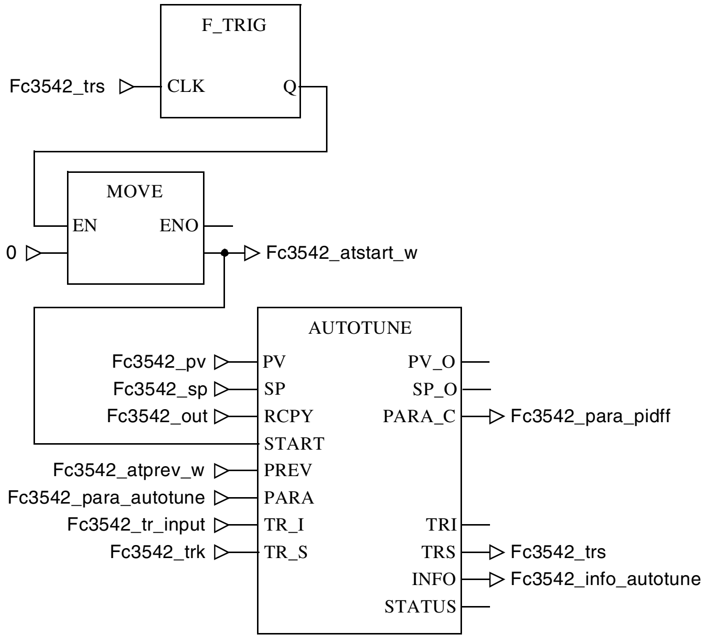
Performance index: perf

The controller can be modulated for each value in the performance index. The perf. performance index varies between 0 and 1, which enables the perf. parameter to stabilize close to 0 or to achieve a more dynamic control (and therefore optimize the reaction time of disturbance variables), if the perf. is set close to 1.

Starting the autotune: START

If this bit is set to 1, the function is activated. At the end of the setting process, this bit must be manually set to 0. If it has just been set automatically, setting the bit to 0 allows the function to be stopped. The PARA_C then retain the last active value. In the example below, the START bit is automatically reset by the program at the end of the setting process.

Example for starting the autotuning



Reverting to the previous setting: PREV

A modification of this bit value enables the exchange of current and previous parameters assuming that no controlling has occurred up to the given time (two consecutive modifications of this bit give the original configuration).

The following Info_AUTOTUNE structural parameters are valid for PIDFF type controllers:

Element of the data structure	Meaning
p1_prev	KP
p2_prev	TI
p3_prev	TD

The following Info_AUTOTUNE structural parameters are valid for the controllers of the PI_B type.

Element of the data structure	Meaning
p1_prev	KP
p2_prev	TI

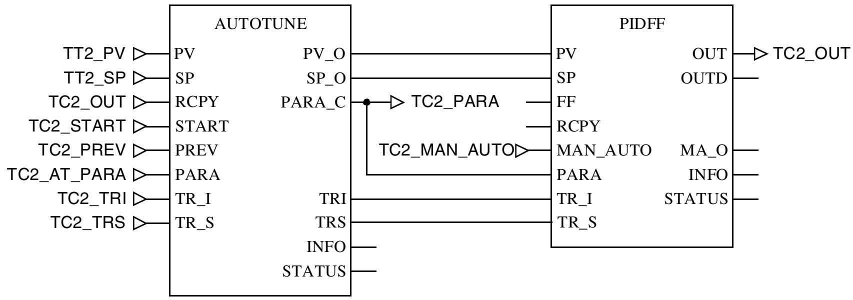
Diagnosis during autotuning: diag

The diagnosis data for the autotune is saved in a double word. The value of this word is retained until autotune is restarted. Additional details on this double word can be found in the Diagnosis section.

Controller coupling

Application example with a PIDFF controller type EFB

The following diagram is an application example of an AUTOTUNE EFB with a PIDFF controller type EFB :



The AUTOTUNE EFB exchanges with the controller parameter: Access to the controller parameters is via the link between the output PARA_C of the AUTOTUNE function block and the input PARA of the controller. The PARA_C output is of the ANY type and enables the connection of the AUTOTUNE EFB to various controller types (PIDFF or PI_B).

The AUTOTUNE EFB and the controller also share the following interlinkable variables: PV, SP, TR_I and TR_S. These variables display AUTOTUNE inputs, which lead to the corresponding outputs, in order to switch to controller inputs

If the autotune is active, the TRS output transfers to 1 and the manipulated variable is attached at the TRI output. The purpose of these outputs is to connect to the inputs TR_I and TR_S of the function blocks following AUTOTUNE. In this way, these can be set to the tracking operation mode (PIDFF, PI_B, MS,…).

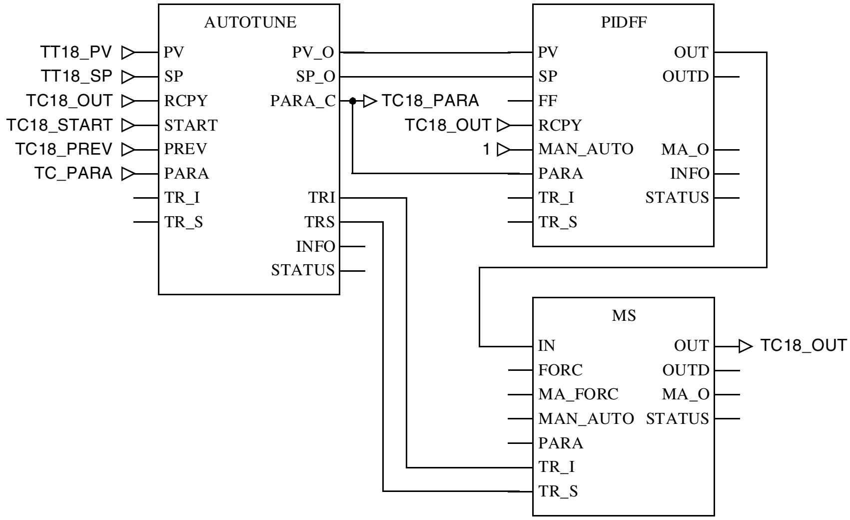
Example for connection: Servoloops with a simple PID controller

This section is concerned with the automatic setting of a single controller (most frequent case). The controller can be of PI_B or PIDFF type.

The AUTOTUNE EFB requires the scaling parameters of the controller (PARA_C structure parameters) pv_inf, pv_sup, out_inf, out_sup as well as the controller’s structure type, which is specified via the mix_par bit. The EFB creates the parameters of the PID controller (KP, TI, TD) from this. The direction of action of the controller (rev_dir) is checked when testing the autotune and is compared to the sign for the gain of the model. When incompatibility occurs, an error is shown for the "diag." Parameters.

Example for connection: Servoloops with simple PID controller and MS function block

If the servoloop contains a MS-EFB, the structure can appear as follows:



When starting the autotune, the AUTOTUNE EFB sets the MS function block to tracking mode and hence controls the output of the servoloop directly. Using AUTOTUNE and PIDFF blocks' RCPY inputs enables a bumpless restart of the servoloop.

Operating modes

Operating modes

The various operating modes of the autotuning and their priorities in descending order of validity are shown in the following table:

Operating mode	TR_S	START
Tracking	1	1 or 0
Autotuning	0	1

On completion of the autotuning, the TRS output is set to 0, so as the servoloop is set back to its previous operating mode (manual or automatic). If the autotuning fails, the TRI variable will be set back to its value from before the autotuning was started and the servoloop will be set back to its previous operating mode.

Diagnosis

Overview of the diagnosis

There are a number of reasons that can lead to the autotuning not starting, being cancelled or failing. In such a case, depending on the cause of failure, it can be possible to supply a parameter set. Every bit of the diagnostic word diag. allows for a type of error to be created.

This word contains the current operating mode of the autotuning.

The following cases are explained:

- *Status of the autotuning, p. 61*
- *Causes of a faulty start, p. 62*
- *Causes of autotuning termination, p. 63*
- *Generating a test after stopping the autotuning, p. 65*

Diagnostic word

The meaning of the data structure Info_AUTOTUNE element diag can be found in this table.

Bit	Meaning
Bit 0 = 1	Autotuning is running
Bit 1 = 1	Autotuning aborted
Bit 2 = 1	Parameter error
Bit 3 = 1	Alteration of parameters, which have just been set automatically
Bit 4 = 1	Stop as a consequence of system error
Bit 5 = 1	Process value saturated
Bit 6 = 1	Alteration too small
Bit 7 = 1	Sampling interval invalid
Bit 8 = 1	Incomprehensible reaction
Bit 9 = 1	Non-stabilized measuring at the start
Bit 10 = 1	Length of actuating pulse (tmax) too short
Bit 1 1= 1	Too much noise/interference
Bit 12 = 1	Length of actuating pulse (tmax) too long
Bit 13 = 1	Process with significant exceeding of the thresholds
Bit 14 = 1	Process without minimum phase
Bit 15 = 1	Asymmetric process
Bit 16 = 1	Process with integral component

Status of the autotuning

Overview

The following bits of the diagnostic word (the diag element) show the status of the autotuning.

Bit	Meaning
0	1 = automatic regulator setting is running
1	1 = automatic regulator setting is stopped

Bit 0 of the element diag

This Bit indicates that the automatic regulator setting is running. On quitting the automatic regulator setting or terminating using the START-Bit, this is set to zero.

Bit 1 of the element diag

This Bit indicates that the user stopped the last control by means of the START-Bit or by setting the operating mode to Tracking.

Causes of a faulty start

Overview

The following bits of the diagnostic word (see element diag) indicate a faulty start:

Bit	Meaning
2	1 = Parameter error
7	1 = incorrect sampling interval

Bit 2 of the element diag

The following causes can lead to a faulty start :
- Length of actuating pulse too short (tmax < 4 s),
- Amplitude too weak (step_ampl < 1% of output range),
- Cannot perform this protocol: If the output + n x the amplitude of the actuating pulse (where n = 1 for adjustment during a warm start and n = 2 for adjustment during a cold start) is outside the output range (out_inf, out_sup), then the test protocol cannot be used. Step_ampl must be set to a value that is compatible with the current work point.

Bit 7 of the element diag

If the sampling interval is too large in relation to the length of the actuating pulse (> tmax / 25), then the response test is too imprecise and the automatic regulator setting will be blocked. This typically occurs during very rapid regular processes (where tmax is larger than the rise time of the process, a matter of a few seconds). In this case tmax can be increased, because the algorithm reacts only slightly to this parameter (in the ratio of 1 to 3), or alternatively, the sampling interval can be set to correspond.

Causes of autotuning termination

Overview

The following bits of the diagnostic word (see element diag) show the reason for terminating the autotuning:

Bit	Meaning
3	1 = Modification of parameters during tuning
4	1 = Terminated due to system error
5	1 = Process value saturated
6	1 = Ascent too small
8	1 = Illogical reaction

Bit 3 of the element diag

If the parameters tmax or step_ampl are modified during the tuning, the operation will be cancelled.

Bit 4 of the element diag

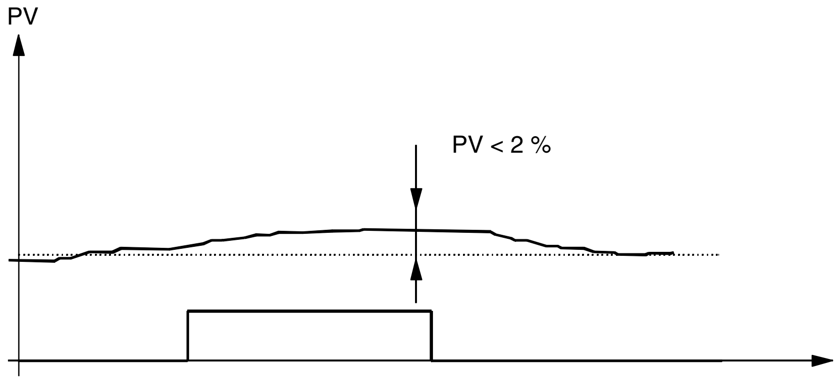
The autotuning will be cancelled if the PLC experiences a system error that prevents the completion of the chain. For example, the function will automatically stop should a voltage return occur.

Bit 5 of the element diag

If the measurement exceeds the range (pv_inf, pv_sup), then the autotuning will be cancelled, and the regulator set to the previous operating mode. Estimating the future measurements enables the autotuning to stop before the range is exceeded (if a first model has been identified).

Bit 6 of the element diag

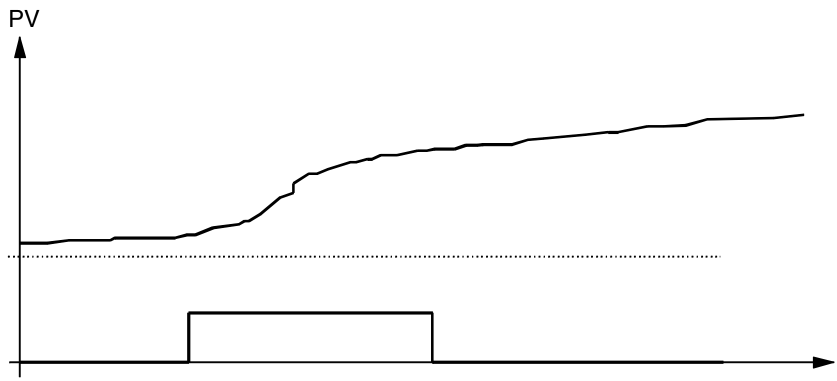
This picture shows the behavior when the ascent is too small:



The amplitude of the actuating pulse is too small too influence the process. In this case, the value of step_ampl can be increased.

Bit 8 of the element diag

This picture shows the behavior during an illogical reaction.



The reaction of the control process is incomprehensible (gain factors with various signs). This can be due to a larger disturbance, coupling with other servoloops or some other reason.

Generating a test after stopping the autotuning

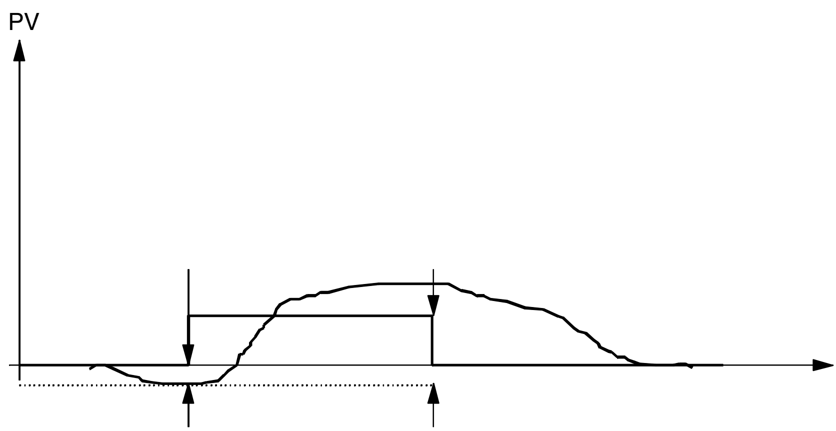
Overview

The following bits of the diagnostic word (see element diag) show the status of the autotuning:

Bit	Meaning
9	1 = Initial non-stabilized measurement
10	1 = Length of actuating pulse (tmax) too short
11	1 = Too much noise/interference
12	1 = Length of actuating pulse (tmax) too long
13	1 = Measured value has been significantly exceeded
14	1 = Process without minimum phase
15	1 = Asymmetrical Process
16	1 = Integrating Process

Bit 9 of the element diag

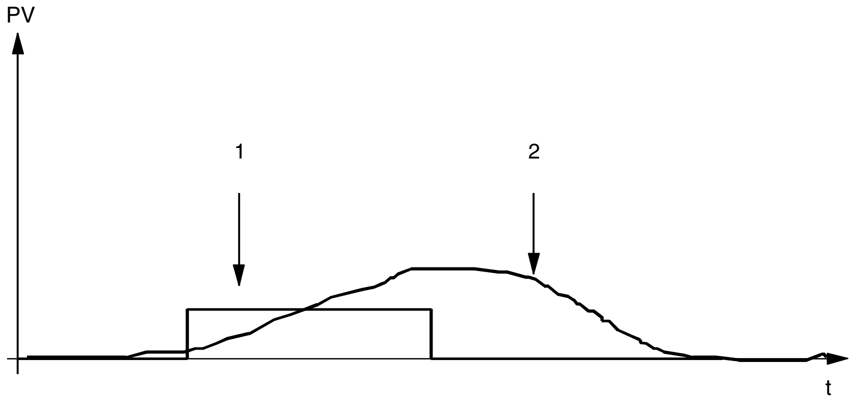
This image illustrates behavior when measurements are not initially stabilized:



The automatic regulator setting was implemented, although the measurement was not stable. If the measured change is large relative to the reaction of the actuating pulse, then the test results will be distorted.

Bit 10 of the element diag

This image illustrates behavior when the actuating pulse is too short:

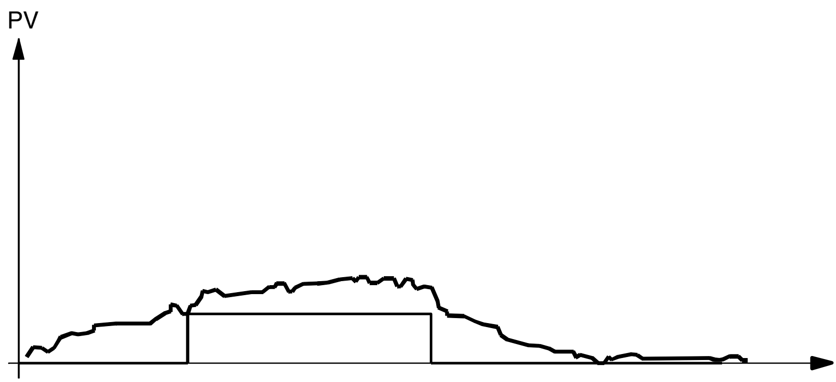


1 Actuating pulse test
2 Process reaction

The reaction will not be stabilized before returning to the original manipulated variable. The calculated parameters are therefore false.

Bit 11 of the element diag

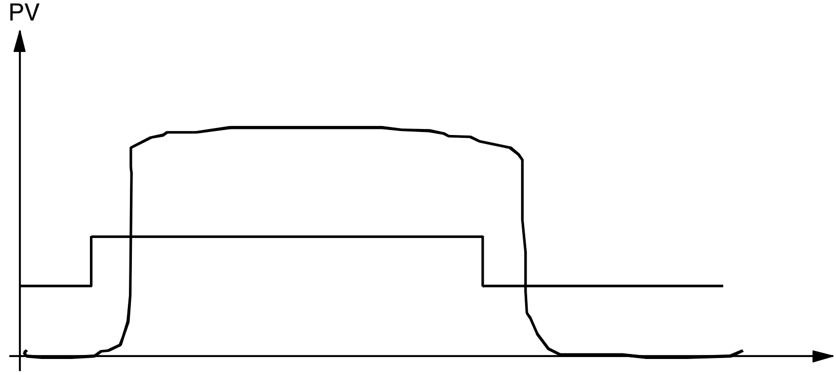
This image illustrates the behavior when noise/interference is too high:



The reaction of the process to the actuating pulse is insufficient relative to the level of noise/interference. The measurement should be filtered or step_ampl should be increased.

Bit 12 of the element diag

This image illustrates behavior when the actuating pulse is too long:



tmax specifies the frequency with which the measurement is taken, i.e. the value that is used to calculate the coefficients. tmax must be between 1 and 5 times the rise time of the repeated task.

Bit 13 of the element diag

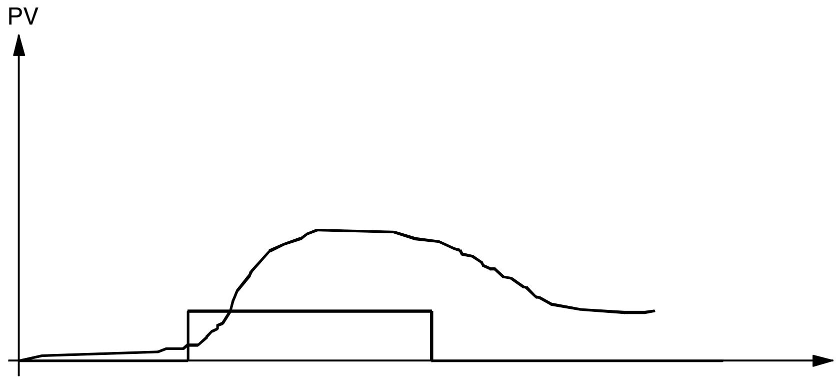
This bit is used when the reaction to an actuating pulse significantly exceeds (overshoots) the measured value (i.e. by more than 10%). The process does not conform to the models used by the algorithms.

Bit 14 of the element diag

This bit is used when the reaction to an actuating pulse leads to inversion of the reaction at the initial stage (i.e. undershoots by more than 10%). The process does not conform to the models used by the algorithms.

Bit 15 of the element diag

This image illustrates the behavior when the process is asymmetrical.



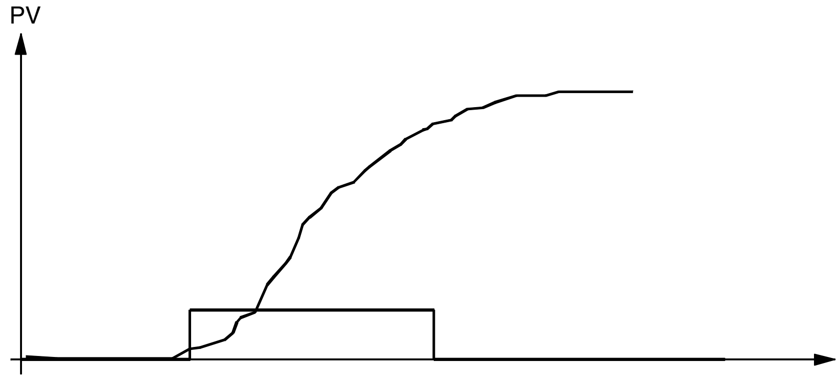
The reaction of the process is asymmetrical.

The last parameter set must be a compromise between the reactions at ascent and descent. Both cases concern average performance.

If the desired criterium is the length of the reaction on ascent, then the first parameter set must be taken into consideration. During the return phase (to the original manipulated variable) the automatic regulator setting is turned off. If the desired criteria is the length of descent, then a negative amplitude must be used.

Bit 16 of the element diag

This image illustrates the behavior during an integration process.



The process includes an integral component or tmax is too small and the process asymmetrical. The calculated coefficients must correlate to the process with the integral coefficient If this is not the case, the automatic regulator setting should be restarted, after tmax has been increased.

Runtime error

Status word

The status word bits have the following meaning:

Bit	Meaning
Bit 0 = 1	Error in a floating point value calculation
Bit 1 = 1	Invalid value recorded at one of the floating point inputs
Bit 2 = 1	Division by zero calculation when calculating in floating point values
Bit 3 = 1	Capacity overflow during calculation in floating point values
Bit 4 = 1	The parameter perf is outside the [0,1] range: in calculating the function block uses the value 0 or 1.
Bit 7 = 1	The thresholds (pv_inf and pv_sup) of the controller to be set are identical
Bit 8 = 1	The PARA_C output is not connected to the parameters of an autotunable controller
Bit 9 = 1	Autotuning failed
Bit 10 = 1	The last autotune was successful

Error message

This error is displayed when a non-floating point has been recorded at an input, when a problem occurs during a calculation with floating points or when the thresholds pv_inf and pv_sup of the controller are identical. In this case, all the outputs of the function block remain unchanged.

Warning

A warning is issued, if the parameter perf is outside the [0,1] range. In this case, the block can use either the value 0 or 1 for the purpose of calculations.

COMP_DB: Comparison

5

Overview

At a glance

This chapter describes the COMP_DB block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The COMP_DB function block enables two numerical values, IN1 and IN2 to be compared.

Depending on whether IN1 is greater, equal to or smaller than IN2, the appropriate output GREATER, EQUAL or LESS is set to 1 by the function block.

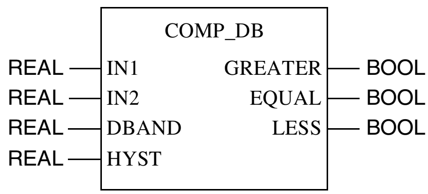
The function block takes any dead zone or hysteresis into account.

EN and ENO can be configured as additional parameters.

Representation

Symbol

Block representation



Parameter description

Block parameter description

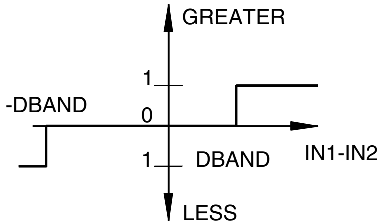
Parameter	Data type	Meaning
IN1	REAL	Input No. 1
IN2	REAL	Input No. 2
DBAND	REAL	Dead zone
HYST	REAL	Hysteresis
GREATER	BOOL	Greater-than marker
EQUAL	BOOL	Equals marker
LESS	BOOL	Less-than marker

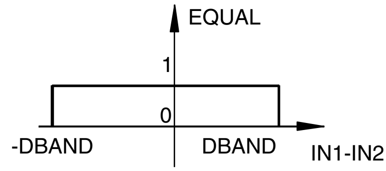
Detailed description

Dead zone

The D_BAND parameter enables a dead zone to be specified, within which deviation between IN1 and IN2 will be regarded as zero. If the deviation IN1 - IN2 remains within this zone, the EQUAL output is set to 1.

Dead zone specification

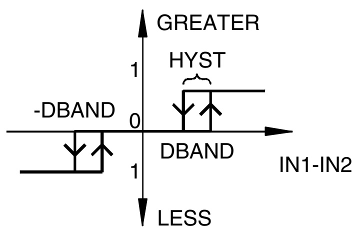


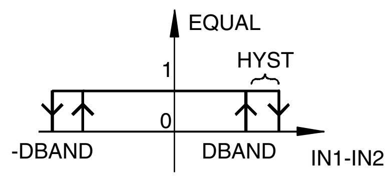


Hysteresis

The HYST parameter enables a hysteresis effect to be generated, if the deviation between IN1 and IN2 decreases: starting from a situation where either the GREATER or LESS output has the value 1, the EQUAL output will only take the value 1 when the deviation IN1 – IN2 is less than DBAND – HYST.

Generating a hysteresis effect

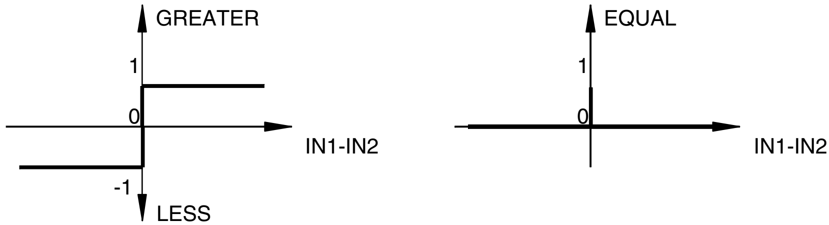




DBAND = 0 and HYST = 0

In this case, the block behaves like a classic comparison function:
- If IN1 is always greater than IN2, then GREATER = 1
- When IN1 is equal to IN2, then EQUAL = 1
- If IN1 is less than IN2, then LESS = 1

Classic comparison function (DBAND = 0 and HYST = 0



Runtime error

Error message

This error appears if a non floating point value is recorded at an input or if there is a problem with a floating point calculation. In this case the outputs GREATER, EQUAL and LESS remain unchanged.

Warning

A warning message appears if:
- The DBAND parameter is negative: the function block then uses the value DBAND=0 for calculation.
- The HYST parameter is outside the [0, DBAND] range: the function block then uses the closest correct value, i.e. if HYST is less than 0 and DBAND, or when HYST is larger than DBAND.

COMP_PID: Complex PID controller

6

Overview

At a glance

This chapter describes the COMP_PID block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block represents a complex PID controller that in its design specifically includes cascade treatment. The control structure is displayed in the *Structure diagram, p. 80*.

EN and ENO can be configured as additional parameters.

Properties

The function block has the following properties:

- real PID controller with independent gain, ti, td setting
- Manual, halt, automatic, cascade, reset, manual value operating modes tracking
- Velocity limit for manual operation
- Adjustable manual manipulated value tracking
- Velocity limit for reference variable
- bumpless changeover between manual and automatic
- Manipulated variable limiting
- bumpless, individually connectable P, I and D components
- bumpless gain modification
- Choice of antiwindup reset and antiwindup halt
- Displacement of antiwindup limits compared to control limits
- Antiwindup measure with an active I component only
- definable delay of the D-component
- D component connectable to controlled variable PV or system deviation EER
- Dead zone with gain reduction
- external operating point (in P, PD and D operation)
- Choice of bump/bumpless manual/automatic switchover

Transfer function

The transfer function is:

$$G(s) = \text{gain} \times \left(1 + \frac{1}{\text{ti} \times s} + \frac{\text{td} \times s}{1 + \text{td\_lag} \times s}\right)$$

YD
YI
YP

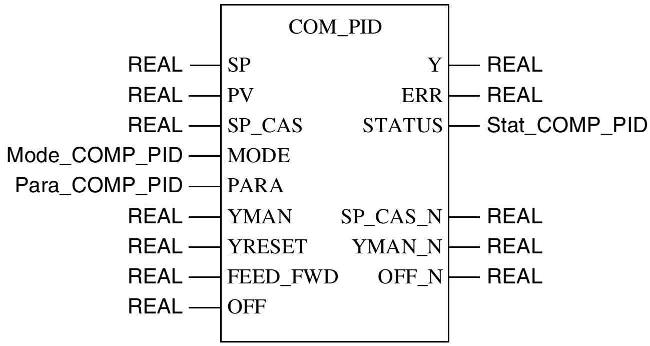
Explanation of the variables:

Variable	Meaning
YD	D component (only if en_d = 1)
YI	I component (only if en_i = 1)
YP	P component (only if en_p = 1)

Representation

Symbol

Block representation:



Parameter description COMP_PID

Block parameter description

Parameter	Data type	Meaning
SP	REAL	Reference variable
PV	REAL	Controlled variable
SP_CAS	REAL	Cascade reference variable
MODE	Mode_COMP_PID	Operating mode
PARA	Para_COMP_PID	Parameter
YMAN	REAL	Manually manipulated value
YRESET	REAL	Manipulated variable reset value
FEED_FWD	REAL	Disturbance input
OFF	REAL	Offset for P/PD operation
Y	REAL	Manipulated variable
ERR	REAL	System deviation
STATUS	Stat_COMP_PID	Output status
SP_CAS_N	REAL	Cascade reference variable
YMAN_N	REAL	Manually manipulated value
OFF_N	REAL	Offset for P/PD operation

Parameter description Mode_COMP_PID

Data structure description

Element	Data type	Meaning
r	BOOL	"1": Reset mode
man	BOOL	"1": Manual mode
halt	BOOL	"1": Halt mode
cascade	BOOL	"1": Cascade mode
en_p	BOOL	"1": P component in
en_i	BOOL	"1": I component in
en_d	BOOL	"1": D component
d_on_pv	BOOL	"1": D component on controlled variable "0": D component on system deviation
halt_aw	BOOL	"1": Antiwindup Halt "0": Antiwindup reset
bump	BOOL	"0": Bumpless operating mode switchover
ymanc	BOOL	"1": YMAN tracking

Parameter description Para_COMP_PID

Data structure description

Element	Data type	Meaning
gain	REAL	Proportional action coefficient (gain)
ti	TIME	Reset time
td	TIME	Rate time
td_lag	TIME	D component delay time
db	REAL	Dead zone
gain_red	REAL	Gain reduction in dead zone (db)
rate_sp	REAL	Setpoint velocity (SP) [1/s]
rate_man	REAL	Manually manipulated velocity value (YMAN) [1/s]
ymax	REAL	Upper threshold for Y
ymin	REAL	Lower threshold for Y
delt_aw	REAL	Limit expansion for antiwindup

Parameter description Stat_COMP_PID

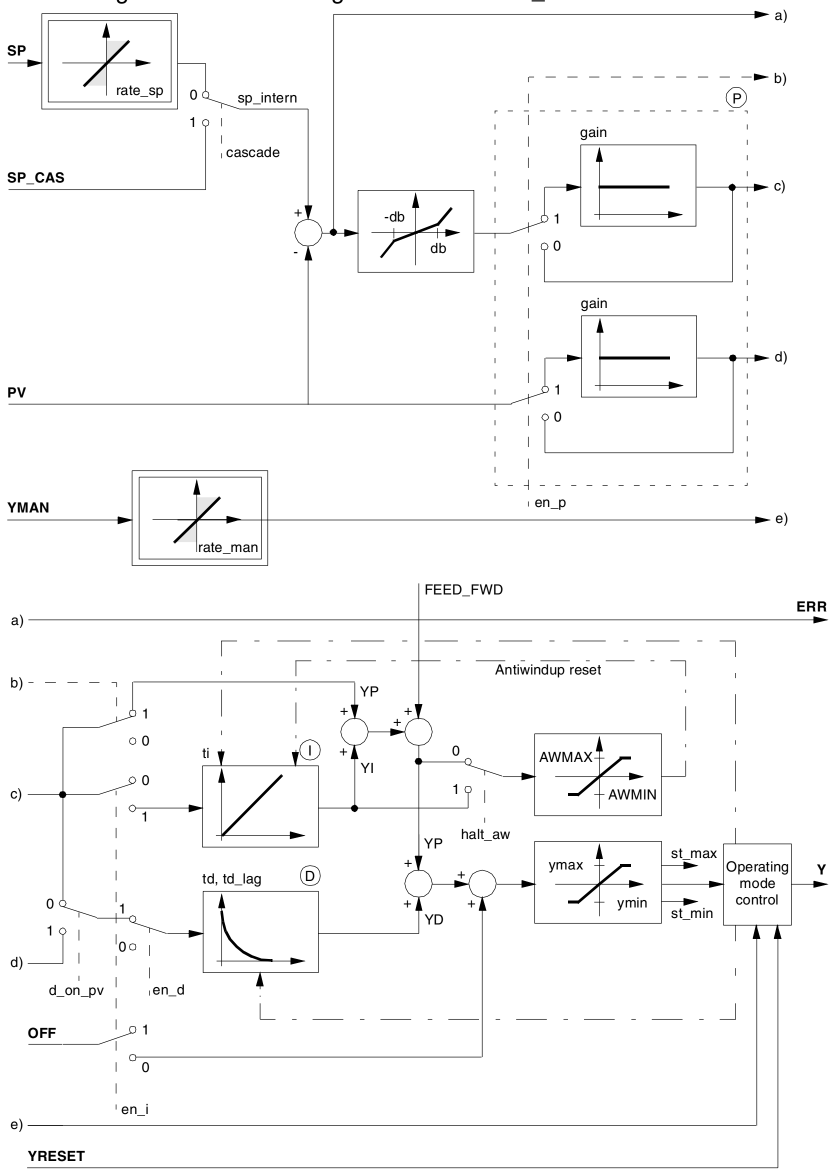
Data structure description

Element	Data type	Meaning
st_r	BOOL	"1": COMP_PID is in reset mode
st_man	BOOL	"1": COMP_PID is in manual mode
st_halt	BOOL	"1": COMP_PID is in halt mode
st_auto	BOOL	"1": COMP_PID is in automatic mode
st_cascade	BOOL	"1": COMP_PID is in cascade mode
st_max	BOOL	"1": Y ≥ Para_COMP_PID.ymax
st_min	BOOL	"1": Y ≤ Para_COMP_PID.ymin

Complex PID controller structure diagram

Structure diagram

The following is the structure diagram of the COMP_PID controller:



Parametering of the COMP_PID controller

Parametering

The COMP_PID control structure is displayed in the*Structure diagram, p. 80*.

The parametering of the function block is initially performed by the pure PID parameters, i.e. the proportional action coefficient gain, the reset time ti and the rate time td.

The D component is delayed by the time td_lag. The td/td_lag ratio is termed the differential gain, and is generally selected between 3 and 10. The D component can either be based upon the system deviation ERR (d_on_pv = "0") or the controlled variable PV (d_on_pv = "1"). Should the D component be determined by the controlled variable PV, then the D component will not be able to cause jumps when reference variable fluctuations (changes in input SP) take place. Generally, the D component only affects disturbances and process variances.

Note: The EFB has 3 I/O parameters (SP_CAS, OFF, YMAN) that are updated by the cascade mode function itself. To use the block in cascade mode, you have to establish the connection between these inputs and the appropriate outputs (SP_CAS_N, OFF_N, YMAN_N) through variables.

Control direction reversal

A reversed behavior of the controller can be achieved by reversing the sign of gain. Given a positive disturbance value, a positive/negative gain brings about a rise/fall of the manipulated variable. A negative value at gain causes the manipulated variable to drop when there is a positive deviation.

Forming the system deviation

In cascade mode, the ERR system deviation is formed by SP_CAS and PV:
- sp_intern = SP_CAS
- ERR = sp_intern - PV

The system deviation in automatic mode is formed by sp_intern and PV, whereby sp_intern is set to the value of parameter SP via a velocity limiter. The internal reference variable sp_intern is driven in ramp-type fashion toward the SP parameter value using the velocity specified in parameter rate_sp (unit 1/s).

The amount will be evaluated by parameter rate_sp. The function of the velocity limiter for SP is disabled if rate_sp = 0. SP is transferred directly to sp_intern.

System deviation is determined by the condition of parameter cascade when in reset, manual and halt modes.

If cascade = 1, sp_intern is set to the PV parameter value and ERR goes to 0.

If cascade = 0 and the setting is bumpless operation (bump = 0), sp_intern is set to the SP parameter value. Otherwise (bump = 1), sp_intern is also set to the PV parameter value.

Gain reduction for small system deviation values

Parameter db determines the size of a dead zone in which the proportional action coefficient gain is not effective, but rather a proportional action coefficient reduced by the parameter gain_red. The parameter db has an effect on the system deviation ERR = SP - PV in the form shown in the illustration *Representation of the dead zone, p. 83*. Unnecessary actuator loads caused by small controlled variable disturbances or measurement noise can be reduced by the dead zone.

Enter the db parameter as positive.

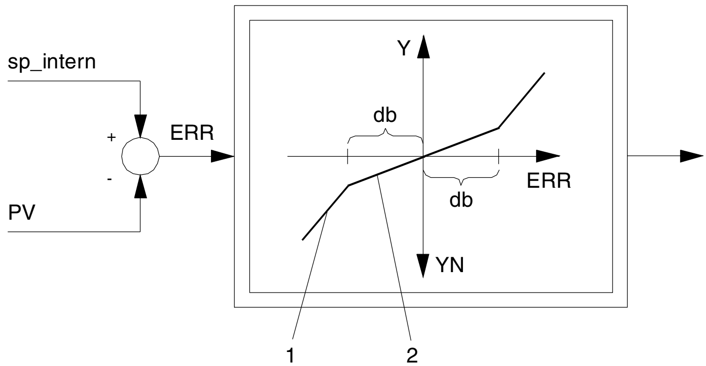
Enter values between 0 and 1 for gain_red.

Tracking of manual value YMAN

When manual tracking mode is enabled (ymanc = 1), the input YMAN is tracked to the manipulated variable value Y when in automatic and cascade modes, this means: YMAN = Y. If manual tracking mode is disabled (ymanc = 0), the YMAN value remains unchanged.

Representation of the dead zone

Dead zone:



1 Gradient 1
2 Gradient gain_red

Manipulated variable limiting

The limits ymax and ymin retain the manipulated variable within the prescribed range. Hence, ymin ≤ Y ≤ ymax. .

The elements qmax and qmin signal that the manipulated variable has reached a limit, and thus been capped:
- st_max = 1 if Y ≥ ymax
- st_min = 1 if Y ≤ ymin

For limiting the manipulated variable, the upper limit ymax should be greater than the lower limit ymin.

Antiwindup for COMP_PID

Definition

The antiwindup measure ensures that the I component does not grow too much causing the controller to lock if it has been limited at a control limit too long. Antiwindup measures are only performed for an active I component of the controller.

Limits for the antiwindup measure are by default the manipulated variables of the controller (delt_aw = 0). The parameter delt_aw can be used to either increase (delt_aw > 0) or decrease (delt_aw < 0) the limits with regard to the control limits (ymax, ymin).

Therefore, the limits used for the antiwindup measure are:
- AWMAX = ymax + delt_aw
- AWMIN = ymin - delt_aw.

Through displacement of the antiwindup limits in relation to the control limits (in particular with very noisy signals), the manipulated variable Y can be stopped from repeatedly 'jumping away' from the control limit (D component effect to disturbances) and subsequently returning to the limiting position (I component effect with system deviation ERR ≠ 0). If the control limits are to be simultaneously effective for the antiwindup measure, select the parameter delt_aw = 0.

By utilizing negative delt_aw values, antiwindup limits can be kept smaller than control limits (useful for antiwindup halt).

Antiwindup reset (halt_aw = 0)

Antiwindup measures disregard D component values to avoid being falsely triggered by D component peaks. The antiwindup-reset measure corrects the I component such that: AWMIN ≤ YP + FEED_FWD + YI ≤ AWMAX.

Antiwindup halt (halt_aw = 1)

The antiwindup measure only considers the I component. When antiwindup halt and I component are enabled, the antiwindup halt measure corrects the I component such that: AWMIN ≤ YP + FEED_FWD + YI ≤ AWMAX.

The parameters rate_sp and rate_man represent velocity limiters for the manual values SP and YMAN (see also function block VLIM). A 0 value disables the functionality of the corresponding velocity limiter (rate_sp = 0 or rate_man = 0, respectively). The SP and YMAN values are then utilized without delay.

Controller type selection for COMP_PID

Controller types

There are four different control types, which are selected via the parameters en_p, en_i and en_d.

Controller type	en_p	en_i	en_d
P controller	1	0	0
PI controller	1	1	0
PD controller	1	0	1
PID controller	1	1	1
I controller	0	1	0

The I-component can also be disabled with ti = 0.

The D contribution can also be disabled with td = 0.

OFF parameter influence

If the I contribution is enabled (en_i = 1), the manipulated variable Y is determined from the summation of the contributions YP, YI, YD, and FEED_FWD. Offset is not included in the calculation when the I contribution is enabled.

However, if the I component is disabled (EN_I = 0), the manipulated variable Y is formed from the summation of the components YP, YD, FEED_FWD, and the offset OFF.

Note: The OFF parameter is only designed for P, D, or PD controllers.

Bumpless operating mode switchover

Method of switching over

Bumpless on/off switching of the various components (P, I, D) is implemented.

Bumpless switching with enabled I component

If the P component is connected/disconnected, the internal I component will be corrected by the P component. This way, the connection/disconnection of the P contribution is bumpless even if the system deviation is not 0.

If the D component is disconnected, the internal I component takes over the remaining D component. If the D component is connected, it is set to 0.

Bumpless switching for disconnected D component

Bumpless switching for a disconnected D component is only implemented if parameter bump = 0. In this case, the OFF parameter is used to achieve the bumpless switchover.

If the P component is connected/disconnected, the value in the OFF parameter is corrected by the P component. This way, the connection/disconnection of the P component is bumpless even if the system deviation is not 0.

If the D component is disconnected, the remaining D component is added to the OFF parameter value. If the D component is connected, it is set to 0 (OFF remains unchanged).

Bumpless I component switching

Bumpless I component disconnection is only performed if parameter bump = 0. In this case, the OFF parameter as well as the internal I component (YI) are used to make the bumpless switchover possible.

Bumpless switchover from a PI(D) to a P(D) controller

The principle consideration for bumpless switching from a PI(D) to P(D) controller is based on the assumption that the PI(D) controller has reached a static condition. In this case, the process is in an idle state. The I component has a specific value in this case. To allow a bumpless switch to P(D) operation now, the I contribution of the PI(D) controller would have to serve as the PD controller operating point (offset), thus allowing the switch to take place without equalization processes (new transient condition) taking place. Based on the above consideration, bumpless I component disconnection is implemented in such a way that the OFF parameter retrieves its value.

Value of the manipulated variable Y depending on en_i:

If...	Then...
en_i = 1	Y = YP + YI + YD + FEED_FWD
en_i = 0	Y = YP + OFF + YD + FEED_FWD

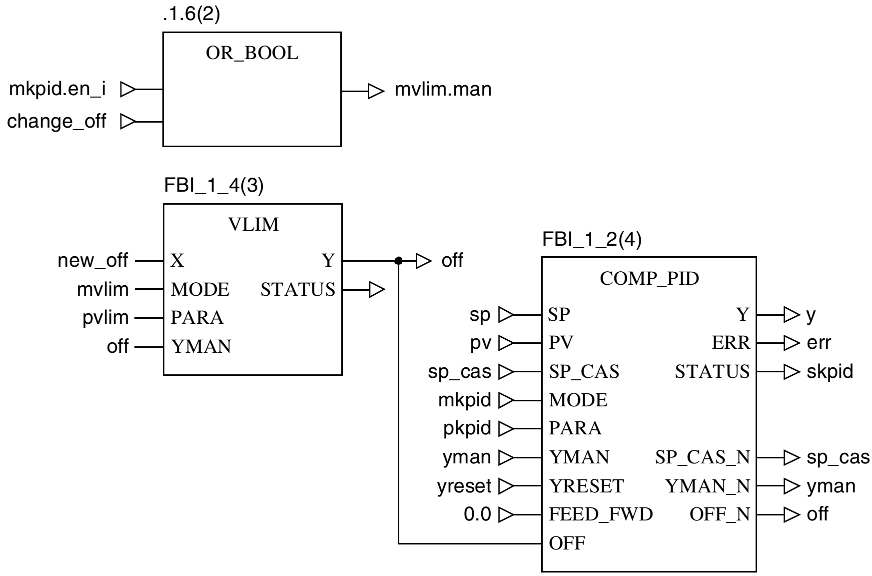
Starting up the I component

I component enabling is based on an analog consideration. The internal I component is set to the OFF parameter value. This allows the I component to be connected without giving rise to equalization processes.

Note: If the OFF parameter is calculated by a previous function block (EFB or DFB output, e.g. MOVE), the corrections for bumpless switching become ineffective (at the latest, when this function block is edited).

Example of a bumpless switchover of the D component

In order to achieve the bumpless P(D) controller switchover as well as OFF parameter modification by the user program, the following example can serve as a starting point.



In this example, the OFF parameter is set to the new_off variable value via a velocity limiter VLIM in ramp form using the velocity provided in pvlim.rate.

Note on the example

In this example, it is important to note the use of the OFF variable at the YMAN input of the VLIM as well as at the Y output of the VLIM, and the link of the output from VLIM to the OFF input of COMP_PID. The link between the Y output from VLIM and the OFF input from COMP_PID causes the VLIM function block to be processed prior to the COMP_PID function block (this is a prerequisite for proper operation). As long as the manual mode (mvlim.man = 1) is enabled in the VLIM, the manual value of the VLIM function block is transferred to the COMP_PID OFF parameter. The COMP_PID function block is now able to modify the content of the variable for bumpless handling. In the next cycle, this modified value is now available at the YMAN input of the VLIM function block. At an appropriate time, the manual mode in the VLIM function block can be disabled, and the function block drives up the value of the OFF variable from its current value to that of new_off. In the example above, manual mode enabling is controlled in the function block OR_BOOL. As long as COMP_PID has enabled the I component (mkpid.en_i = 1), the VLIM function block remains in manual mode.

Note: If mkpid.en_i = 1, the OFF parameter from COMP_ID will not be included in the calculation of the COMP_PID output.

In the above example, the OR_BOOL function block requires a second condition in order to change off to new_off: The variable change_off must be 1.

Bumpless alteration of gain

Modification of the proportional action coefficient gain is bumpless. As in the connection/disconnection of operating modes, this requires an internal correction to be carried out.

If the I component is enabled (en_i = 1 and ti > 0), the internal I component will be corrected by the expected P component jump which is caused by the gain modification.

If the I component is disconnected, the value in the OFF parameter will be corrected by the expected P component jump, provided the parameter bump = 0. If bump = 1, OFF is not modified and a P(D) controller gain variation leads to equalization processes.

Selecting the operating mode of the COMP_PID

Operating modes

There are five operating modes selectable through reset, man, halt, and cascade.

Operating mode	r	man	halt	cascade
Reset	1	1 or 0	1 or 0	1 or 0
Manual	0	1	1 or 0	1 or 0
Halt	0	0	1	1 or 0
Cascade	0	0	0	1
Automatic	0	0	0	0

Automatic and cascade modes

In automatic mode, the manipulated variable Y is determined through the discrete PID closed-loop control algorithm subject to controlled variable X and reference variable SP.

In cascade mode, the manipulated variable Y is determined through the discrete PID closed-loop control algorithm subject to controlled variable X and reference variable SP_CAS.

The distinction between these two operating modes, automatic and cascade, is only external in their different use of the reference variable SP. SP_CAS refers to cascade, SP to all other operating modes (with velocity limit). The SP_CAS variable is an input in cascade mode only, in all other modes it is an output. In SP_CAS, the X variable is returned to the master controller when in the modes reset, manual, halt or automatic as well as during startup, permitting bumpless switching from, for instance, fixed setpoint control to cascade control.

In both operating modes, the manipulated variable Y is limited by ymax and ymin. The control limits for the antiwindup measure can be extended using the parameter delt_aw.

Manual mode

In manual mode, the manual manipulated value YMAN is transferred to the manipulated variable Y with a velocity limiter. The manipulated variable Y is set to the YMAN parameter value in ramp form using the velocity (unit 1/s) rate set in the parameter rate_man.

The amount is evaluated by the parameter rate_man. +If rate_man = 0, the velocity limiter function for YMAN is disconnected. YMAN is transferred directly to the manipulated variable. The manipulated variable is limited by ymax and ymin.

Internal variables will be manipulated in such a manner that the controller changeover from manual to automatic (with I component enabled) can be bumpless. The antiwindup measure is designed just like in automatic mode.

In this operating mode the D component is automatically set to 0.

Reset mode

In Reset mode, the reset value YRESET is transferred directly to the manipulated variable Y. The manipulated variable is limited by ymax and ymin. Internal variables will be manipulated in such a manner that the controller changeover from manual to automatic (with I component enabled) can be bumpless. The antiwindup measure is performed just like in automatic mode.

Halt mode

In halt mode, the control output remains as is, i.e. the function block does not change the manipulated variable Y. Internal variables will be manipulated in such a manner that the controller can be driven smoothly from it's current position. Manipulated variable limits and antiwindup measures are as those in automatic mode. Halt mode is also useful in allowing an external operator device to adjust control output Y, whereby the controller's internal components are given the chance to continuously react to the external influence.

In this operating mode the D component is automatically set to 0.

Non-bumpless operation (bump = 0)

The definition of non-bumpless operation is when the controller exhibits a jump during operating mode switchover (e.g. manual to automatic) due to the P component in the manipulated variable Y. Depending on the controller's area of utilization, it might be useful for the controller to make a jump-type correction of the manipulated variable when switching over, for instance from manual to automatic, provided the system deviation is not equal to 0.

The jump height corresponds to the P component of the controller and is:

YP = ERR x gain

Bumpless operation (bump = 1)

The definition of bumpless operation is, the controller does not produce a discontinuity in the manipulated variable Y during an operating mode switchover. That is, it should continue at exactly the same location where it was positioned last. In this operating mode, the internal I component is corrected by the P contribution. If no I component is enabled, bumpless operation is achieved by tracing the operating point OFF such that the controller can continue during operating mode change without a bump in spite of system deviation being not equal to 0.

Detailed formulas

Explanation of formula variables

Meaning of the variables in the following formulas:

Variable	Meaning
dt	Time differential between the current cycle and the previous cycle
ERR	The current internally formed System deviation
$ERR_{(new)}$	System deviation value from the current sampling step
$ERR_{(old)}$	System deviation value from the previous sampling step
FEED_FWD	Disturbance (only in P, D or PD controllers)
OFF	Offset
$PV_{(new)}$	Value of controlled variable from the current sampling step
$PV_{(old)}$	Value of controlled variable from the previous sampling step
Y	current output (halt mode) or YMAN (manual mode)
YD	D component (only if en_d = 1)
YI	I component (only if en_i = 1)
YP	P component (only if en_p = 1)

Manipulated variable

The manipulated variable consists of various terms which are dependent on the operating mode:

$$Y = YP + YI + YD + OFF + FEED\_FWD$$

After summation of the components manipulated variable limiting takes place, so that:

$$ymin \le Y \le ymax$$

Overview of the calculation of the control components

The following is an overview on the different calculations of the control components in relation to the elements en_-, en_I and en_d:
- P component YP for manual, halt, automatic and cascade modes
- I component YI for automatic mode
- I component YI for manual and halt modes
- D component YD for automatic and cascade mode
- D component YD for manual and halt modes

P component YP for all operating mode

YP for manual, halt, automatic and cascade modes is determined as follows:

For en_p = 1 the following applies:

$$YP = gain \times ERR$$

For en_p = 0 the following applies:

$$YP = 0$$

I component YI for automatic mode

YI for automatic mode is determined as follows:

For en_i = 1 the following applies:

$$YI_{(new)} = YI_{(old)} + gain \times \frac{dt}{ti} \times \frac{ERR_{(new)} + ERR_{(old)}}{2}$$

For en_i = 0 the following applies:

$$YI = 0$$

The I component is formed according to the trapezoid rule.

I component YI for manual and halt modes

YI for manual, halt and automatic modes is determined as follows:

For en_i = 1 the following applies:

$$YI = Y - (YP - FEED\_FWD)$$

For en_i = 0 the following applies:

$$YI = 0$$

D component YD for automatic and cascade mode

YD for automatic mode and cascade is determined as follows:

For en_d = 1 and d_on_pv = 0 the following applies:

$$YD_{(new)} = \frac{YD_{(old)} \times td\_lag + td \times gain \times (ERR_{(new)} - ERR_{(old)})}{dt + dt\_lag}$$

For en_d = 1 and d_on_pv = 1 the following applies:

$$YD_{(new)} = \frac{YD_{(old)} \times td\_lag + td \times gain \times (PV_{(old)} - PV_{(new)})}{dt + dt\_lag}$$

For en_d = 0 the following applies:

$$YD = 0$$

D component YD for manual and halt modes

YD for manual, halt and automatic modes is determined as follows:

$$YD = 0$$

Runtime error

Error message

An Error message appears, if

- an unauthorized floating point number is placed at the input PV
- gain_red > 1 or gain_red < 0 is
- db < 0 is
- or ymax < is ymin

DEADTIME: Deadtime device

7

Overview

At a glance

This chapter describes the DEADTIME block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

With this function block an input signal is delayed by a time, the so-called deadtime.

The function block delays the signal X by the deadtime T_DELAY before it appears again at Y.

The function block utilizes a 128 element delay buffer to hold a sequence of X values, i.e. during the T_DELAY time 128 discrete X values are detained. The buffer is used in such a way that it corresponds with the operating mode.

The value of Output Y remains unchanged after cold and warm system starts. The internal values are set to the value of X.

After a change of deadtime T_DELAY or a cold or warm system start, the output READY goes to "0". This means: that the buffer is empty and not ready.

The function block has the following operating mode:
- Manual
- Halt
- Automatic.

EN and ENO can be projected as additional parameters.

> **Note:** The delay time continues to run even if the block is disabled via the EN parameter, because the block calculates its time differences according to the system clock.

Formula

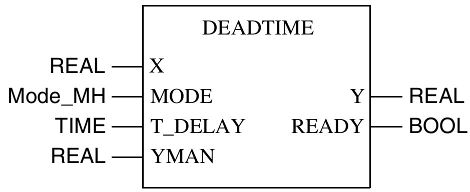
The transfer function is:

$$G(s) = e^{-s \times T\_DELAY}$$

Representation

Symbol

Representation of the block



DEADTIME parameter description

Block parameter description

Parameter	Data type	Meaning
X	REAL	Input value
MODE	Mode_MH	Operating mode
T_DELAY	TIME	Deadtime
YMAN	REAL	Manual manipulated value
Y	REAL	Output
READY	BOOL	"1" = internal buffer is full "0" = internal buffer is not full (e.g. after warm/cold start or modification of deadtime)

Parameter description Mode_MH

Data structure description

Element	Data type	Meaning
man	BOOL	"1" = Manual mode
halt	BOOL	"1" =Halt mode

Operating mode

Selecting the operating modes

There are three operating modes, which are available via the man and halt parameter inputs:

Operating mode	man	halt
Automatic	0	0
Manual	1	0 or 1
Halt	0	1

Automatic operating mode

In the automatic mode, the function block operates according to the following rules:

If…	Then…
Scan time > $\frac{T\_Delay}{128}$	the current X value is transferred to the buffer, and the oldest X value in the buffer is placed on the output Y. If the scan time is more than T_DELAY / 128, resolution is less than 128 causing a systematic error, i.e. some X-values are double-stored (see the following Example).
Scan time < $\frac{T\_Delay}{128}$	not all X values can be stored in the buffer. In this case the X value is not saved in some cycles. After completion of T_DELAY, output Y may correspondingly remain unchanged in two (or more) consecutive cycles.

Example of automatic mode

In the example the following values are accepted:

Cycle time = 100 ms

T_DELAY = 10 s

tin = T_DELAY / 128 = 78 ms

As the reading time tin is shorter than the cycle time, each X value is transferred to the buffer. On the fourth execution of the function block (after 400 ms) the X value is saved twice rather than once (as 3 x 78 = 312 and 4 x 78 = 390).

Manual mode

In manual mode the manual value YMAN is consistently transferred to the control output Y. The internal buffer is charged with the manual value YMAN. The buffer is marked as charged (READY =1).

Halt mode

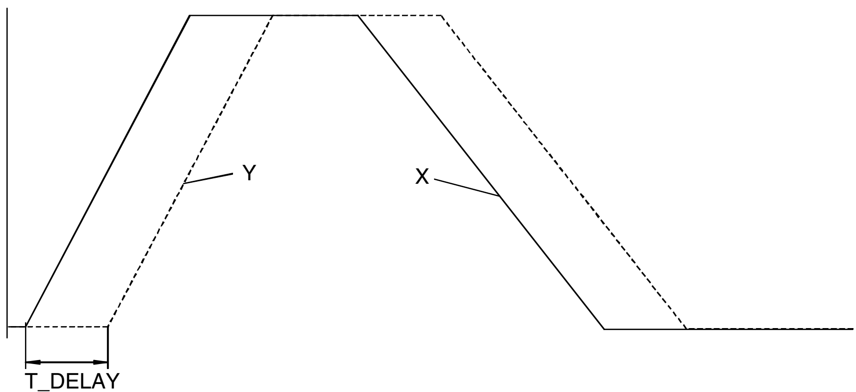
The output Y is held at the last calculated value in Halt mode. The output will no longer be changed, but can be overwritten by the user. The internal buffer still continues to operate as in automatic mode.

Example for behavior of the function block

Example

The following diagram shows an example for behavior of the function block. Input X follows a ramp function from one value to a new value. Delayed by the deadtime T delay, X values appear at Y.

DEADTIME function block diagram



Runtime error

Error message

An Error message, appears when an invalid floating point number lies at input YMAN or X.

DELAY: Deadtime device

8

Overview

At a glance

This chapter describes the DELAY block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

With this function block the input signal is delayed by a deadtime.

The function block delays the signal X by the deadtime T_DELAY before it appears again at Y.

The function block incorporates a delay buffer for 128 elements (X-values), meaning that during the time span T_DELAY 128 X-values can be stored. The buffer is used in accordance with the various operating mode.

The value of Output Y remains unchanged after cold and warm system starts. The internal values are set to the value of X.

After a change of deadtime T_DELAY or a cold or warm system start, the output READY goes to "0". This means: that the buffer is not ready because it is empty.

The function block has the following operating mode: Manual, halt and automatic mode.

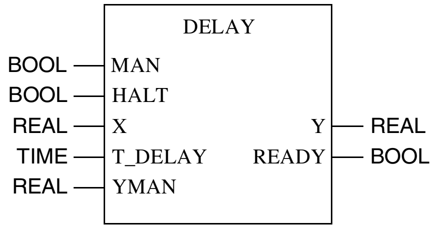
EN and ENO can be projected as additional parameters.

Note: The delay time continues to run even if the block is disabled via the EN parameter, because the block calculates its time differences according to the system clock.

Representation

Symbol

Representation of the block



Parameter description

Block parameter description

Parameter	Data type	Meaning
MAN	BOOL	"1" = Manual mode
HALT	BOOL	"1" =Halt operating mode
X	REAL	Input value
T_DELAY	TIME	Deadtime
YMAN	REAL	Manual manipulated value
Y	REAL	Output
READY	BOOL	"1" = internal buffer is full "0" = internal buffer is not full (e.g. after warm/cold start or modification of deadtime)

Operating mode

Selecting the operating modes

There are three operating modes, which are selected via the inputs MAN and HALT.

Operating mode	MAN	HALT
Automatic	0	0
Manual	1	0 or 1
Halt	0	1

Automatic operating mode

In the automatic mode, the function block operates according to the following rules:

If	Then
Scan time > $\frac{T\_Delay}{128}$	the current X value is transferred to the buffer, and the oldest X value in the buffer is placed on the output Y. If a cycle time is greater than T_DELAY / a resolution of less than 128 will result, causing a systematic error leading to double storage of some X values. (see the following Example).
Scan time < $\frac{T\_Delay}{128}$	not all X values can be stored in the buffer. In this case the X value is not saved in some cycles, and Y remains unchanged in these cycles.

Example of automatic mode

In the example the following values are accepted:

Cycle time = 100 ms

T_DELAY = 10 s

tin = T_DELAY / 128 = 78 ms

As the reading time tin is shorter than the cycle time, each X value is transferred to the buffer. On the fourth execution of the function block (after 400 ms) the X value is saved twice rather than once (as 3 x 78 = 312 and 4 x 78 = 390).

Manual mode

In manual mode the manual value YMAN is consistently transferred to the control output Y. The internal buffer is charged with the manual value YMAN. The buffer is marked as charged (READY =1).

Halt mode

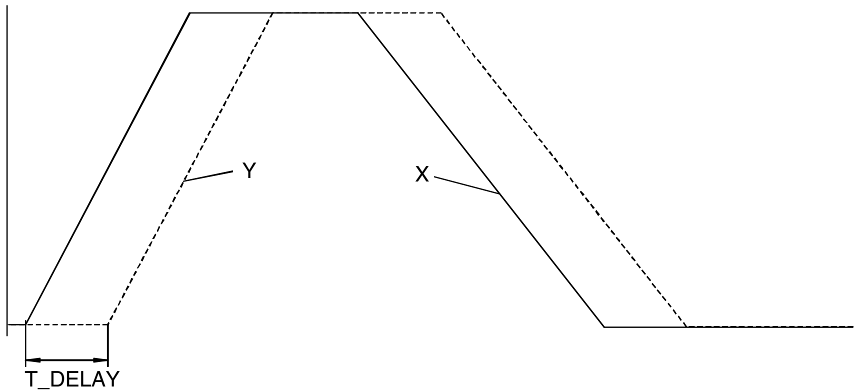
The output Y is held at the last calculated value in Halt mode. The output will no longer be changed, but can be overwritten by the user. The internal buffer still continues to operate as in automatic mode.

Example of the behavior of the function block

Example

The following diagram shows an example of the behavior of the function block. Input X follows a ramp function from one value to a new value. Delayed by the Deadtime T delay, X values appear at Y.

Diagram of the DELAY function block



DERIV: Differentiator with smoothing

9

Overview

At a glance

This chapter describes the DERIV block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block is a differential element with a delayed output Y respecting the delay time constant lag.

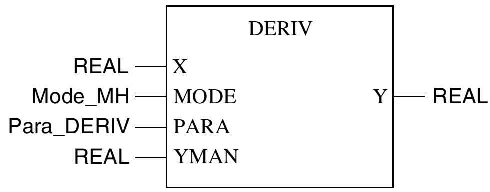
The function block contains the following operating mode: Manual, halt and automatic mode.

EN and ENO can be projected as additional parameters.

Representation

Symbol

Representation of the block



Parameter description DERIV

Block parameter description

Parameter	Data type	Meaning
X	REAL	Input variable
MODE	Mode_MH	Operating Modes
PARA	Para_DERIV	Parameter
YMAN	REAL	Manual manipulated value
Y	REAL	Output derivative unit with smoothing

Parameter description Mode_MH

Data structure description

Element	Data type	Meaning
man	BOOL	"1" = Manual mode
halt	BOOL	"1" =Halt operating mode

Parameter description Para_DERIV

Data structure description

Element	Data type	Meaning
gain	REAL	Gain of the differentiation
lag	TIME	Delayed time constants

Formulas

Transmission function

The transfer function for Y is:

$$G(s) = gain \times \frac{s \times lag}{1 + s \times lag}$$

Calculation formula for Y

The calculation formula for Y is:

$$Y = \frac{lag}{dt + lag} \times (Y_{(old)} + gain \times (X_{(new)} - X_{(old)}))$$

Special case: lag = 0

This amounts to pure differentiation without a 1st order time limiter.

In this situation the transfer function is:

$$G(s) = gain \times s$$

The formula of calculation is:

$$Y = gain \times \frac{X_{(new)} - X_{(old)}}{dt}$$

Meaning of the sizes

The meaning of the formula sizes is asfollows:

size	Meaning
$X_{(new)}$	the input X value for the current cycle
$X_{(old)}$	the input X value from the previous cycle
$Y_{(old)}$	the output Y value from the previous cycle
dt	is the time differential between the current cycle and the previous cycle

Detailed description

Parametering

The parameter assignments of the function block are effected by the determination of gain, the differentiator and the time constant lag, by which the output Y is delayed.

For very short sampling times and an input X unit step (input X jumps from 0 to 1.0), the output Y jumps to the value gain (in theory _ in reality somewhat smaller, due to the sampling time not being infinitely small), and then returns to 0 with the delay time constant lag.

Operating mode

There are three operating modes selectable via the man and halt parameter inputs:

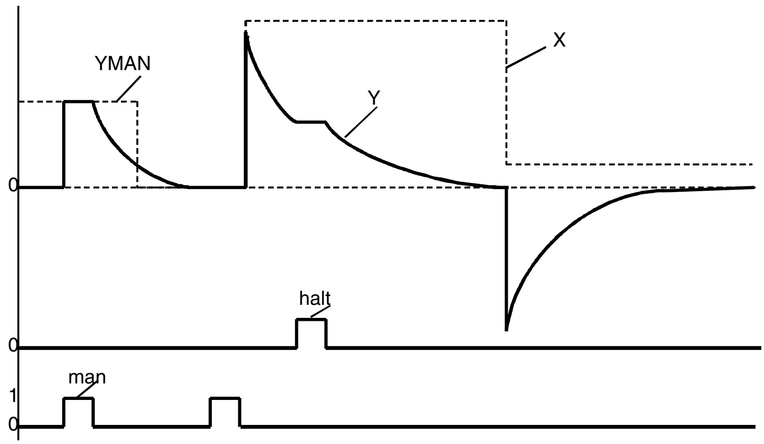
Operating mode	man	halt	Meaning
Automatic	0	0	The function block operates as described in "Parametering".
Manual	1	0 or 1	The input YMAN will be transferred directly to the output Y.
Halt	0	1	The output Y will be held at the last calculated value. The output remains at this value, but can still be overwritten by the user.

Example for the function block

DERIV example

The following example shows the step response of the DERIV function block.

Jump response with gain = 1 and lag = 10 s



Runtime error

Error message

An Error message, appears when an invalid floating point number lies at input YMAN or X.

DTIME: Delay

10

Overview

At a glance

This chapter describes the DTIME block.

What's in this Chapter?

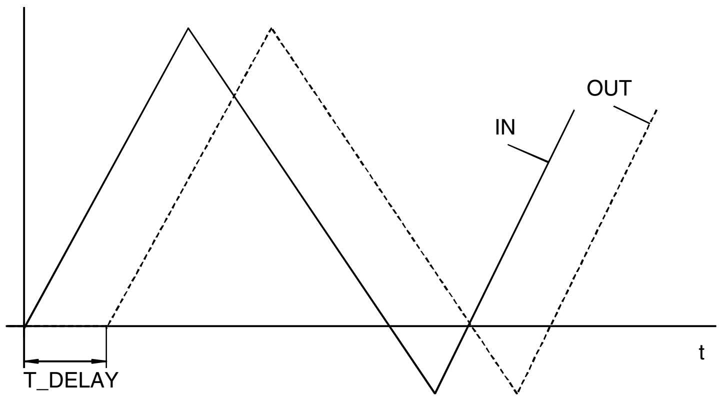
This chapter contains the following topics:

Brief description

Function description

The function blockDTIME generates a delay when numerical input variables are transferred. The numerical output variable OUT generates the same behavior as the numerical input variable when the delay T_DELAY, which can vary, is included.

Behavior of the DTIME function block:



EN and ENO can be projected as additional parameters.

Formula

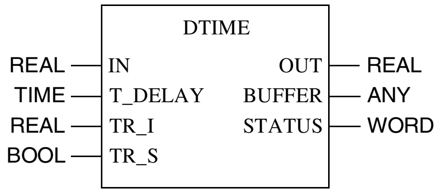
This function block implements the following transfer function :

$G_{(p)} = e^{-p.T\_DELAY}$

Representation

Symbol

Representation of the block



Parameter description

Block parameter description

Parameter	Data type	Meaning
IN	REAL	Digital value to be delayed
T_DELAY	TIME	Desired delay
TR_I	REAL	Initialization input
TR_S	BOOL	Initialization command
OUT	REAL	Delayed output
BUFFER	ANY*)	Memory for the purpose of storing delayed values.
STATUS	WORD	Status word

*) It is essential for this to be linked to a variable (see"*Parametering, p. 116*").

Parametering

Saving the input values (BUFFER output)

The BUFFER output must be linked to a variable (generally of the Buffer_DTIME) type. The values to be delayed are contained in these variables. Each time the function block is executed a new value is saved for the IN input.

The size of the variable linked to the BUFFER output determines the number of values which can be saved and therefore also the allowable maximum delay value.

$$T\_DELAY_{maximum} = n \times T\_Period$$

The following applies here

Formula size	Meaning
n	Number of real values which the BUFFER can contain.
T_PERIOD	Sampling interval of the function block

> **Note:** As soon as a variable has been linked to the BUFFER output, it can only be replaced by a variable of the same type. To replace it with a greater variable, which would enable a higher delay value to be reached for example, the function block must be deleted and a new one put in place.

Data type of the buffer output

The BUFFER output is of the ANY type. This means any variable type can be assigned to it. It is generally an advantage to use a variable of the Buffer_DTIME type at first. This also involves a table containing up to 100 real values. With this variable type it is possible to attain a delay which corresponds to 100 times the sampling interval of the DTIME function block.

Procedure for large delay times

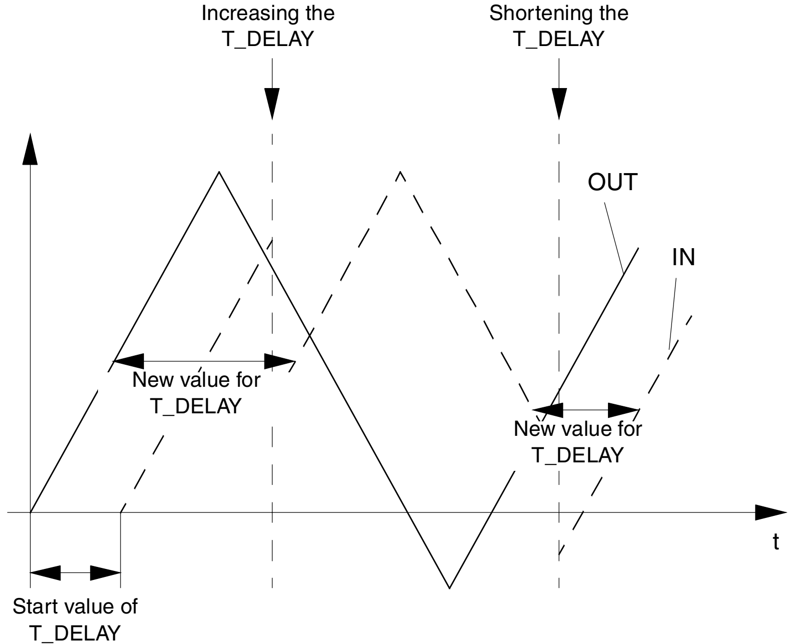
To attain delay values which are equivalent to over 100 times the sampling interval of the function block, a larger variable must be assigned to the BUFFER parameter:

Step	Action
1	Define a new derived data type, e.g. a table with 200 floating point values
2	Declare a variable of this type and link it to the BUFFER parameter of the DTIME function block.
3	In this case the maximum delay corresponds to 200 times the sampling interval of the function block

Dynamic modification of the T_DELAY delay

It is possible to raise or lower the T_DELAY delay time while the program is running. As long as the re-adjusted delay time is compatible with the size of the BUFFER output, the new delay is effective immediately.

Presentation of the dynamic modification of the T_DELAY delay



If the T_DELAY value is too great in relation to the BUFFER size, it is no longer possible to save enough input values to attain the delay desired. In this case the delay remains at the longest time possible (bit 8 of the status word then goes to 1 over).

To prevent this problem it is advisable to define the dimensions of the variable assigned to the BUFFER parameter so that a possible increase in the T_DELAY can be provided for.

When T_DELAY = 0, the OUT output always corresponds to the IN input.

Initialization and operating mode

Initialization and operating mode

The first time the function block is executed (when loading the program or during online calls), all the values contained in the buffer are initialized with the value of TR_I. The OUT output retains this value for the duration of the T_DELAY. If the TR_I input is not attached, the value 0 serves to initialize the BUFFER output and the OUT output retains the value 0 during the T_DELAY.

In the tracking operating mode (TR_S = 1), the input TR_I is transferred to the OUT output and the BUFFER output is also initialized with the value of TR_I. After returning to normal operating mode, the output retains this value for the duration of T_DELAY, as was the case with the first cycle.

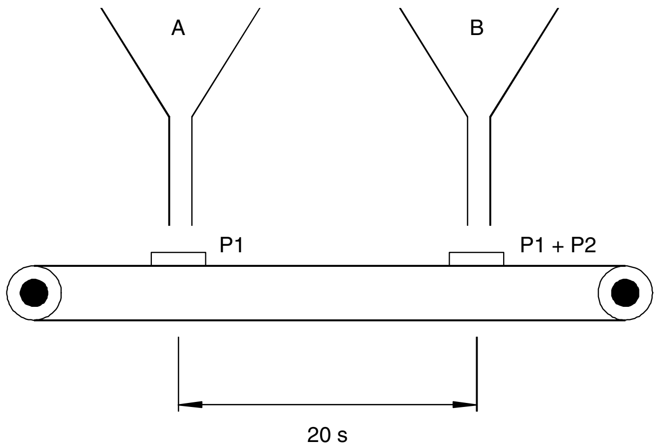
Example for measuring a rate of flow

Measuring a rate of flow

The DTIME function block can be used for example to model a process delay, whose uses include a design to measure flow rates or the number of revolutions of propulsion systems.

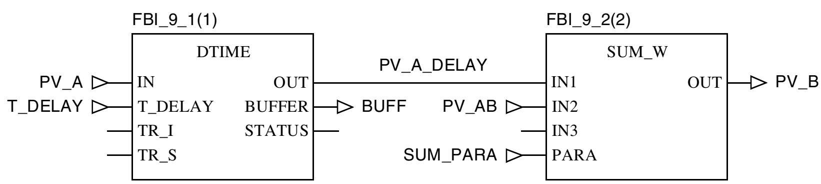
In the following example two products, A and B, are poured into a container one after the other and mixed. First, the container is placed under the dosing device for product A, to give the amount P1. Then it is moved on a conveyor belt to the dosing device for product B to give the amount P2. The time interval between the two dosing devices is 20 s.

Measuring flow rates



The product amount P2 is regulated, but the weight in the container is P1+P2. P1 should be removed. The amount P2 corresponds to the amount measured minus the amount P1 dosed 20 s beforehand.

Measuring the servo loop at P2 corresponds to the following illustration:



Values of the data structure elements of the SUM_PARA variables:

Element of SUM_PARA	value
SUM_PARA.K1	1
SUM_PARA.K2	1

Runtime error

Status word

In the status word the following messages are displayed:

Bit	Meaning
Bit 0 = 1	Error in a calculation with floating point values
Bit 1 = 1	Invalid value recorded at one of the floating point value inputs
Bit 2 = 1	Division by zero with calculation in floating point values
Bit 3 = 1	Capacity overflow with calculation in floating point values
Bit 8 = 1	T_DELAY exceeds the maximum value that can be represented on the BUFFER output.

Error message

This error appears if a non floating point value is inputted or if there is a problem with a floating point calculation. In this case the outputs OUT and BUFFER remain unchanged.

Alert

There will be an alert if a T_DELAY exceeds the maximum possible value. In this case the function block uses the maximum value. If an outgoing value is required, which is above the default value, only the BUFFER output needs to be linked to a larger variable.

FGEN: Function generator

11

Overview

At a glance

This chapter describes the FGEN block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

TheFunction block FGEN represents a function generator. It generates a signal form at output Y which is defined in the data structure Para_FGEN. The function block can be cascaded, i.e. if several of these EFBs are used, various signal forms can be created and laid over one another.

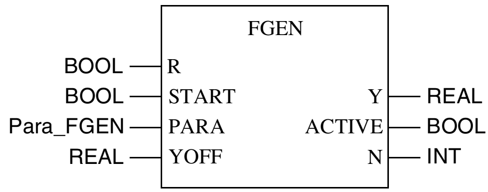
The following 8 different signal forms can be generated:
- Jump function
- Ramp function
- Delta function
- Saw-tooth function
- Square wave function
- Trapezoid function
- Sine function
- Random Number

As additional parameters, EN and ENO can be projected.

Representation

Symbol

Block representation



Parameter description FGEN

Block parameter description

Parameter	Data type	Meaning
R	BOOL	"1": Reset
START	BOOL	1": Start function generator
PARA	Para_FGEN	Parameter
YOFF	REAL	Output Y offset
Y	REAL	Function generator output
ACTIVE	BOOL	ACTIVE = 1: Function generator is active
N	INT	Number of intervals since start

Parameter description Para_FGEN

Data structure description

Element	Data type	Meaning
func_no	INT	Generator function choice (1-8)
amplitude	REAL	Function amplitude
halfperiod	TIME	Half cycle duration
t_off	TIME	Idle time constant
t_rise	TIME	Rise time constant
t_acc	TIME	Smoothing time
unipolar	BOOL	"1 "= Signal unipolar "0 "= Signal bipolar

Parametering

Reset

Parameter R stands for RESET. If this parameter is set (R = 1), all running functions will be immediately terminated and output Y goes to the value of parameter YOFF (offset). Simultaneously the cycle counter N is also reset to 0 and ACTIVE returns to "0".

Starting the function generator

The parameter START (START = 1) starts the function defined with the data structure. Output N is incremented with the beginning of each new cycle. If the parameter START returns to "0", the active cycle of the selected function runs to completion. As long as a function runs, the output ACTIVE is 1. If the period ends the output ACTIVE is reset to 0.

Offset

Waveforms produced by the function generator have an amplitude with the value of parameter "amplitude", i.e. values range from "amplitude" to -"amplitude" for bipolar operation (unipolar = "0") resp. from 0 to "amplitude" in unipolar operation (unipolar = "1"). Waveform values can be shifted away from the zero reference point through the parameter YOFF.

Note: Should the output of another function generator be applied to parameter YOFF, the waveforms produced by both function generators are overlaid.

Rise time t_rise

Rise time t_rise is used only by the functions "ramp" and "trapezoid". In the "saw-tooth" function rise time is determined by halfperiod - t_off. Rise time is 0.5 * (halfperiod - t_off) for the "delta" function.

Function selection

Selection

There are a total of 8 functions which can be produced by the function generator. Function selection is made through func_no. At a function change the last selected running function still proceeds to completion.

The following function numbers are allowed:

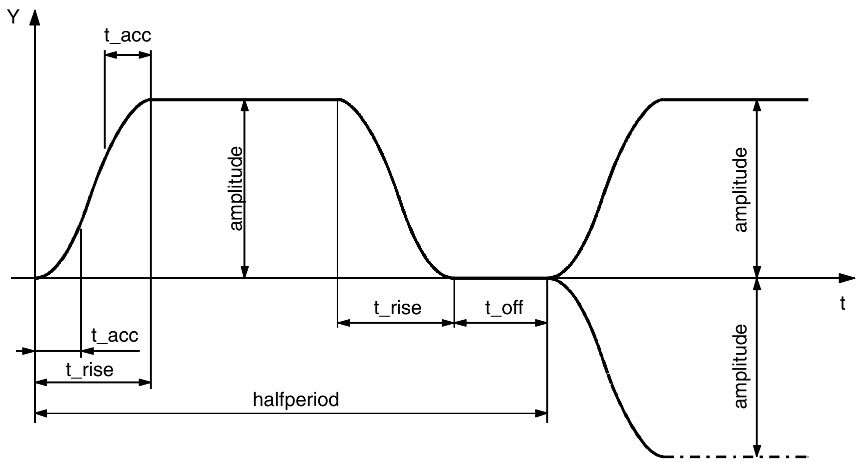
func_no	Function
1	Jump
2	Ramp
3	Saw-tooth
4	Delta
5	Square
6	Trapezoid
7	Sine
8	Random Number

Function definition

Definition

The function is defined completely in the data structure Para_FGEN. First of all the waveform must be determined (refer to *Function selection, p. 125*).

Trapezoid (Delta, Saw-tooth, Square) unipolar/bipolar is selected as the basic type for the definition.



Function amplitude is determined in the parameter amplitude. It should be noted that this declaration applies to unipolar operation. Amplitude in bipolar operation is doubled and consists of amplitude and -amplitude.

The parameter halfperiod defines the half cycle duration.

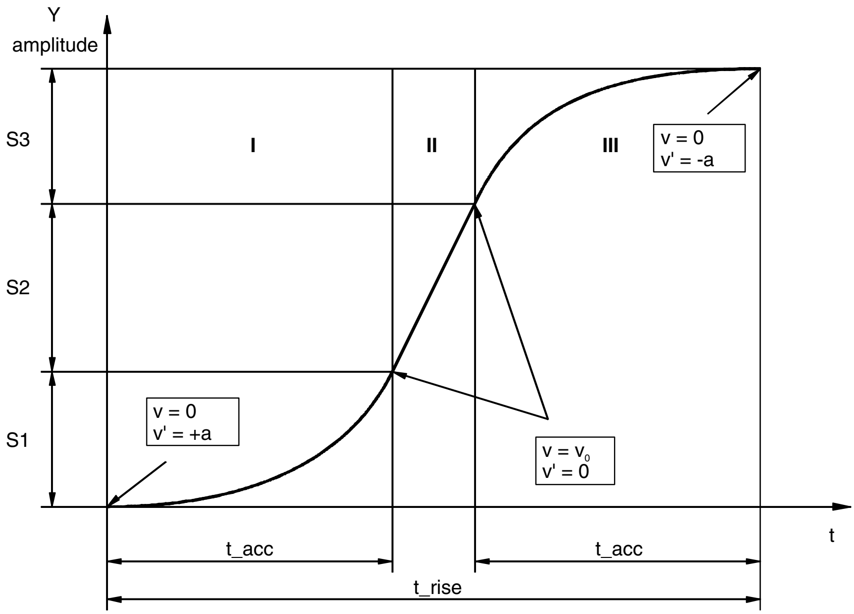
Parameter t_off defines an idle time. A half cycle of the function is then output within the time halfperiod - t_off.

For the trapezoid function definition the rise time t_rise is also required. This is the time in which the signal should accelerate from 0 to amplitude. This time is also taken for the descent from amplitude back to 0.

"Smoothing" a function

If a function in ramp form is to rise or decline, the transitions are first of all always made in a sharp crease. The gradient is not constant in this case. "Smoothing" is used to achieve a soft rise and descent, i.e. the ramp turns into an S-curve.

"Smoothing" a function



This is then divided into three sections. Section I "accelerates" directly from 0. Section II is traversed with the velocity attained at the end of section I. In section III, the acceleration from section I is used to brake, and thus approach the terminal point softly. The size of the section is user-definable. They are defined by specifying t_acc and t_rise.

The acceleration involved is calculated by the following formulas:

$$\text{amplitude} = S1 + S2 + S3$$

with

$$S3 = S1 = \frac{a}{2} \times t\_acc^2$$

and

$$S2 = a \times t\_acc \times (t\_rise - 2 \times t\_acc)$$

It then follows that:

$$a = \frac{amplitude}{t\_acc \times t\_rise - t\_acc^2}$$

Note: Smoothing is used only by the functions "Ramp", "Saw-Tooth", "Delta" and "trapezoid". "Jump", "Square" and "Sine" are not "smoothable" functions.

Individual Parameter Usage

Parameter use within the various functions.

Function	amplitude	halfperiod	t_off	t_rise	t_acc	uni-polar
Jump	X					
Ramp	X			X	X	
Saw-tooth	X	X	X	halfperiod - t_acc	X	X
Delta	X	X	X	(halfperiod - t_acc)/2	X	X
Square	X	X	X			X
Trapezoid	X	X	X	X	X	X
Sine	X	X	X			X
Random number	X					X

Function diagrams can be found in the section *Diagrams of the individual functions, p. 129*.

Unipolar operation

The unipolar parameter defines whether the selected function should be output as a unipolar or bipolar function. Particular attention should be paid to the fact that in unipolar operation a cycle is still characterized by 2 "unipolar" half waves.

Altering function parameters

During a currently executing cycle, all function parameters may be altered. However, any alterations made will not take effect until the cycle has completed. Should, for example, the idle time t_off be altered during the running cycle, it does not apply until the start of the next cycle.

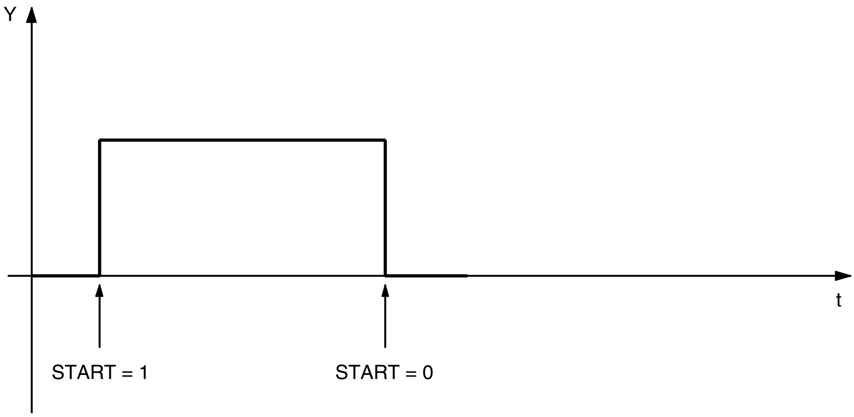
Altering a function

If the parameter func_no is changed during a currently executing cycle, it will also not take effect until the cycle has completed with the previously selected function. The new function is then started. This resets the cycle counter N, which indicates the period number, to 0.

Diagrams of the individual functions

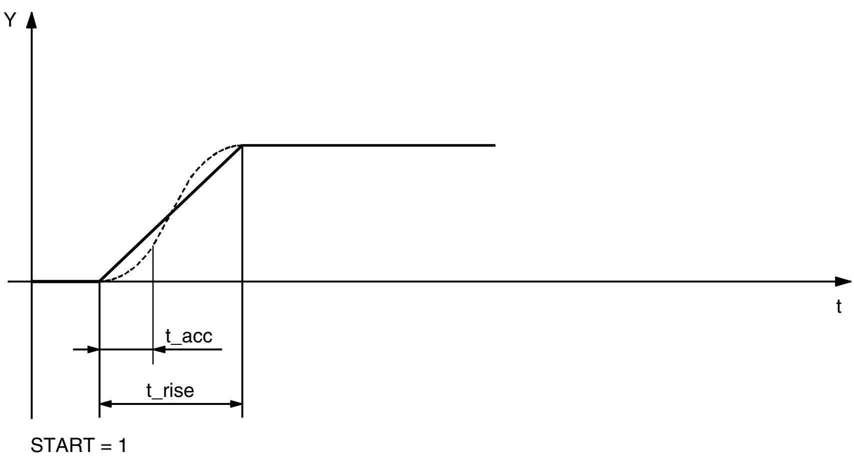
Jump function

Representation of the Jump function



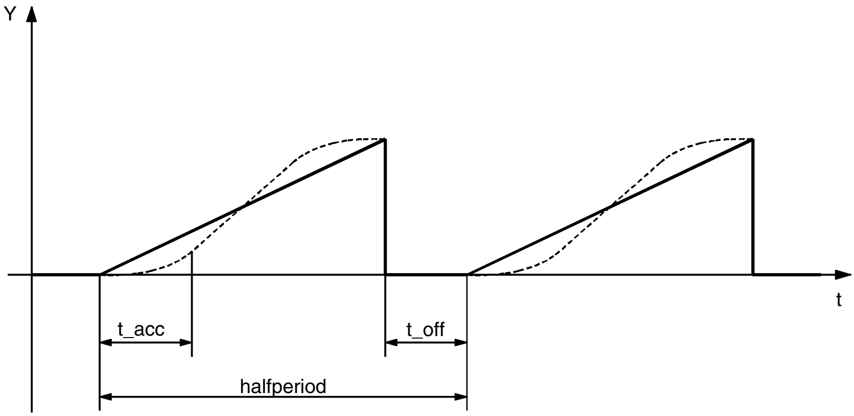
Ramp function

Representation of the Ramp function



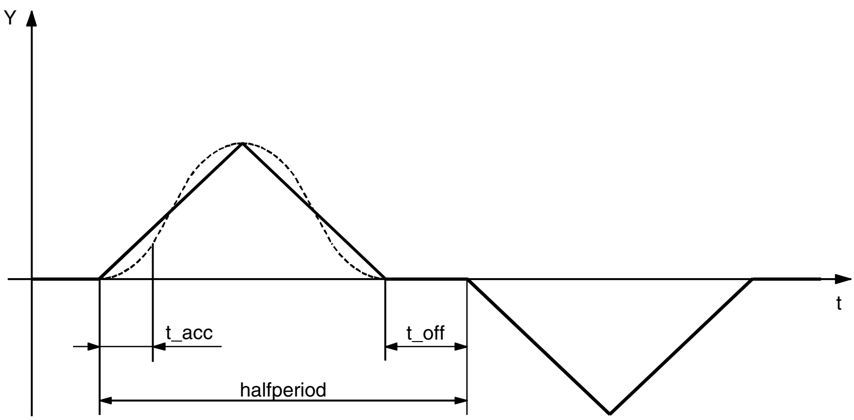
Saw-tooth function

Representation of the Saw-tooth function



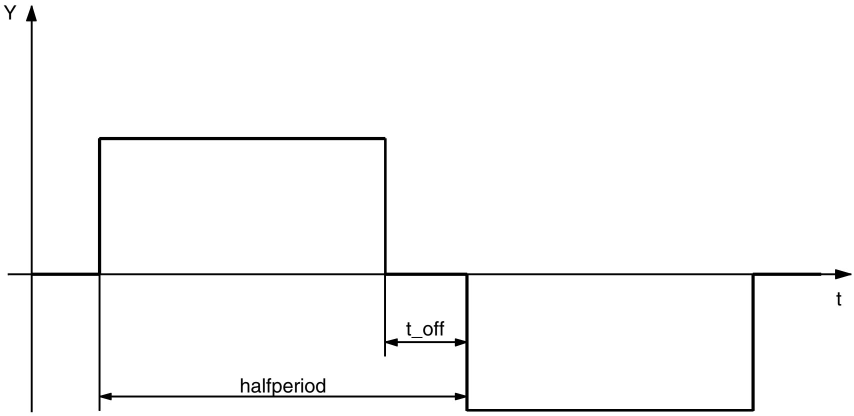
Delta function

Representation of the Delta function



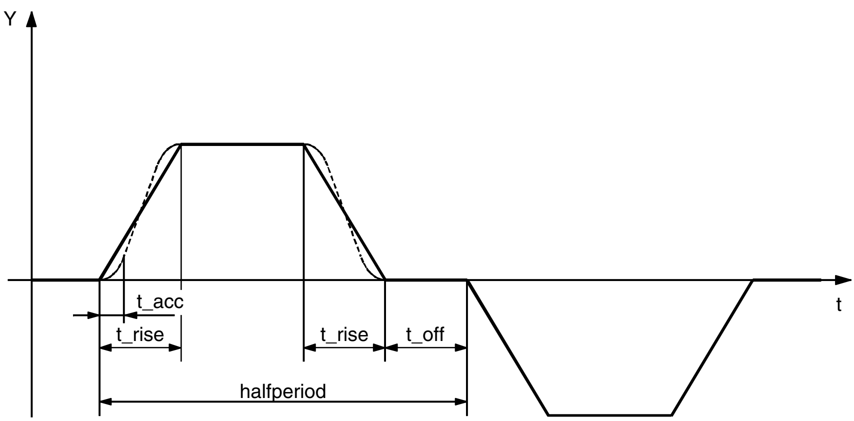
Square wave function

Representation of the Square wave function



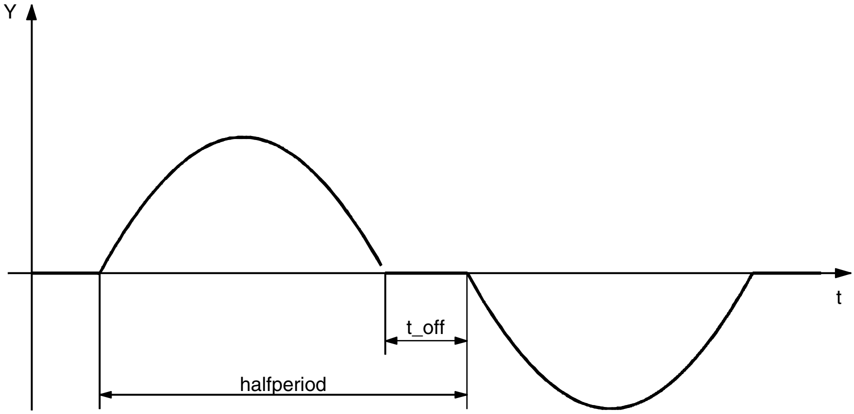
Trapezoid function

Representation of the Trapezoid function



Sine function

Representation of the Sine function



Special cases

Jump function

On the "Jump" function the output goes to

the value Y = OFF if START = 0

and

the value Y = OFF + amplitude if START = 1

set

The time specifications (t_off, t_rise, t_acc) do not play a role in this function.

Output N is incremented for every new 0 →1 transition of input START.

There is no bipolar mode for this function, i.e. the unipolar parameter value is disregarded.

Ramp function

In the "Ramp" function output Y ramps upward from value YOFF to YOFF + amplitude. While START is unchanged at 1, output Y remains at the value YOFF + amplitude. Output Y jumps back to value YOFF should START be taken back to 0.

Run up is determined by the times t_rise and t_acc. The time needed for run up from Y = YOFF to Y = YOFF + amplitude is specified by t_rise. "Smoothing" can be influenced by t_acc.

Output N is incremented for every new 0 →1 transition of input START.

There is no bipolar mode for this function, i.e. the unipolar parameter value is disregarded.

Random number

In the "Random number" function output Y is set to a number resulting "by chance" between

YOFF ≤ Y ≤ YOFF + amplitude, in unipolar operation

and

YOFF - amplitude ≤ Y ≤ YOFF + amplitude, when the operation is bipolar

.

The time specifications (t_off, t_rise, t_acc) do not play a role in this function.

Output N is incremented for every new 0 →1 transition of input START.

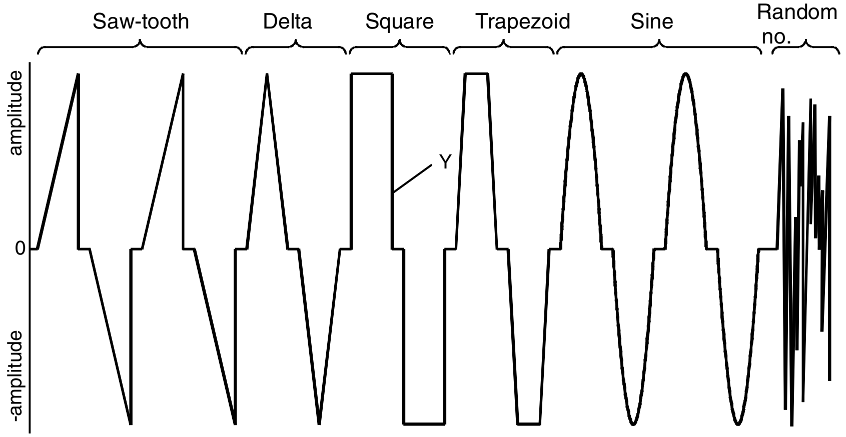
Timing diagrams

Bipolar operation

The following parameter specifications represent the various functions in bipolar operation:

Parameter	Specification
amplitude	1
halfperiod	10
t_off	2
t_rise	2
t_acc	0
unipolar	0

Bipolar operation

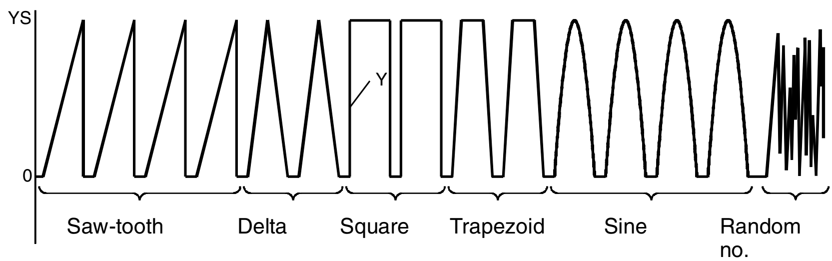


Unipolar operation

The following parameter specifications represent the various functions in unipolar operation:

Parameter	Specification
amplitude	1
halfperiod	10
t_off	2
t_rise	2
t_acc	0
unipolar	1

Unipolar operation

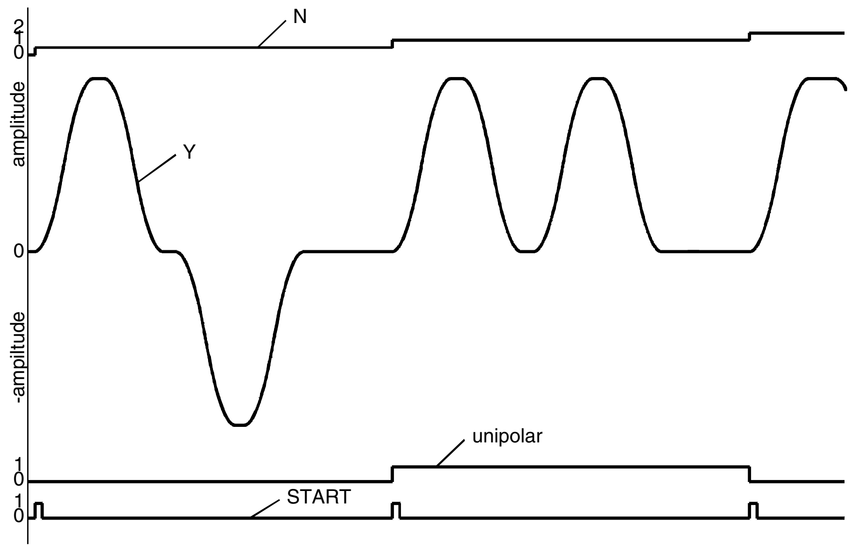


Trapezoid function

The following parameter specification represents the trapezoid function:

Parameter	Specification
amplitude	1
halfperiod	10
t_off	1
t_rise	4
t_acc	1.5

Trapezoid function



INTEG: Integrator with limit

12

Overview

At a glance

This chapter describes the INTEG block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block replicates a limited integrator.

The function block has the following properties:
- Operating modes, Manual, Stop, Automatic
- Manipulated variable limiting in automatic mode

As additional parameters, EN and ENO can be projected.

Formula

The transfer function is:

$$G(s) = \frac{gain}{s}$$

The formula of calculation is:

$$Y = Y_{(old)} + gain \times dt \times \frac{X_{(new)} + X_{(old)}}{2}$$

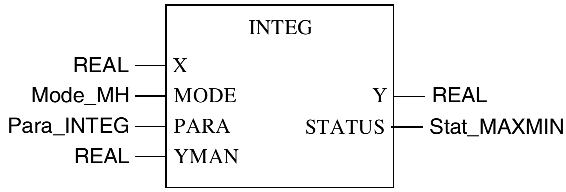
Meaning of the sizes

Size	Meaning
$X_{(old)}$	Value of input X from the previous cycle
$Y_{(old)}$	Value of output Y from the previous cycle
dt	Time difference between current and previous cycle

Representation

Symbol

Block representation



Description of the INTEG parameter

Block parameter description

Parameter	Data type	Meaning
X	REAL	Input variable
MODE	Mode_MH	Operating modes
PARA	Para_INTEG	Parameter
YMAN	REAL	Manually manipulated value
Y	REAL	Output
STATUS	Stat_MAXMIN	Output status

Parameter description Mode_MH

Data structure description

Element	Data type	Meaning
man	BOOL	"1" = Manual operating mode
halt	BOOL	"1" =Halt operating mode

Parameter description Para_INTEG

Data structure description

Element	Data type	Meaning
gain	REAL	Integral gain (units/second)
ymax	REAL	Upper limit
ymin	REAL	Lower limit

Parameter description Stat_MAXMIN

Data structure description

Element	Data type	Meaning
qmin	BOOL	"1" = Y has reached lower limit
qmax	BOOL	"1" = Y has reached upper limit

Detailed description

Parametering

The parameter assignments of the function block are satisfied by the determination of gain, the integral gain and the limiting values ymax und ymin for output Y.

The values ymax and ymin limit the upper and lower values of the output. So that means ymin ≤ Y ≤ ymax

If the threshold value is reached or the output signal is limited this will be indicated by qmax and qmin.
- qmax = 1 if Y ≥ ymax
- qmin = 1 when Y ≤ ymin

Operating mode

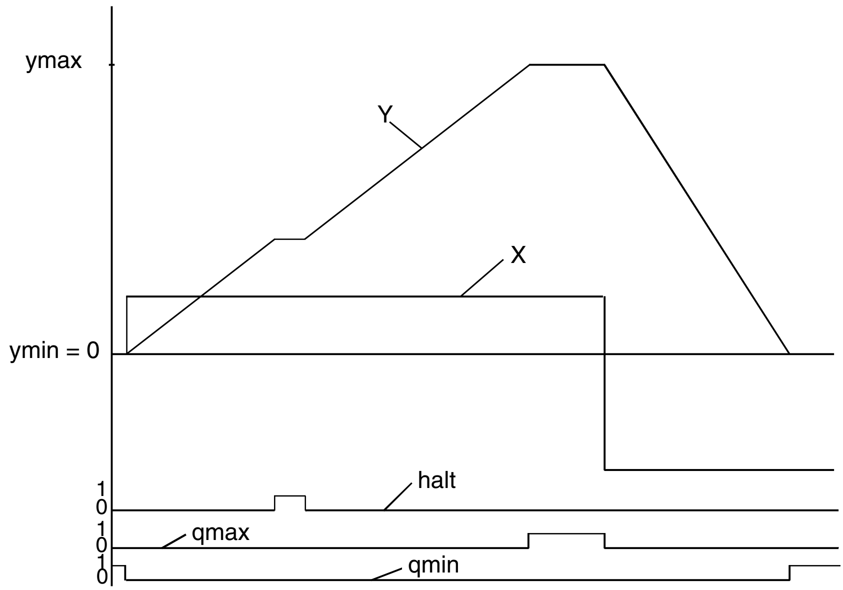
There are three operating mode selectable through the man and halt parameter inputs:

Operating mode	man	halt	Meaning
Automatic	0	0	The function block operates as described in "Parametering".
Manual mode	1	0 or 1	The manual value YMAN will be transmitted fixed to the output Y. The control output is, however, limited by ymax and ymin.
Halt	0	1	The output Y will be held at the last calculated value. The output will no longer be changed, but can, however, be overwritten by the user.

Example

The input signal is integrated via the time. The output follows jumps of the input X value in a ramp function of like polarity. Limiting of output Y within ymax and ymin with the appropriate signals at qmax and qmin can also be clearly seen.

Representation of the integrator jump response



Runtime error

Error message

There is an Error message, if
- an unauthorized floating point number is placed at the input YMAN or X,
- ymax < is ymin

INTEGRATOR: Integrator with limit

13

Overview

At a glance

This chapter describes the INTEGRATOR block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block replicates a limited integrator.

The function block has the following properties:
- Tracking and automatic modes
- Manipulated variable limiting in automatic mode

EN and ENO can be configured as additional parameters.

Formulas

The transfer function is:

$$G(s) = \frac{GAIN}{s}$$

The formula for the output OUT is:

$$OUT = OUT_{(old))} + GAIN \times dt \times \frac{IN_{(new)} + IN_{(old)}}{2}$$

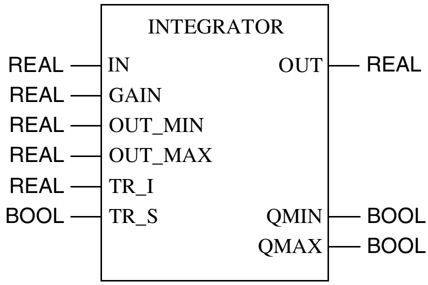
Meaning of variables

Variable	Meaning
$IN_{(new)}$	current value of input IN
$IN_{(old)}$	Value of the input IN from the previous cycle
$OUT_{(old)}$	Value of the output OUT from the previous cycle
dt	Time difference between the current cycle and the previous cycle

Display

Symbol

Block display



Parameter description

Block parameter description

Parameter	Data type	Meaning
IN	REAL	Input variable
GAIN	REAL	Integral gain
OUT_MIN	REAL	Lower output limit
OUT_MAX	REAL	Upper output limit
TR_I	REAL	Initialization input
TR_S	BOOL	Initialization type "1" = Tracking mode "0" = Automatic mode
OUT	REAL	Output
QMIN	BOOL	"1" = Output OUT has reached lower limit
QMAX	BOOL	"1" = Output OUT has reached upper limit

Detailed description

Parametering

Parameter assignment for the function block is accomplished by specifying the integration gain GAIN and the limiting values OUT_MAX and OUT_MIN for the output OUT.

The limits OUT_MAX and OUT_MIN retain the output within the prescribed range. So that means OUT_MIN ≤ OUT ≤ OUT_MAX.

The markers QMAX and QMIN are signalling that the limits or a limitation of the output signal have/has been reached.

- QMAX = 1 if OUT ≥ OUT_MAX
- QMIN = 1 if OUT ≤ OUT_MIN

Operating mode

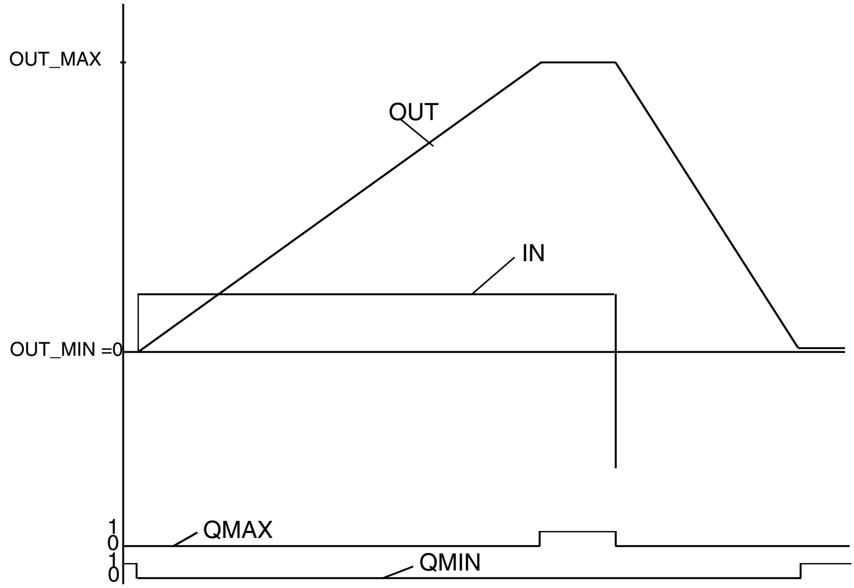
There are two operating mode selectable through the TR_S parameter input.

Operating mode	TR_S	Meaning
Automatic	0	The Function block operates as described in "Parametering".
Tracking	1	The tracking value TR_I is transferred permanently to the output OUT. The control output is, however, limited by OUT_MAX and OUT_MIN.

Example

The input signal is integrated using the time. In the event of a transition at the input IN, the output will rise (if the IN values are positive) or fall off (if the IN values are negative) along a ramp function. OUT will always be between OUTMAX and OUT_MIN; if OUT is equal to OUT_MAX or OUT_MIN, it will be so indicated in QMAX or QMIN.

It displays the integrator jump response:.



Runtime error

Error message

If OUT_MAX < OUT_MIN an Error message is generated.

INTEGRATOR1: Integrator with limit

14

Overview

At a glance

This chapter describes the INTEGRATOR1 block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block replicates a limited integrator.

The function block has the following properties:
- Manual, halt and automatic modes
- Manipulated variable limiting in automatic mode

EN and ENO can be configured as additional parameters.

Formulas

The transfer function is:

$$G(s) = \frac{GAIN}{s}$$

The formula for the output Y is:

$$Y = Y_{(old))} + GAIN \times dt \times \frac{X_{(new)} + X_{(old)}}{2}$$

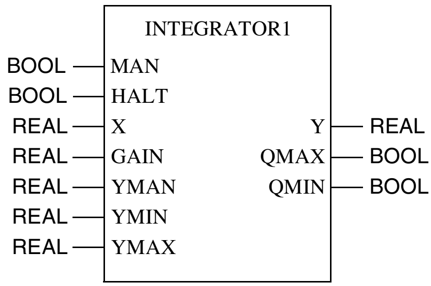
Meaning of variables

Variable	Meaning
$X_{(old)}$	Value of the input X from the previous cycle
$Y_{(old)}$	Value of the output Y from the previous cycle
dt	Time difference between current and previous cycle

Display

Symbol

Block display



Parameter description

Block parameter description

Parameter	Data type	Meaning
MAN	BOOL	"1" = Hand mode
HALT	BOOL	"1" = Halt mode
X	REAL	Input variable
GAIN	REAL	Integral gain
YMAX	REAL	Upper output limit
YMIN	REAL	Lower output limit
YMAN	REAL	Manual manipulated value
Y	REAL	Output
QMAX	BOOL	"1" = Output Y has reached upper limit
QMIN	BOOL	"1" = Output Y has reached lower limit

Detailed description

Parametering

The parametering of the function block is accomplished by specifying the integral gain GAIN and the limiting values YMAX and YMIN for the output Y.

The limits YMAX and YMIN retain the output within the prescribed range. Hence, YMIN ≤ Y ≤ YMAX.

The outputs QMAX and QMIN signal that the output has reached a limit, and thus been capped.

- QMAX = 1 if Y ≥ YMAX
- QMIN = 1 if Y ≤ YMIN

Operating mode

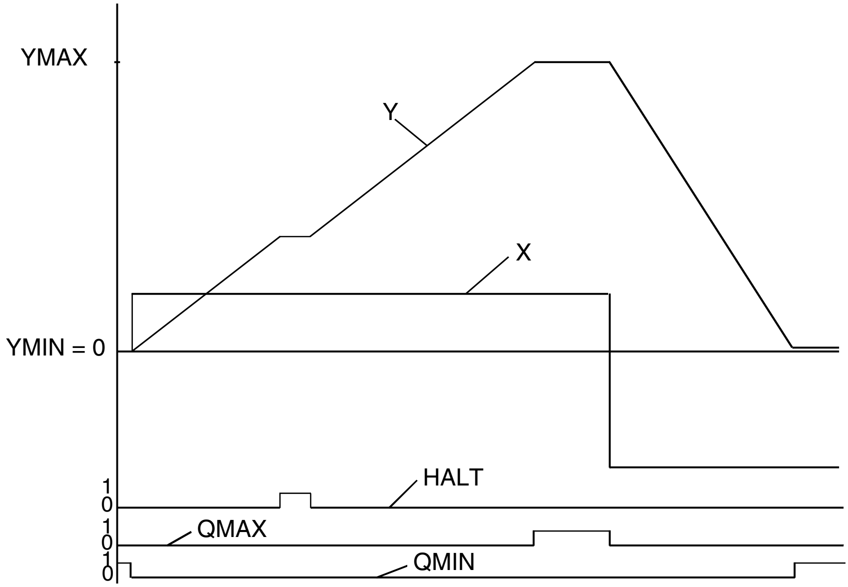
There are three operating mode selectable through the inputs MAN and HALT:

Operating mode	MAN	HALT	Meaning
Automatic	0	0	The function block operates as described in "Parametering".
Manual	1	0 or 1	The manual value YMAN will be transferred directly to the output Y. The control output is, however, limited by YMAX and YMIN.
Halt	0	1	The output Y will be set at the last calculated value.

Example

The input signal is integrated via the time. The output follows jumps of the input X value in a ramp function of like polarity. Limiting of output Y within YMAX and YMIN with the appropriate signals at QMAX and QMIN can also be clearly seen.

Representation of the integrator jump response:.



Runtime error

Error message

If YMAN < YMIN an Error message is generated.

K_SQRT: Square root

15

Overview

At a glance

This chapter describes the K_SQRT block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This Function block calculates the weighted square root of a numerical value. A division can be defined under which the function block issues the value zero.

Taking the square root typically serves to linearize a flow measurement using a throttle device.

EN and ENO can be configured as additional parameters.

Formula

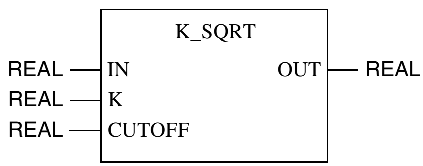
The function block performs the following calculation:

Calculation	Condition
$OUT = K\sqrt{IN}$	$IN \geq CUTOFF$
$OUT = 0$	$IN < 0$ or $IN < CUTOFF$

Presentation

Symbol

Block display



Parameter description

Block parameter description

Parameter	Data type	Meaning
IN	REAL	Numerical value to process
K	REAL	Weighting coefficient
CUTOFF	REAL	Division
OUT	REAL	Result of the calculation

Runtime error

Error message

An error is displayed if a non floating point value is recorded at input or if there is a problem with floating point calculation. In this case the output OUT remains unchanged.

Warning

A warning is given if the CUTOFF input is negative. The function block then uses the value 0 for calculation.

LAG: Time lag device: 1st order

16

Overview

At a glance

This chapter describes the LAG block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block represents a first order delay (low pass)

The function block contains the following operating mode:
- Manual
- Halt
- Automatic

EN and ENO can be projected as additional parameters.

Equation

The transmission function says:

$$G(s) = gain \times \frac{gain}{1 + s \times lag}$$

The calculation equation says:

$$Y = Y_{(old)} + \frac{dt}{lag + dt} \times \left(gain \times \frac{X_{(old)} + X_{(new)}}{2} - Y_{(old)}\right)$$

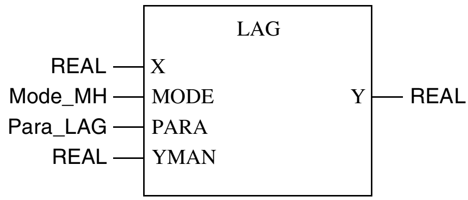
Meaning of the sizes

Size	Meaning
$X_{(old)}$	Value of output X from the previous cycle
$Y_{(old)}$	Value of the output Y from the previous cycle
dt	Time difference between current and previous cycle

Presentation

Symbol

Block display



Parameter description LAG

Block parameter description

Parameter	Data type	Meaning
X	REAL	Input value
MODE	Mode_MH	Operating mode
PARA	Para_LAG	Parameter
YMAN	REAL	Manual manipulation
Y	REAL	Output

Parameter description Mode_MH

Data structure description

Element	Data type	Meaning
man	BOOL	"1" = Operating mode Hand
halt	BOOL	"1" = Halt mode

Parameter description Para_LAG

Data structure description

Element	Data type	Meaning
gain	REAL	Gain factor
lag	TIME	Delayed time constants

Detailed description

Parametering

The parametering of the Function block is achieved through specification of the boost factor gain as well as the parametering of the delayed time constants lag.

The unit jump at input X (jump at input X of 0 to 1.0) succeeds the output Y with delay. Along an e-function

$$\exp(-t/lag)$$

it will approximate the value $gain \times X$.

Operating mode

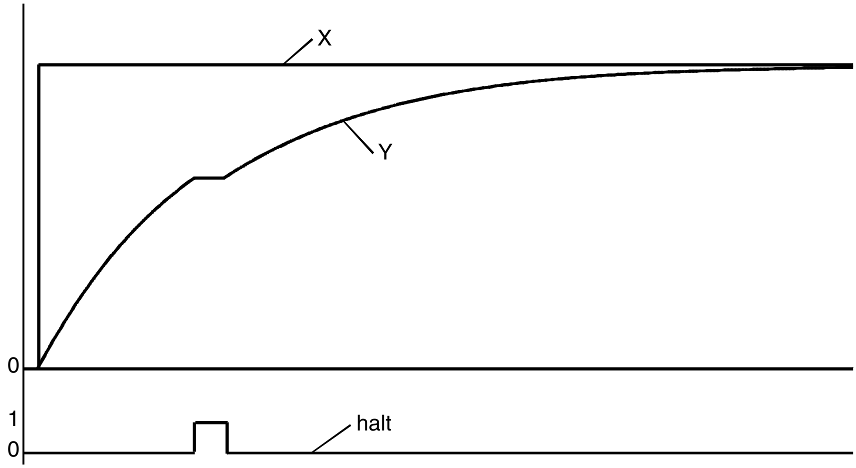
There are three operating modes selectable through the man and halt parameter inputs:

Operating mode	man	halt	Meaning
Automatic	0	0	The function block operates as described in "Parametering".
Manual mode	1	0 or 1	The manual value YMAN will be transmitted fixed to the output Y.
Halt	0	1	The output Y will be set at the last calculated value. The output will no longer be changed, but can be overwritten by the user.

Example

The diagram shows an example of the jump response of the function block. Input X jumps to a new value that output Y approaches exponentially.

Function block LAG jump response with gain = 1



LAG1: Time lag device: 1st order

17

Overview

At a glance

This chapter describes the LAG1 block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block represents a first order delay.

The function block contains the following operating mode:
- Manual mode
- Halt
- Automatic

EN and ENO can be projected as additional parameters.

Equation

The transmission function says:

$$G(s) = gain \times \frac{1}{1 + s \times lag}$$

The calculation equation says:

$$Y = Y_{(old)} + \frac{dt}{LAG + dt} \times \left(gain \times \frac{X_{(old)} + X_{(new)}}{2} - Y_{(old)}\right)$$

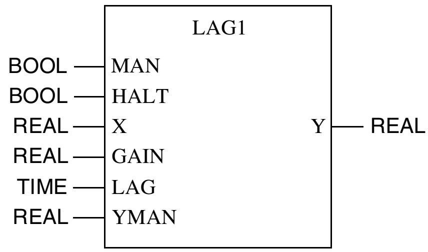
Meaning of the sizes

Size	Meaning
$X_{(old)}$	Value of output X from the previous cycle
$Y_{(old)}$	Value of the output Y from the previous cycle
dt	Time difference between current and previous cycle

Presentation

Symbol

Block display



Parameter description

Block parameter description

Parameter	Data type	Meaning
MAN	BOOL	"1" = Operating mode Hand
HALT	BOOL	"1" = Halt mode
X	REAL	Input value
GAIN	REAL	Gain factor
LAG	TIME	Delayed time constants
YMAN	REAL	Manual manipulation
Y	REAL	Output

Detailed description

Parametering

The parametering of the Function block is achieved through specification of the boost factor GAIN as well as the parametering of the delayed time constants LAG.

The unit jump at input X (jump at input X of 0 to 1.0) succeeds the output Y delay. Along an e-function

$$\exp(-t/(LAG))$$

it will approximate the value $GAIN \times X$.

Operating mode

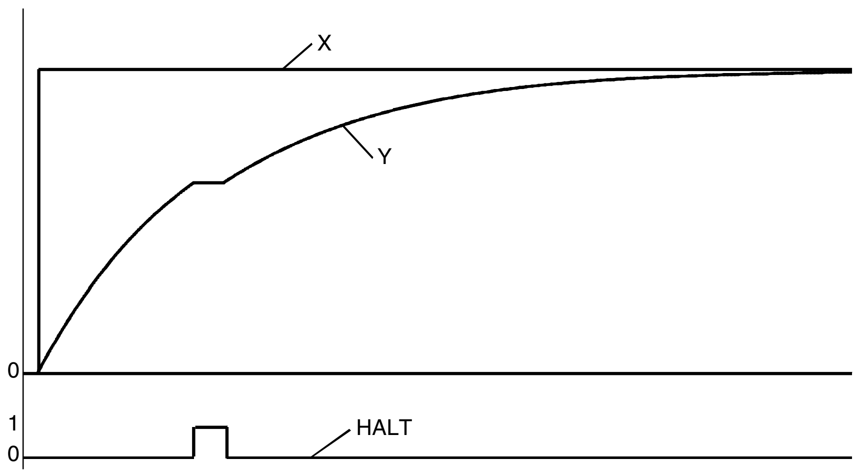
There are three operating mode, which are selected via the elements MAN and HALT:

Operating mode	MAN	HALT	Meaning
Automatic	0	0	The function block operates as described in "Parametering".
Manual mode	1	0 or 1	The manual value YMAN will be transmitted fixed to the output Y.
Halt	0	1	The output Y will be held at the last calculated value.

Example

The diagram shows an example of the jump response of the PLAG device: Input X jumps to a new value that output Y approaches exponentially.

Function block LAG1 jump response with GAIN = 1



LAG2: Time lag device: 2nd order

18

Overview

At a glance

This chapter describes the LAG2 block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block LAG2 represents a second order with delay.

The function block contains the following operating mode:
- Manual mode
- Halt
- Automatic

EN and ENO can be projected as additional parameters.

Equation

The transmission function says:

$$G(s) = gain \times \frac{1}{1 + s \times 2 \times \frac{dmp}{freq} + \left(\frac{s}{freq}\right)^2}$$

The calculation equation is as follows:

$$Y_{(new)} = A \times B$$

where

$$A = \frac{gain \times X \times (freq \times dt)^2 + Y_{(old)}}{1 + 2 \times dmp \times freq \times dt + (freq \times dt)^2}$$

and

$$B = \frac{(2 \times dmp \times freq \times dt \times 2) - Y_{(old2)}}{1 + 2 \times dmp \times freq \times dt + (freq \times dt)^2}$$

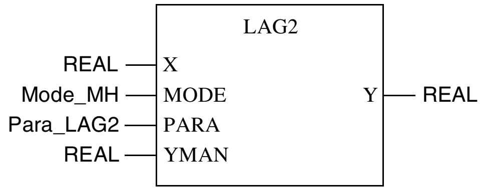
Meaning of the sizes

Size	Meaning
$Y_{(old)}$	Value of the output Y from the previous cycle
$Y_{(old2)}$	Value of the output Y from the cycle preceding the previous
dt	Time difference between current and previous call

Presentation

Symbol

Block display



parameter description LAG2

Block parameter description

Parameter	Data type	Meaning
X	REAL	Input value
MODE	Mode_MH	Operating mode
PARA	Para_LAG2	Parameter
YMAN	REAL	Manual manipulated value for output
Y	REAL	Output

Parameter description Mode_MH

Data structure description

Element	Data type	Meaning
man	BOOL	"1" = Operating mode Hand
halt	BOOL	"1" = Halt mode

Parameter description Para_LAG2

Data structure description

Element	Data type	Meaning
gain	REAL	Gain factor
dmp	REAL	Dampening
freq	REAL	Natural frequency

Detailed description

Parametering

The parameter assignments of the function block are satisfied by the determination of gain, the gain and the values for dampening dmp, and natural frequency freq.

Dampening dmp and natural frequency freq must have positive values.

Output Y follows input X jumps in a dampened oscillation. The period of undampened oscillation is T = 1/freq. For dampening values dmp < 1 reference is made to a dampened oscillation. For dampening values ≥ 1 reference is made to non-resonant behavior (i.e. without oscillation); in this case the output follows the input in the same way as with 2 LAG function blocks, which are switched in series.

Operating mode

There are three operating mode selectable through the man and halt parameter inputs:

Operating mode	man	halt	Meaning
Automatic	0	0	The function block operates as described in "Parametering".
Manual mode	1	0 or 1	The manual value YMAN will be transmitted fixed to the output Y.
Halt	0	1	The output Y will be held at the last calculated value. The output will no longer be changed, but can be overwritten by the user.

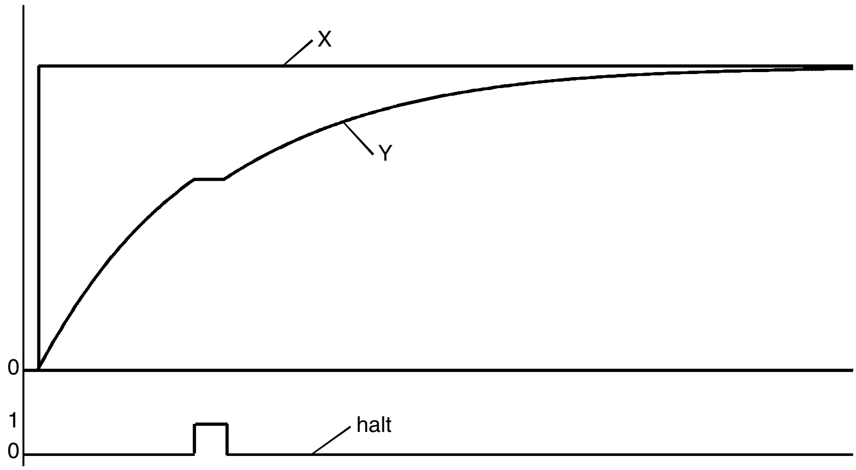
Timing diagrams

Overview

The following diagrams show examples of the LAG2 device's jump response with varying parameters.

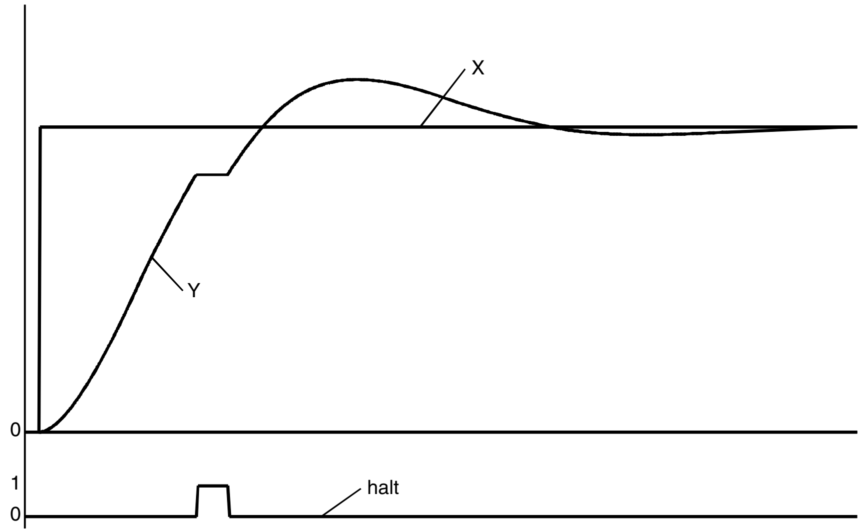
Dampening dmp = 1

For a dampening of dmp = 1 the output Y follows input X with a non-resonant action.



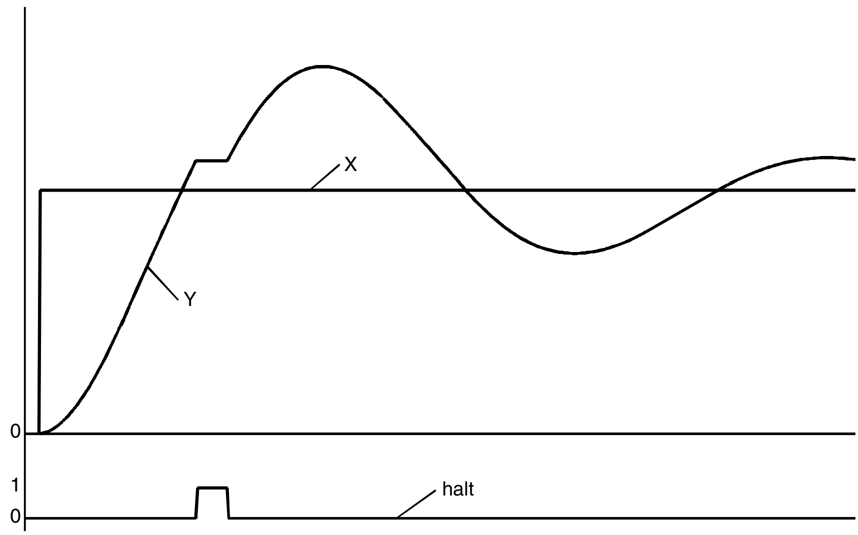
Dampening dmp = 0.5

For a dampening of dmp = 0.5 the output Y follows input X in a dampened periodic manner.



Dampening dmp = 0.2

For a dampening of dmp = 0.2 it is clear that the jump response is considerably less dampened.



LAG_FILTER: Time lag device: 1st order

19

Overview

At a glance

This chapter describes the LAG_Filter block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block represents a first order delay.

The function block contains the following operating mode:
- Tracking
- Automatic

EN and ENO can be projected as additional parameters.

Equation

The transmission function says:

$$G(s) = GAIN \times \frac{1}{1 + s \times LAG}$$

The calculation equation says:

$$OUT = OUT_{(old)} + \frac{dt}{LAG + dt} \times \left(GAIN \times \frac{IN_{(old)} + IN_{(new)}}{2} - OUT_{(old)}\right)$$

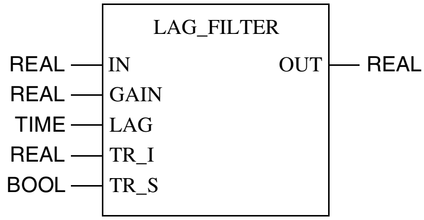
Meaning of the sizes

Size	Meaning
$IN_{(old)}$	Value of the input IN from the previous cycle
$OUT_{(old)}$	Value of the output OUT from the previous cycle
dt	Time difference between current and previous cycle

Representation

Symbol

Representation of the block



Parameter description

Block parameter description

Parameter	Data type	Meaning
IN	REAL	Input value
GAIN	REAL	Gain factor
LAG	TIME	Delayed time constants
TR_I	REAL	Initialization input
TR_S	BOOL	Initialization type "1" = Operating mode Tracking "0" = Halt mode
OUT	REAL	Output

Detailed description

Parametering

The parametering of the Function block is achieved through specification of the boost factor GAIN as well as the parametering of the delayed time constants LAG.

The unit step at the input IN (jump at the input IN from 0 to 1.0) is followed by the output OUT with a lag time. Along an e-function

$$\exp(-t / LAG)$$

it will approximate the value $GAIN \times X$.

Operating mode

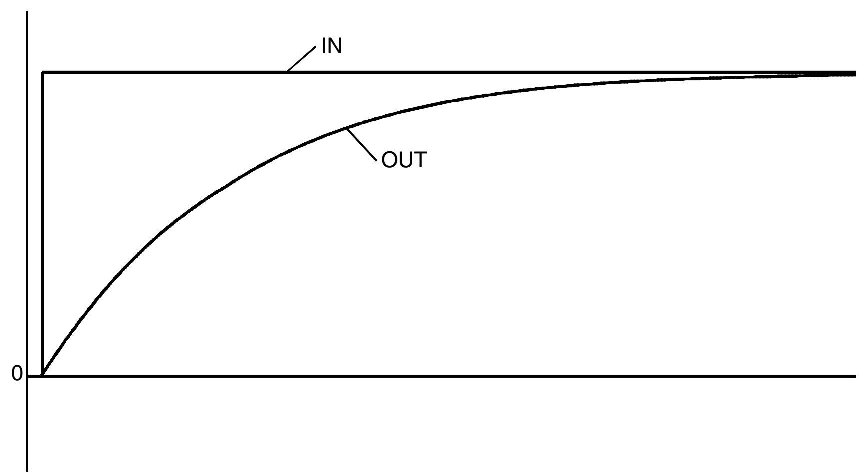
There are two operating mode, which can be selected via the input TR_S:

Operating mode	TR_S	Meaning
Automatic	0	The function block operates as described in "Parametering".
Tracking	1	The tracking value TR_I is transmitted permanently to the output OUT.

Example

The diagram shows an example of the jump response of the LAG_FILTER function block. The input IN jumps to a new value and the output OUT follows the input IN along an e-function.

Jump response of the function block LAG_FILTER when GAIN = 1



LDLG: PD device with smoothing

20

Overview

At a glance

This chapter describes the LDLG block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block serves as a PD outline with subsequent smoothing.

The function block has the following properties:
- Definable delay of the D-component
- Tracking and automatic modes

EN and ENO can be projected as additional parameters.

Formula

The transfer function is:

$$G(s) = GAIN \times \frac{1 + s \times LEAD}{1 + s \times LAG}$$

The formula of calculation is:

$$OUT = \frac{LAG \times OUT_{(old)} + GAIN \times ((LEAD + dt) \times IN - LEAD \times IN_{(old)})}{LAG + dt}$$

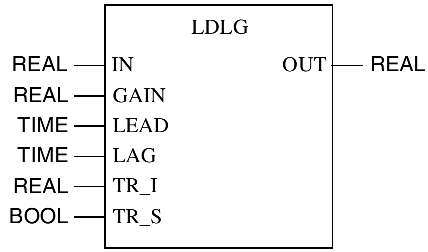
Meaning of the sizes

size	Meaning
$IN_{(old)}$	Value of the input IN from the previous cycle
$OUT_{(old)}$	Value of the output OUT from the previous cycle
dt	is the time differential between the current cycle and the previous cycle

Representation

Symbol

Representation of the block



Parameter description

Block parameter description

Parameter	Data type	Meaning
IN	REAL	Input
GAIN	REAL	Gain factor
LEAD	TIME	Dirivative time constant
LAG	TIME	Delayed time constants
TR_I	REAL	Initialization input
TR_S	BOOL	Initialization type "1" = Operating mode Tracking "0" = Halt mode
OUT	REAL	Output

Detailed description

Parametering

The parametering of the Function block appears through specification of the boost factors GAIN as well as the parametering of the Derivative time constants LEAD and the delayed time constants LAG.

For very small sample times and the unit jump to input IN (jump at line-in IN from 0 to 1.0) output OUT will jump to the value $\mathrm{GAIN} \times \mathrm{LEAD} / \mathrm{LAG}$ (theoretical value - actual slightly smaller, due to the not infinitely small sample times), using the time constant LAG to approximate the value $\mathrm{GAIN} \times 1.0$ closer.

Operating mode

There are two operating mode, which can be selected via the input TR_S:

Operating mode	TR_S	Meaning
Automatic	0	The function block operates as described in "Parametering".
Tracking	1	The tracking value TR_I is transmitted permanently to the output OUT.

Examples of function block LDLG

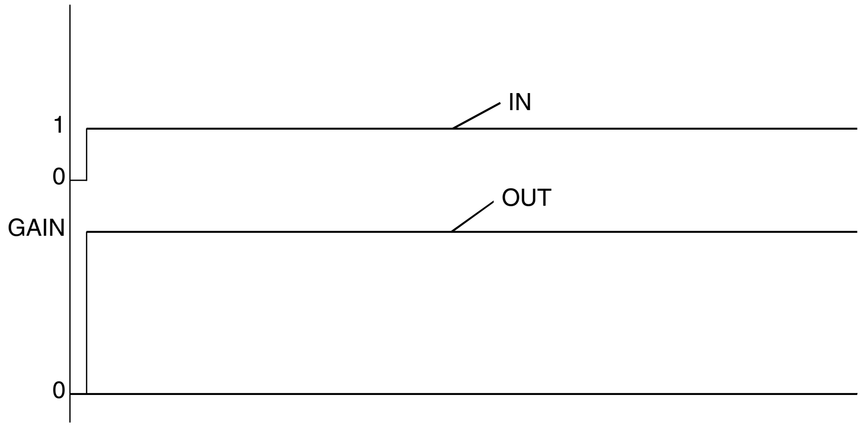
Example-overview

The following examples are presented in the following diagrams:
- LEAD = LAG
- LEAD/LAG = 0,5, GAIN = 1
- LEAD/LAG = 2, GAIN = 1

LEAD = LAG

The function block behaves like a pure multiplication block with the multiplier GAIN.

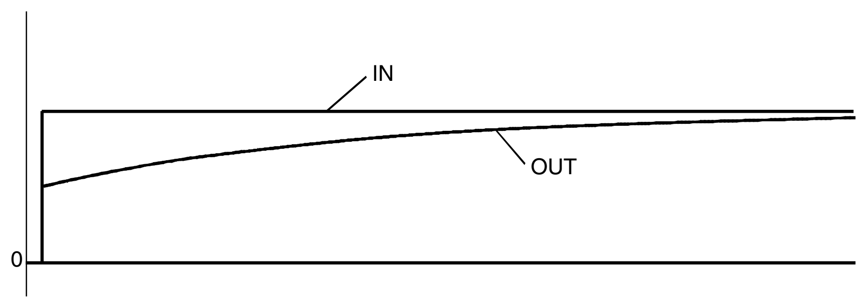
Function block LDLG with LEAD = LAG



LEAD/LAG = 0,5, GAIN = 1

In this case the output OUT will jump to half the accumulated value in order to then transition to the upper range value (GAIN * IN) with the lag time constant LAG.

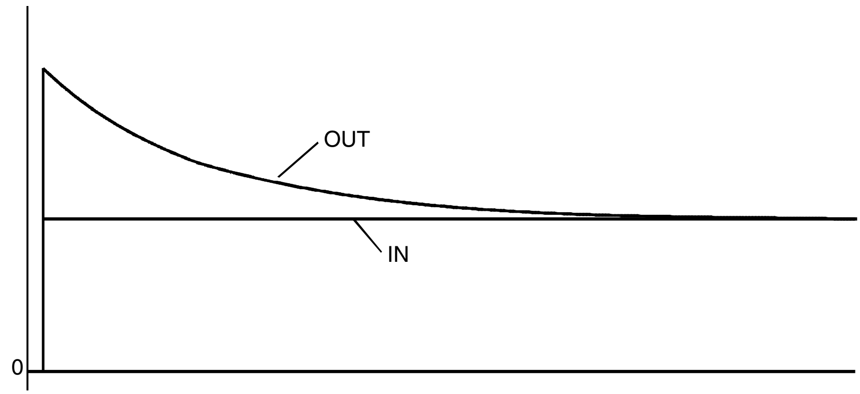
Function block LDLG with LEAD/LAG = 0,5 and GAIN = 1



LEAD/LAG = 2, GAIN = 1

In this case the output OUT will jump to twice the accumulated value in order to then transition to the end value (GAIN * IN) with the lag time constant LAG.

Function block LDLG with LEAD/LAG = 2 and GAIN = 1



LEAD: Differentiator with smoothing

21

Overview

At a glance

This chapter describes the LEAD block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block is a differentiator element with an output OUT delayed by the lag time constant LAG.

The function block contains the following operating modes:
- Tracking
- Automatic

EN and ENO can be projected as additional parameters.

Formula

The transfer function for OUT is:

$$G(s) = GAIN \times \frac{s}{1 + s \times LAG}$$

The formula of calculation is:

$$OUT = \frac{LAG}{dt + LAG} \times (OUT_{(old)} + GAIN \times (IN_{(new)} - IN_{(old)}))$$

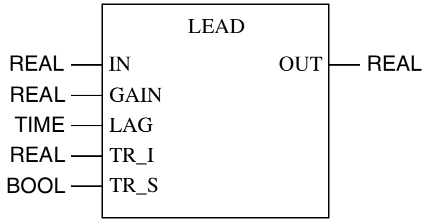
Meaning of the sizes

size	**Meaning**
$IN_{(new)}$	Value of the input IN from the current cycle
$IN_{(old)}$	Value of the input IN from the previous cycle
$OUT_{(old)}$	Value of the output OUT from the previous cycle
dt	is the time differential between the current cycle and the previous cycle

Representation

Symbol

Block representation



Parameter description

Block parameter description

Parameter	Data type	Meaning
IN	REAL	Input value
GAIN	REAL	Gain of the differentiation
LAG	TIME	Delay time constants
TR_I	REAL	Initialization input
TR_S	BOOL	Initialization type "1" = Tracking mode "0" = Automatic mode
OUT	REAL	Output derivative unit with smoothing

Detailed description

Parametering

Parameter assignment for this function block is accomplished by selecting the GAIN of the derivative unit and the lag time constant LAG by which the output OUT will be delayed.

For very short scan times, after a unit step at the input IN (jump at input IN from 0 to 1.0), the output OUT will jump to the value of GAIN (theoretical value - in reality somewhat smaller due to the fact that the scan time is not infinitely short), to then return to 0 with the time constant LAG.

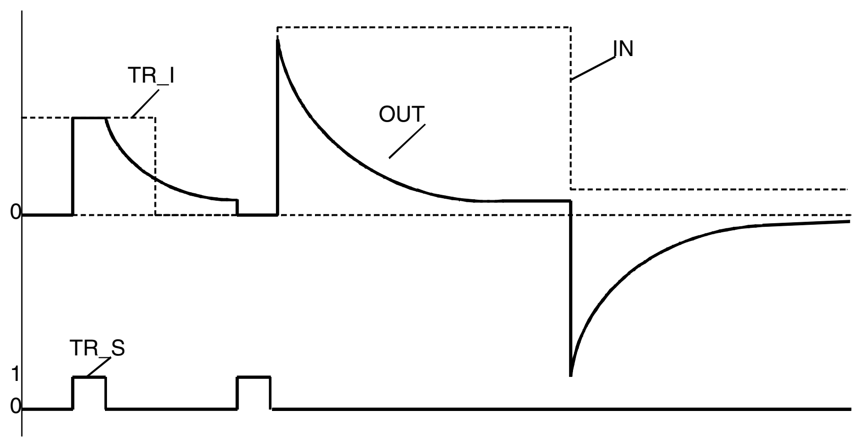
Operating mode

There are two operating mode selectable using the input TR_S:

Operating mode	TR_S	Meaning
Automatic	0	The function block operates as described in "Parametering".
Tracking	1	The tracking value TR_I is transferred directly to the output OUT.

Example

Representation of the LEAD function block jump response with GAIN = 1 and LAG = 10s:



LEAD_LAG: PD device with smoothing

22

Overview

At a glance

This chapter describes the LEAD_LAG block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block implements a PD element with following low-pass filter.

The function block has the following properties:
- Definable delay of the D-component
- Manual, halt and automatic modes

EN and ENO can be configured as additional parameters.

Formula

The transfer function is:

$$G(s) = gain \times \frac{1 + s \times lead}{1 + s \times lag}$$

The calculation formula is:

$$Y = \frac{lag \times Y_{(old)} + gain \times ((lead + dt) \times X - lead \times X_{(old)})}{lag + dt}$$

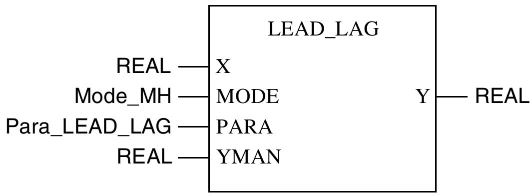
Meaning of the variables

Variable	Meaning
$X_{(old)}$	Value of input X from the previous cycle
$Y_{(old)}$	Value of output Y from the previous cycle
dt	Time difference between current and previous cycle

Representation

Symbol

Block representation



Parameter description LEAD_LAG

Block parameter description

Parameter	Data type	Meaning
X	REAL	Input
MODE	Mode_MH	Operating mode
PARA	Para_LEAD_LAG	Parameter
YMAN	REAL	Manual value manipulated value
Y	REAL	Output

Parameter description Mode_MH

Data structure description

Element	Data type	Meaning
man	BOOL	"1" = Manual mode
halt	BOOL	"1" =Halt mode

Parameter description Para_LEAD_LAG

Data structure description

Element	Data type	Meaning
gain	REAL	Gain factor
lead	TIME	Derivative time constant
lag	TIME	Delay time constants

Detail description

Parametering

The parametering of the Function block is achieved through specification of the boost factor gain as well as the parametering of the Derivative time constant lead and the delayed time constants lag.

For very small sample times and the unit jump at input X (jump at input X from 0 to 1.0) output Y will jump to the value $\mathrm{gain} \times \mathrm{lead}/\mathrm{lag}$ (theoretical value - actual slightly smaller, due to the not infinitely small sample times), using the time constant lag to approximate the value $\mathrm{gain} \times 1.0$

Operating mode

There are three operating mode, which are selected via the elements man and halt:

Operating mode	man	halt	Meaning
Automatic	0	0	The Function block will be handled, as described in "Parametering".
Hand	1	0 or 1	The hand value YMAN will be transmitted permanently to the output Y.
Halt	0	1	The output Y will be set at the last calculated value. The output will no longer be changed, but can be overwritten by the user.

Examples of function blocks LEAD_LAG

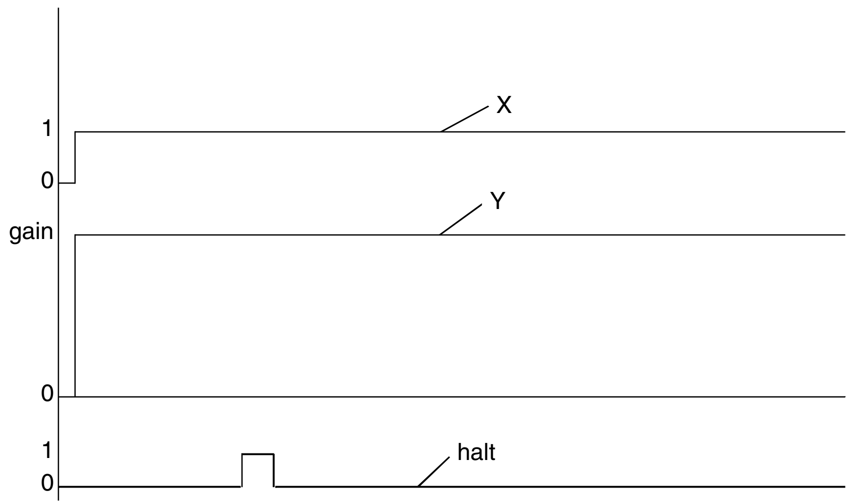
Example-overview

The following examples are presented in the following diagrams:
- lead = lag
- lead=lag * 0.5, gain = 1
- lead/lag = 2, gain = 1

lead = lag

The function blocks behave like a pure multiplication block with the multiplier gain.

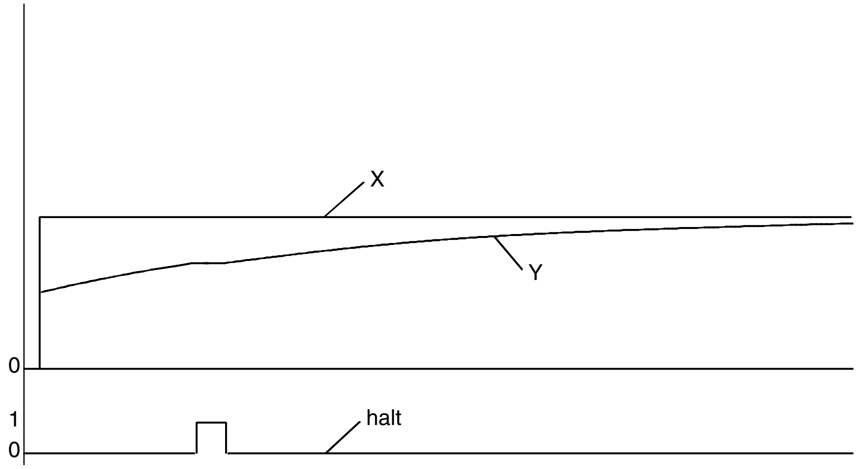
Function blockLEAD_LAG with lead = lag



lead=lag * 0.5, gain = 1

The output Y jumps in this case to half the end value in order to run into the end value with the delayed time constant lag (gain * X)

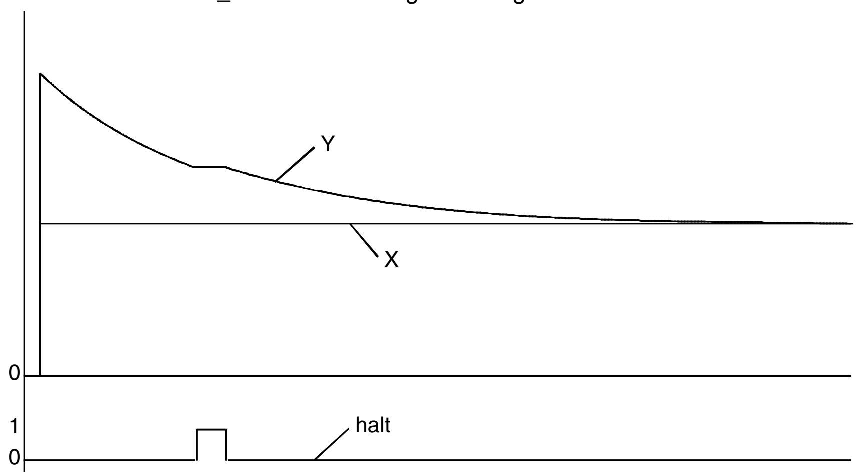
Function block LEAD_LAG with lead/lag = 0.5 and gain = 1



lead/lag = 2, gain = 1

The output Y jumps in this case to double the end value in order to run into the end value with the delayed time constant lag (gain * X)

Function block LEAD_LAG with lead/lag = 2 and gain = 1



Runtime error

Error message

An Error message, appears when an invalid floating point number lies at input YMAN or X.

LEAD_LAG1: PD device with smoothing

23

Overview

At a glance

This chapter describes the LEAD_LAG1 block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block serves as a PD outline with subsequent smoothing.

The function block contains the following properties:
- Definable delay of the D-component
- Operating mode, hand, halt, automatic

EN and ENO can be projected as additional parameters.

equation

The transmission function says:

$$G(s) = GAIN \times \frac{1 + s \times LEAD}{1 + s \times LAG}$$

The calculation equation says:

$$Y = \frac{LAG \times Y_{(old)} + GAIN \times ((LEAD + dt) \times X - LEAD \times X_{(old)})}{LAG + dt}$$

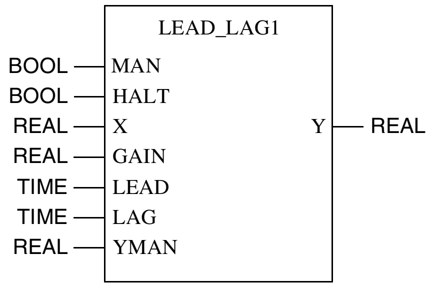
Meaning of the sizes

Size	Meaning
$X_{(old)}$	Value of output Y from the previous cycle
$Y_{(old)}$	Value of the input X from the previous cycle
dt	Time difference between current and previous cycle

Display

Symbol

Block display



Parameter description

Block parameter description

Parameter	Data type	Meaning
MAN	BOOL	"1" = Operating mode Hand
HALT	BOOL	"1" = Halt mode
X	REAL	Input
GAIN	REAL	Gain factor
LEAD	TIME	Derivative time constants
LAG	TIME	Delayed time constants
YMAN	REAL	Manual value-rank value
Y	REAL	Output

Detailed description

Parametering

The parametering of the Function block appears through specification of the boost factors GAIN as well as the parametering of the Derivative time constants LEAD and the delayed time constants LAG.

For very small sample times and the unit jump to input X (jump at line-in X from 0 to 1.0) output Y will jump to the value $\mathrm{GAIN} \times \mathrm{LEAD} / \mathrm{LAG}$ (theoretical value - actual slightly smaller due to the, not infinitely small sample times), using the time constant LAG to approximate the value $\mathrm{GAIN} \times 1.0$ closer.

Operating mode

There are three operating mode, which are selected via the elements MAN and HALT:

Operating mode	MAN	HALT	Meaning
Automatic	0	0	The Function block will be handled as "Parametering" describes.
Hand	1	0 or 1	The hand value YMAN will be transmitted fixed to the output Y.
Halt	0	1	The output Y will be held at the last calculated value.

Examples of function blocks LEAD_LAG1

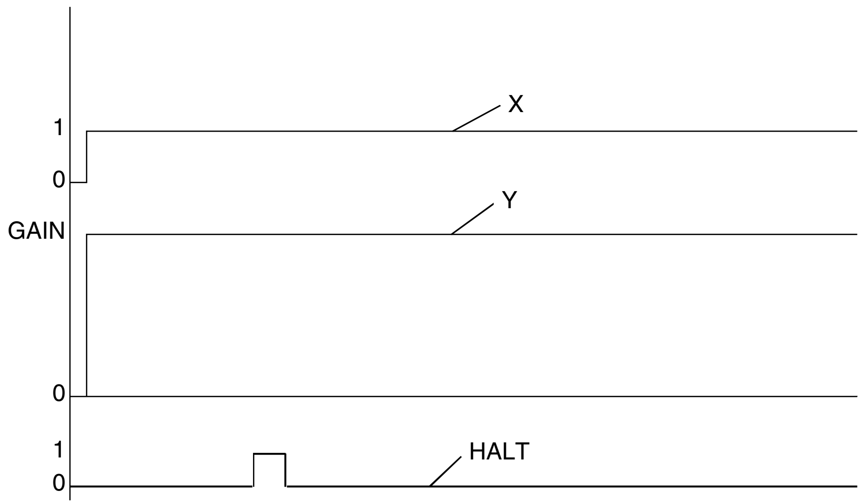
Example-overview

The following examples are presented in the following diagrams:
- LEAD = LAG
- LEAD=LAG * 0.5, GAIN = 1
- LEAD/LAG = 2, GAIN = 1

LEAD = LAG

The function block behaves like a pure multiplication block with the multiplier GAIN.

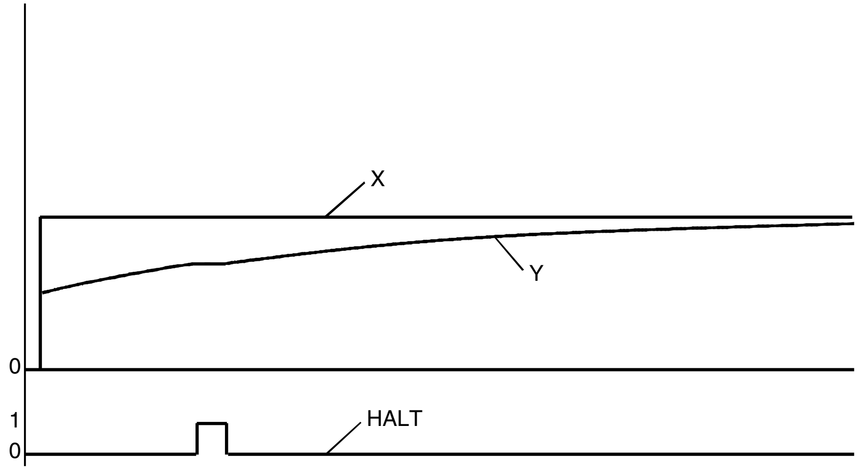
Function blockLEAD_LAG1 with LEAD = LAG



LEAD=LAG * 0.5, GAIN = 1

The output Y jumps in this case to half the end value in order to run into the end value with the delayed time constant lag (GAIN * X)

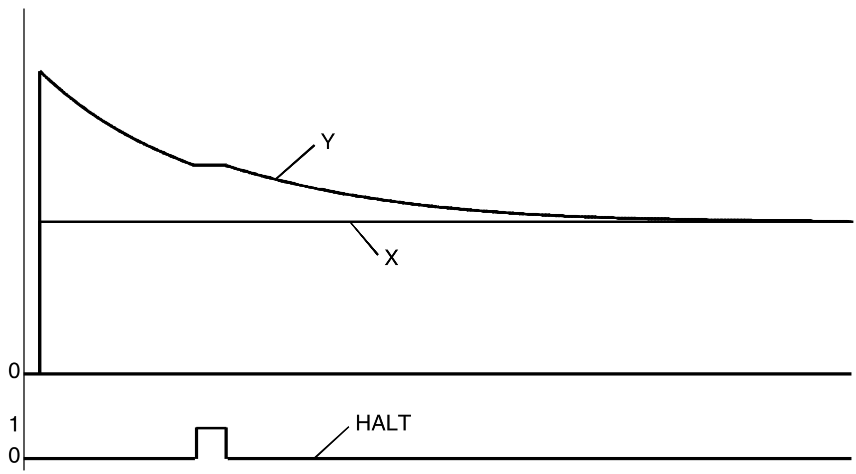
Function block LEAD_LAG1 with LEAD/LAG = 0.5 and GAIN = 1



LEAD/LAG = 2, GAIN = 1

The output Y jumps in this case to double the end value in order to run into the end value with the delayed time constant LAG (GAIN * X).

Function block LEAD_LAG1 with LEAD/LAG = 2 and GAIN = 1



LIMV: Velocity limiter: 1st order

24

Overview

At a glance

This chapter describes the LIMV block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block realizes a velocity limiter 1. Order with limiting of the manipulated variable.

The gradient of the input size X is limited to a specified value RATE. Further the output Y will be limited through YMAX and YMIN. This allows the function block to adjust signals to the technologically limited pace and limits from controlling elements.

EN and ENO can be projected as additional parameters.

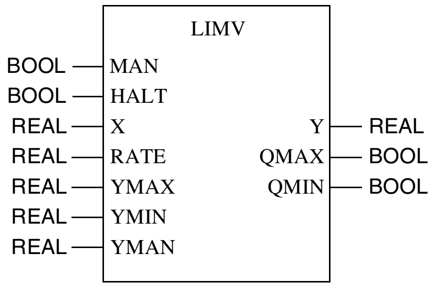
Properties

The function block contains the following properties:
- Operating mode, Hand, Halt, Automatic
- Manipulated variable limiting in automatic action

Display

Symbol

Block display



Parameter description

Block parameter description

Parameter	Data type	Meaning
MAN	BOOL	"1" = Operating mode Hand
HALT	BOOL	"1" = Halt mode
X	REAL	Input
RATE	REAL	Maximum upper limit (maximum x')
YMAX	REAL	Upper limit
YMIN	REAL	Lower limit
YMAN	REAL	Manual manipulated value
Y	REAL	Output
QMAX	BOOL	"1" = Output Y has reached upper limit
QMIN	BOOL	"1" = Output Y has reached lower limit

Detailed description

Parametering

The parametering of the function block appears through specification of the maximum upper speed RATE as well as the limits YMAX and YMIN for output Y. The maximum upper speed specifies to which value the output can change within one second.

The amount will be resolved from the parameter RATE. Ist RATE = 0, then Y = X.

The limits YMAX and YMIN limit the upper output as well as the lower output. So that means YMIN ≤ Y ≤ YMAX.

Reaching the bound value, i.e. a limit of the output signals will be shown at both the outputs, QMAX and QMIN:
- QMAX = 1 if Y ≥ YMAX
- QMIN = 1 if Y ≤ YMIN

Operating mode

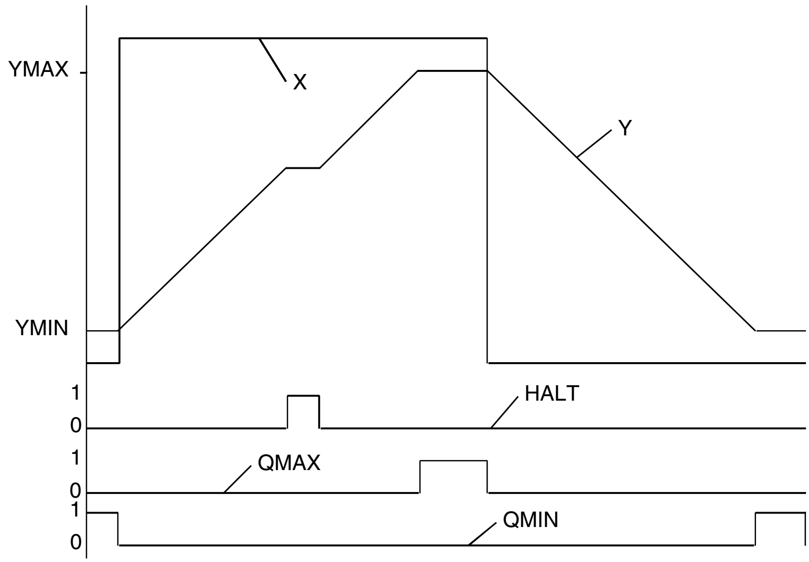
There are three operating mode, which are selected via the elements MAN and HALT:

Operating mode	MAN	HALT	Meaning
Automatic	0	0	The current value for Y will be constantly calculated and spent.
Hand	1	0 or 1	The manual value YMAN will be transmitted fixed to the output Y. The control output is, however, limited through YMAX and YMIN.
Halt	0	1	The output Y will be held at the last calculated value.

Example

The function block follows the jump to input X with maximum change in speed. Output Y remains at a standstill in Halt mode, in order to subsequently move on from the rank at which it has stopped. It is also clear to see the limits of output Y through YMAX and YMIN with the relevant messages QMAX and QMIN.

Dynamic behavior of LIMV



Runtime error

Error message

With YMAN < YMIN an Error message appears

MFLOW: mass flow block

25

Overview

At a glance

This chapter describes the MFLOW block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block MFLOW calculates the mass flow of a gas in a throttle device due to the differential pressure and the temperature and pressure conditions of the gas.

The measure of the differential pressure can be replaced by the speed of the medium or with another measure with pressure and temperature compensation.

EN and ENO can be projected as additional parameters.

Equation

The full equation (i.e. with en_sqrt = 1, en_pres = 1 and en_temp =1) says as follows:

$$OUT = k \times \sqrt{\frac{IN \times PA}{TA}}$$

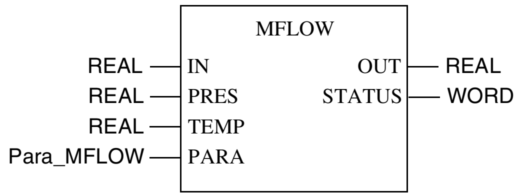
Meaning of the sizes

Size	Meaning
SV	Gas pressure in absolute units
TA	Absolute gas temperature in Kelvin

Representation

Symbol

Block representation



Parameter description MFLOW

Block parameter description

Parameter	Data type	Meaning
IN	REAL	Input
PRES	REAL	Absolute or relative gas pressure
TEMP	REAL	Gas temperature printed out in °C or °F
PARA	Para_MFLOW	Parameter
OUT	REAL	Value of the mass flow, with temperature and pressure correction
STATUS	WORD	Status word

Parameter description Para_MFLOW

Data structure description

Element	Data type	Meaning
k	REAL	Calculating constants (see *Calculation of the constant k, p. 212*)
en_pres	BOOL	"1": Activate the pressure correction
pr_pa	BOOL	"1": PRES is an absolute pressure "0": PRES is a relative pressure
pu	REAL	Value, which in the used pressure unit 1 displays atmosphere
en_temp	BOOL	"1": Activate the temperature correction
tc_tf	BOOL	"1": TEMP will be printed out in Degree Fahrenheit "0": TEMP will be printed out in Degree Celsius
en_sqrt	BOOL	"1": Calculation with Square Root

Detailed description

Calculation of the constant k

The constant k can be calculated because of a work point reference, with which the mass flow (MF_REF), the differential pressure (IN_REF), the absolute pressure (P_REF) and the absolute temperature (T_REF) are recognized.

When the input IN is **a Differential pressure** the equation says as follows:

$$k = MF\_REF \times \sqrt{\frac{T\_REF}{P\_REF \times IN\_REF}}$$

When the input IN is **no Differential pressure** the equation says as follows:

$$k = MF\_REF$$

Specification of the calculation

With the calculation, a simple multiplication is entered: $OUT = k \times IN$. In order to achieve pressure or temperature compensation, the parameters en_pres or en_temp must be set to 1. The square route is also only active when en_sqrt = 1.

When one of the parameters en_sqrt, en_pres, en_temp remains at 0, the calculation of the constant k must be adjusted to correspond (Delete the square route, replace from P_REF or T_REF through 1)

Temperature unit

The temperature TEMP can be printed out in Degree Celsius or Degree Fahrenheit, depending on the value of the parameter tc_tf :

tc_tf	Temperature unit from TEMP
0	Degree Celsius Calculation of the absolute temperature TA: $TA(°K) = TEMP + 273$
1	Degree Fahrenheit Calculation of the absolute temperature TA: $TA(°K) = \frac{5}{9} \times (TEMP - 32) + 273$

Pressure unit

The pressure PRES can be printed out in any unit, as absolute or relative pressure, according to the value of the parameter pr_pa.

pr_pa	Pressure unit from PRES
0	Relative pressure Parameter pu in the used unit 1 atmosphere, must conform Calculation of absolute pressure: PA = PRES + pu
1	Absolute pressure: PA = PRES

Runtime error

Status word

The bits of the status words have the following meaning:

Bit	Meaning
Bit 0 = 1	Error in a calculation in floating point values
Bit 1 = 1	Recording of an invalid value of a floating point value input
Bit 2 = 1	Division by zero with calculation in floating point values
Bit 3 = 1	Capacity overflow with calculation in floating point values
Bit 4 = 1	One of the following sizes is negative: IN, pu, PA, TA. For calculation, the function block uses the value 0.

Error message

In the following cases an Error will be recorded:
- At one of the floating point inputs an invalid value will be recorded
- Division by zero with calculation in floating point values
- Capacity overflow with calculation in floating point values

The output OUT will not be altered.

Warning

A warning is given if the parameter pu is negative, in this case with the calculation the block can use the value 0 in place of the defective value pu.

MS: Manual control of an output

26

Overview

At a glance

This chapter describes the MS block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This Function block serves as the control of a numerical output, which can be switched off via the function block PWM1 (see *PWM1: Pulse width modulation, p. 409*) controlled analog output, server motor or controlling element. The control can appear via server dialog or direct via the SPS-Software.

In general a control-function block serves as the control of a digital output. The MS-block should then be used, if the control output should be uncoupled from the control of the analog output.

EN and ENO can be projected as additional parameters.

Application possibilities

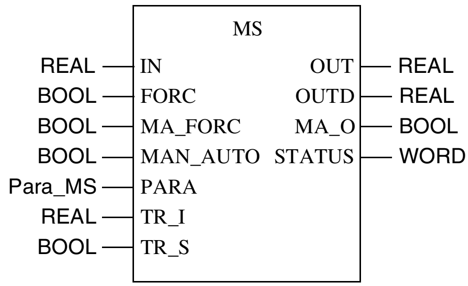
The function block will mainly be used with the following applications:

- For the control of an analog output, which is not controlled via a servo loop (open loop).
- Servo loops, with which the control output and the user controlled output have inserted a processing operation.
- With scanning of the output controlled controller, if the scanning period exceeds 1 to 2 seconds.
- With control of a server motor: the function block MS is in this case the controller block in order to insert the server motor.

Representation

Symbol

Block representation



Parameter description MS

Block parameter description

Parameter	Data type	Meaning
IN	REAL	Manipulated variable used in automatic mode
FORC	BOOL	"1": The mode manual/automatic will be entered via MA_FORC "0": The mode manual/automatic will be entered via MAN_AUTO
MA_FORC	BOOL	Mode manual/automatic (if FORC = 1) "1": Automatic operating mode "0": Manual mode
MAN_AUTO	BOOL	Mode manual/automatic (if FORC = 0) "1": Automatic operating mode "0": Manual mode
PARA	Para_MS	Parameter
TR_I	REAL	Initialization input
TR_S	BOOL	Initialization command
OUT	REAL	Absolute output
OUTD	REAL	Incremental output: Difference between the present output and the output of the previous execution
MA_O	BOOL	Current mode of the function block (0: Manual, 1: Automatic)
STATUS	WORD	Status word

Parameter description Para_MS

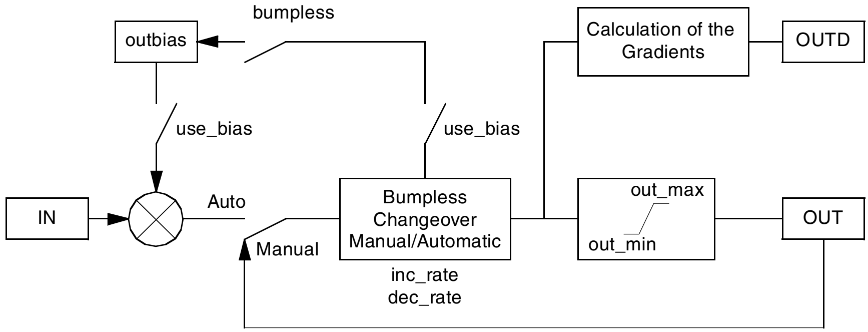
Data structure description

Element	Data type	Meaning
out_min	REAL	lower limit value of the output
out_max	REAL	upper limit value of the output
inc_rate	REAL	Increasing ramp at the changeover manual/ automatic (units per second)
dec_rate	REAL	Decreasing ramp at the changeover manual/ automatic (units per second)
outbias	REAL	Value of the bias
use_bias	BOOL	"1": Enable the bias
bumpless	BOOL	"1": Settings of the bias with changeover manual/ automatic (bumpless)

Detailed description

Structure diagram

In the following diagram the structure of the function block is displayed:



Setting of the mode selection

The mode selection can be set depending on input FORC either via the SPS program or via a server dialog (surveillance device):

Input FORC	Set the operating mode
0	Setting through the input MAN_AUTO (via operating device): MAN_AUTO= 1: Automatic mode MAN_AUTO= 0: Manual mode In this case the input MA_FORC is ineffective.
1	Setting through the input MA_FORC (via SPS-program): MA_FORC = 1: Automatic mode MA_FORC = 0: Manual mode In this case the input MAN_AUTO is ineffective.

The output MA_O always indicates the current operating mode of the function block.

Characteristics of the output OUT

The following characteristics apply to the output OUT:

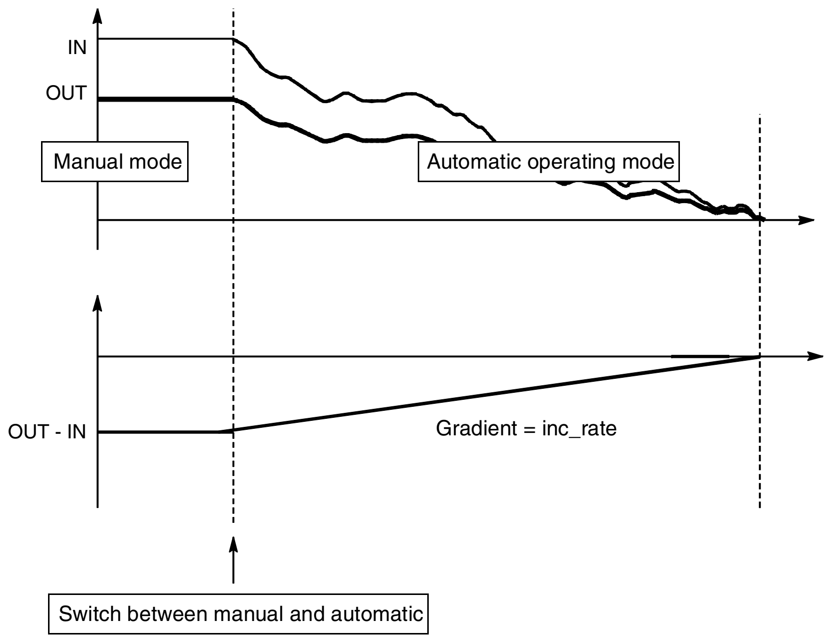
- Automatic mode: The output OUT is a copy of the input IN.
  In this operating mode, the output OUT can be assigned an OUTBIAS value (set _bias to 1). OUT calculates as follows: OUT = IN + outbias.
- Manual mode: The function block does not set the output, the server can directly change the value that is the connected variable at the output OUT.
- The output OUT is principally limited to an area between out_min and out_max. When the value calculated by the function block (or entered by the server in manual mode) exceeds one of these limit values, the value of OUT will be cut (to out_min or out_max). The incremental output OUTD on the other hand, never takes this cut into consideration.

Switch between manual and automatic

The switch manual/automatic at output appears bumpless, as the value of IN is not suddenly led to the output.

The output OUT gets closer to input IN ramps with positive (inc_rate) or negative increase (dec_rate):
- inc_rate applies when IN is larger than OUT at the time of the changeover
- dec_rate applies when IN is smaller than OUT at the time of the changeover

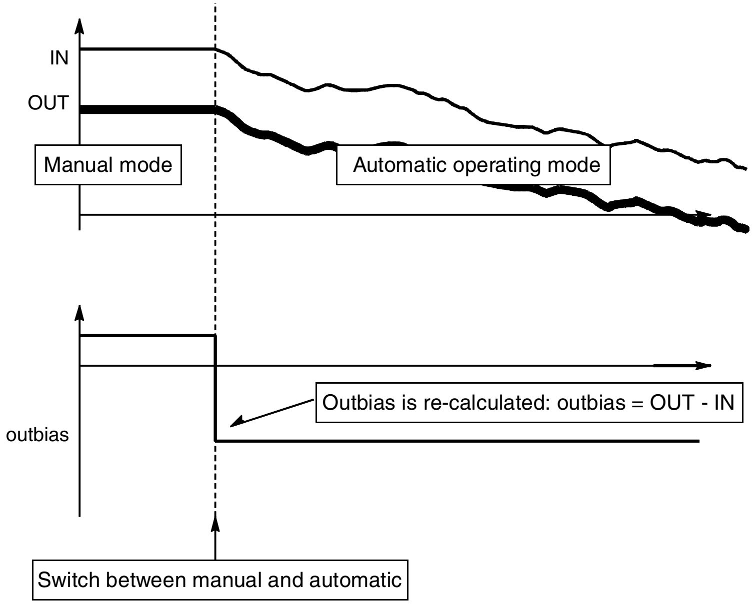
bumpless changeover



The bumpless changeover can be annulled with the increasing ramp, when inc_rate is set to 0. Just as with dec_rate = 0 the changeover is with decreasing ramp with bumps. In both cases the input IN will travel immediately to output OUT when changed over to automatic mode.

When the parameter outbias (use_bias = 1) is used, a bumpless changeover manual/automatic can be achieved without change of the output, when the parameter is set to 1. In this case the parameter outbias will be recalculated by the block to compensate the difference between the input IN and the output OUT.

Bumpless changeover with the parameter Outbias



The bumpless changeover manual/automatic is advisable when the input of the function block is not connected to any controller or to a controller output without integral component.

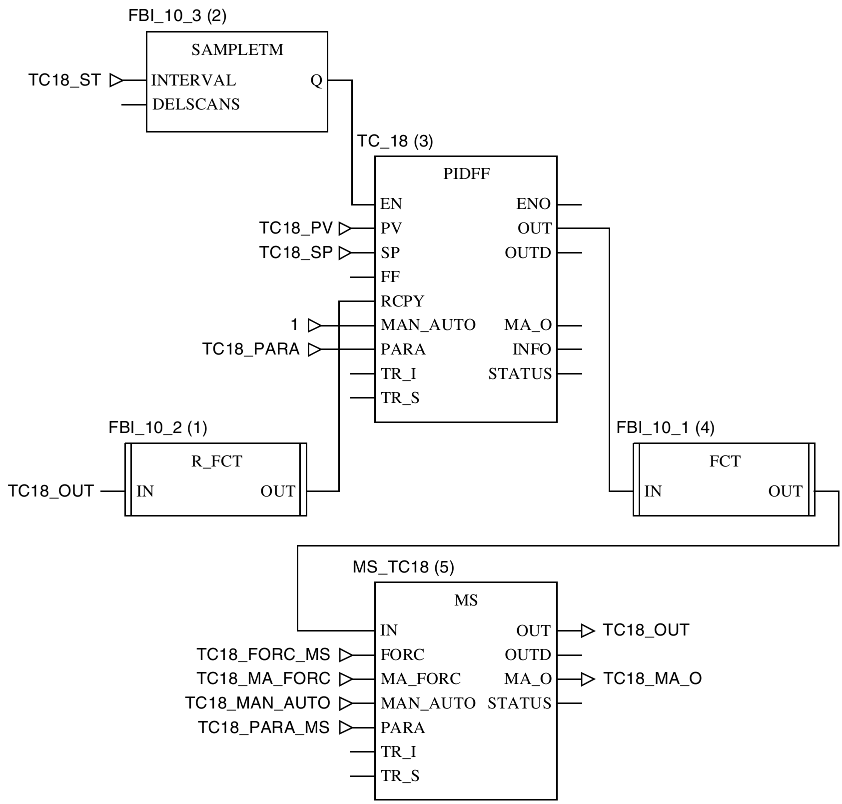
Example

Example

In this example the output of the control block and the output controlled by the server will insert a processing operation (through the DFB FCT).

In order to guarantee a bumpless changeover between the modes manual/ automatic, the reversed processing operation (R_FCT) will be assigned to the output of the MS function block and the result led back to the control input RCPY, which remained in automatic mode (MAN_AUTO = 1).

Display of the function plans



Runtime error

Status word

The bits of the status words have the following meaning:

Bit	Meaning
Bit 0 = 1	Error in a calculation in floating point values
Bit 1 = 1	Invalid value recorded at one of the floating point value inputs
Bit 2 = 1	Division by zero with calculation in floating point values
Bit 3 = 1	Capacity overflow with calculation in floating point values
Bit 4 = 1	The following error will be shown: ● One of the following sizes is negative: inc_rate, dec_rate. For calculation, the function block uses the value 0. ● The parameter Outbias lies out of the area $[(out\_min - out\_max), (out\_max - out\_min)]$. In this case the function block uses a cut value: $(out\_min - out\_max)$ and/or. $(out\_max - out\_min)$.
Bit 5 = 1	The output OUT has reached the lower limit value out_min (see Note)
Bit 6 = 1	The output OUT has reached the upper limit value out_max (see Note)

Note

Note: In manual mode these bits stay at 1 for only one program cycle. When the user enters a value for OUT which exceeds one of the limit values, the function block sets the Bit 5 or 6 to 1 and cuts them from the user entered value. With the following execution of the function block the value of OUT no longer lies outside the area and the Bits 5 and 6 are set to 0 again.

Error message

An error appears if a non floating point value is inputted or if there is a problem with a floating point calculation. In this case the outputs OUT, OUTD and MA_O remain unchanged.

Warning

In the following cases a warning is given:
- The parameter inc_rate is negative: in this case the function block uses the value 0 in place of the faulty value from inc_rate.
- The parameter dec_rate is negative: in this case the function block uses the value 0 in place of the faulty value from dec_rate.
- The parameter outbias lies outside the area [(out_min –out_max), (out_max – out_min)]. In this case for calculating the value the function block uses (out_min – out_max) and/or (out_max – out_min).

MULDIV_W: Multiplication/ Division

27

Overview

At a glance

This chapter describes the MULDIV_W block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block MULDIV_W carries out a weighted multiplication/division from 3 numerical input variables.

EN and ENO can be projected as an additional parameter.

Equation

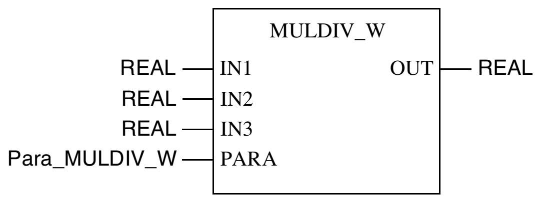
The equation says:

$$OUT = \frac{k \times (IN1 + c1) \times (IN2 + c2)}{IN3 + c3} + c4$$

Representation

Symbol

Block representation



Parameter description MULDIV_W

Block parameter description

Parameter	Data type	Meaning
IN1 to IN3	REAL	Numerical variables to be processed
PARA	Para_MULDIV_W	Parameter
OUT	REAL	Result of the calculation

Parameter description PARA_ MULDIV_W

Data structure description

Element	Data type	Meaning
k, c1 to c4	REAL	Calculation coefficients

Runtime error

Error message

This error will be signaled if a non floating point value is inputted or if there is a problem with a floating point calculation. In general, the output OUT keeps its previous value, apart from with a division by 0, where the value corresponds to INF depending on which sign the counter uses.

PCON2: Two point controller

28

Overview

At a glance

This chapter describes the PCON2 block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block forms a two-point controller, which maintains PID-similar behavior through two dynamic feedback paths.

EN and ENO can be projected as additional parameters.

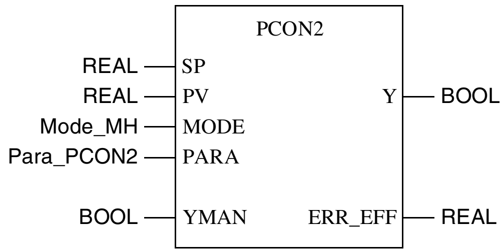
Properties

The function block contains the following properties:
- Operating mode, Manual, Halt, Automatic
- two internal feedback paths (delay 1. order)

Presentation

Symbol

Block display:



Parameter description PCON2

Block parameter description

Parameter	Data type	Meaning
SP	REAL	Setpoint input
PV	REAL	Process value input
MODE	Mode_MH	Operating mode
PARA	Para_PCON2	Parameter
YMAN	BOOL	"1" = Manual value for ERR_EFF
Y	BOOL	"1" = Output manipulated variable
ERR_EFF	REAL	Effective switch value

Parameter description Mode_MH

Data structure description

Element	Data type	Meaning
man	BOOL	"1" = Manual mode
halt	BOOL	"1" = Halt mode

Parameter description Para_PCON2

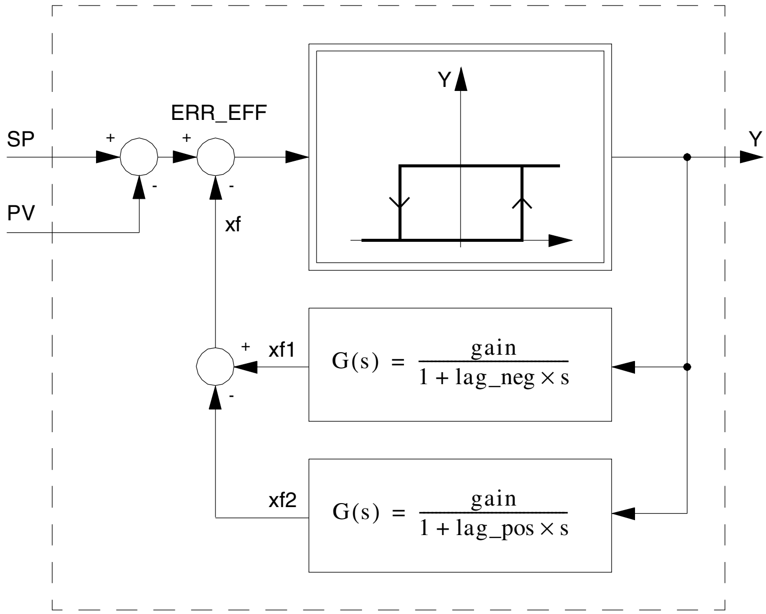
Data structure description

Element	Data type	Meaning
gain	REAL	Reset boost
lag_neg	TIME	Time constants of the quick reset
lag_pos	TIME	Time constants of the slow reset
hys	REAL	Hysteresis from two point switch
xf_man	REAL	Reset value of the reset in % (0 – 100)

Detailed description

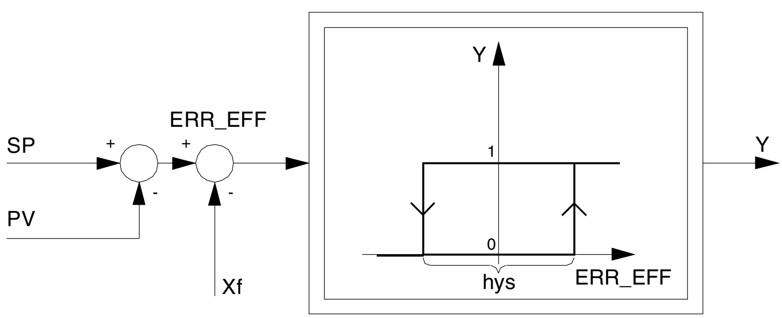
Structure of the controller

Structure of the two-point controller:



Principle of the two-point controller

The actual two-point controller will have 2 dynamic feedback paths (PT1-element) added. Through appropriate selection of the time constants of the reset-element, the two-point controller maintains dynamic behavior that corresponds to the behavior of a PID controller.



Reset

The revert- parameter set, made up of the revert boost gain and the revert time constant lag_neg and lag_pos, allows universal usage of the two point controller.

The following table provides more exact information about it:

Revert	lag_neg	lag_pos
2-Point-Behavior (without revert)	= 0	= 0
negative revert	> 0	= 0
negative + positive revert	> 0	> lag_neg
Warning, regeneration (neg. feedback with lag_pos)	= 0	> 0
Warning, regeneration (pos. Feedback disabled)	> lag_pos	> 0

Select revert-boost gain is greater than zero!

Enter xf_man (meaning 0% to 100%) values between 0 and 100!

Hysteresis

The parameter hys indicates the connector hysteresis, i.e. the value that the effective switch value ERR_EFF outgoing from control point hys/2 must be reduced by, before the output Y is reset to"0". The dependence of the output Y depending of the effective switch value ERR_EFF and the Parameter hys, becomes clear in the picture *Principle of the two-point controller, p. 232* The value of the hys parameter is typically set to 1% of the maximum control area [max. (SP – PV].

Operating mode

There are three operating mode, which are selected via the elements man and halt:

Operating mode	man	halt	Meaning
Automatic	0	0	The Function block will be handled as described above.
Manual	1	1 or 0	The output Y are set to the value YMAN. xfl and xf2 are calculated using the following formula: xf1 = xf_man * gain /100 xf2 = xf_man * gain /100
Halt	0	1	The output Y will be held at the last value. xf1 and xf2 are set to gain * Y.

Runtime error

Warning

In the following cases a Warning will be given:

Causes	Behavior of the controller
lag_neg = 0 and lag_pos > 0	The controller works as if it had only a negative feedback lag_pos.
lag_pos < lag_neg > 0	The controller works as if it had only a negative feedback with the time constant lag_neg.
xf_man < 0 or xf_man > 100	The controller works without internal feedback paths.

PCON3: Three point controller

29

Overview

At a glance

This chapter describes the PCON3 block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block forms a three-point controller, which maintains PID-similar behavior through two dynamic feedback paths.

EN and ENO can be projected as additional parameters.

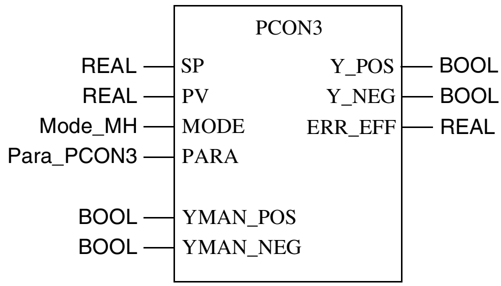
Properties

The function block PCON3 contains the following properties:
- Operating mode, Manual, Halt, Automatic
- two internal feedback paths (delay of the 1st order)

Presentation

Symbol

Block display:



Parameter description PCON3

Block parameter description

Parameter	Data type	Meaning
SP	REAL	Setpoint input
PV	REAL	Process value input
MODE	Mode_MH	Operating mode
PARA	Para_PCON3	Parameter
YMAN_POS	BOOL	Manual manipulation for Y_POS
YMAN_NEG	BOOL	Manual manipulation for Y_NEG
Y_POS	BOOL	"1" = positive manipulated variable at output ERR_EFF
Y_NEG	BOOL	"1" = negative manipulated variable at output ERR_EFF
ERR_EFF	REAL	Effective switch value

Parameter description Mode_MH

Data structure description

Element	Data type	Meaning
man	BOOL	"1" = Manual mode
halt	BOOL	"1" = Halt mode

Parameter description Para_PCON3

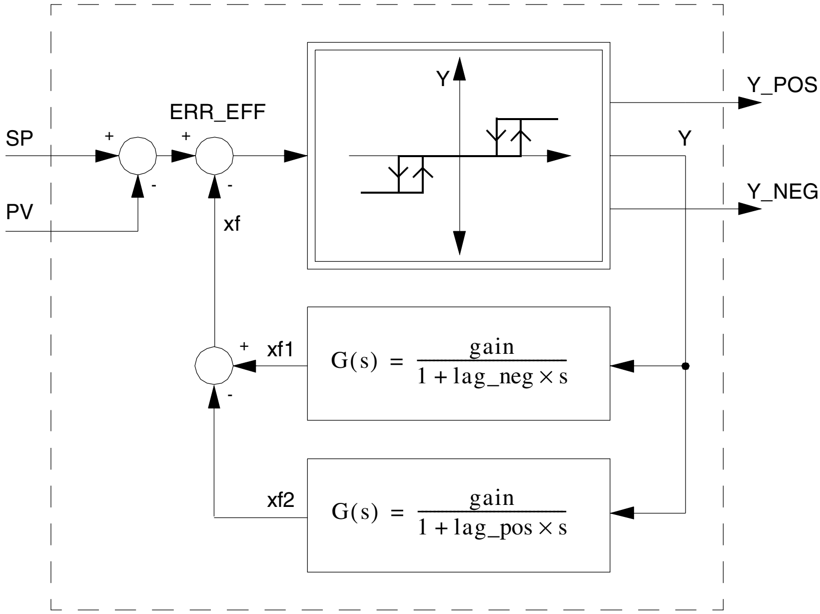
Data structure description

Element	Data type	Meaning
gain	REAL	Reset-boost (reset-parameter-sequence)
lag_neg	TIME	Time constant of the quick reset (reset-parameter-sequence)
lag_pos	TIME	Time constant of the slow reset (reset-parameter-sequence)
hys	REAL	Hysteresis from three point switch
db	REAL	Insensitivity zone
xf_man	REAL	Reset value of the reset in % (0 – 100)

Detail description

Structure of the controller

Structure of the three-point controller:

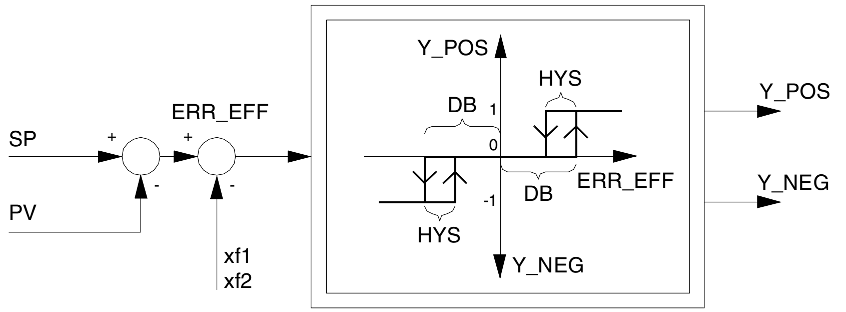


The following applies:

If…	Then…
Y = 1	Y_POS = 1 Y_NEG = 0
Y = 0	Y_POS = 0 Y_NEG = 0
Y = -1	Y_POS = 0 Y_NEG = 1

Principle of the three-point controller

The actual three-point controller will have 2 dynamic feedback paths (PT1-elements) added. Through appropriate selection of the time constants of the reset-element, the three-point controller maintains dynamic behavior that corresponds to the behavior of a PID controller.



Feedback

The function block has a parameter sequence for the internal feedback paths, comprised of the reset-boost gain and the reset time constant lag_neg and lag_pos.

The following table provides more exact information about it:

Feedback	lag_neg	lag_pos
3-Point-Behavior (without revert)	= 0	= 0
negative revert	> 0	= 0
negative + positive revert	> 0	> lag_neg
Warning, regeneration (neg. feedback with lag_pos)	= 0	> 0
Warning, regeneration (pos. Feedback disabled)	> lag_pos	> 0

The parameter gain must be > 0

The amount will be resolved from the Hysterisis hys and the no-sensitivity zone db!

For xf_man (meaning -100 to 100%) values between -100 and 100 are to be entered!

No-sensitivity zone

The parameter db fixes the connection point for the outputs Y_POS and Y_NEG. If the effective switch value ERR_EFF is positive and is greater than db, then the output Y_POS will switch from "0" to "1". If the effective switch value ERR_EFF is negative and is smaller than db, then the output Y_NEG will switch from "0" to "1". The value of the db parameter is typically set to 1% of the maximum control area (max. SP – PV).

Hysteresis

The parameter hys indicates the connector-hysteresis, i.e. the value which the effective switch value ERR_EFF outgoing from control point db must be reduced by, before the output Y_POS (Y_NEG) is reset to "0". The connection between Y_POS and Y_NEG depending on effective switch value ERR_EFF and the parameters db and hys Is illustrated in the image *Principle of the three-point controller, p. 240*. The value of the hys parameter is typically set to 0.5% of the maximum control area (max. SP – PV).

Operating mode

There are three operating modes, which are selected via the elements man and halt:

Operating mode	man	halt	Meaning
Automatic	0	0	The Function block will be handled as described above.
Manual	1	0 or 1	The outputs Y_POS and Y_NEG are set to the values YMAN_POS and YMAN_NEG. In this case, the built in priority-logic – Y_NEG is dominant over Y_POS, which prohibits both outputs from being set simultaneously. xf1 and xf2 are calculated using the following formula: xf1 = xf_man * gain /100 xf2 = xf_man * gain /100
Halt	0	1	In Halt mode, both outputs Y_POS and Y_NEG will be held at the last value. xf1 and xf2 are set to gain * Y.

Runtime error

Error message

With hys > 2 * db, an Error Message appears.

Warning

In the following cases a Warning will be given:

Causes	Behavior of the controller
lag_neg = 0 and lag_pos > 0	The controller works as if it had only a negative feedback with the constant lag_pos.
lag_pos < lag_neg > 0	The controller works as if it had only a negative feedback with the time constant lag_neg.
xf_man < 0 or xf_man > 100	The controller works without internal feedback paths.

PD_or_PI: Structure changeover PD/PI controller

30

Overview

At a glance

This chapter describes the PD_or_PI block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block PD_or_PI can work equally well as either PD-Controller or PI-Controller. Depending on the system deviation (SP – PV) and a specified switch value, trig_err will automatically perform a structural changeover from PD- to PI-Controller and vice-versa from PI- to PD-Controller.

This EFB is particularly suitable for starting control purposes. When the process is runup the controller reacts as a P(D) controller, whereby the controlled variable is to reach the adjusted reference variable value as fast as possible. Shortly before the given setpoint value is reached, the control algorithm is switched over and an I component makes sure that the remaining control deviation fades out.

EN and ENO can be projected as additional parameters.

Properties

The function block contains the following properties:
- PI controller with independent gain, ti-adjust
- PI controller with independent gain, td-adjust
- Limited manipulated variable in automatic mode
- Antiwindup reset in PI operation
- definable delay of the D-component
- Operating mode, Manual, Halt, Automatic
- smooth switch between manual and automatic
- Automatic bumpless changeover from PDPI operation

The PI controller transfer function

The PI controller transfer function is:

$$G(s) = gain\_i \times \left(1 + \frac{1}{ti \times s}\right)$$

The PD controller transfer function

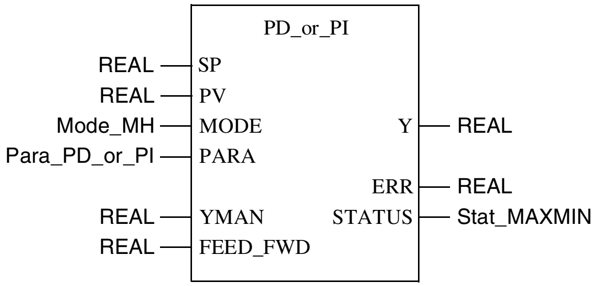
The PD controller transfer function is:

$$G(s) = gain\_d \times \left(1 + \frac{td \times s}{1 + td\_lag \times s}\right)$$

Presentation

Symbol

Block display:



Parameter description PD_or_PI

Block parameter description

Parameter	Data type	Meaning
SP	REAL	Setpoint input (reference variable)
PV	REAL	Process variable (controlled variable)
MODE	Mode_MH	Operating mode
PARA	Para_PD_or_PI	Parameter
YMAN	REAL	Manual manipulated variable
FEED_FWD	REAL	Disturbance variable
Y	REAL	Manipulated variable
ERR	REAL	System deviation
STATUS	Stat_MAXMIN	Output status

Parameter description Mode_MH

Data structure description

Element	Data type	Meaning
man	BOOL	"1": Manual mode
halt	BOOL	"1": Halt operating mode

Parameter description Para_PD_or_PI

Data structure description

Element	Data type	Meaning
trig_err	REAL	Changeover switching value for PDPI controller
gain_d	REAL	PD controller proportional action coefficient (gain)
td	TIME	PD controller rate time
td_lag	TIME	Delay of the PD controller rate time
gain_i	REAL	PI controller proportional action coefficient (gain)
ti	TIME	PI controller reset time
ymax	REAL	Upper limit
ymin	REAL	Lower limit

Parameter description Stat_MAXMIN

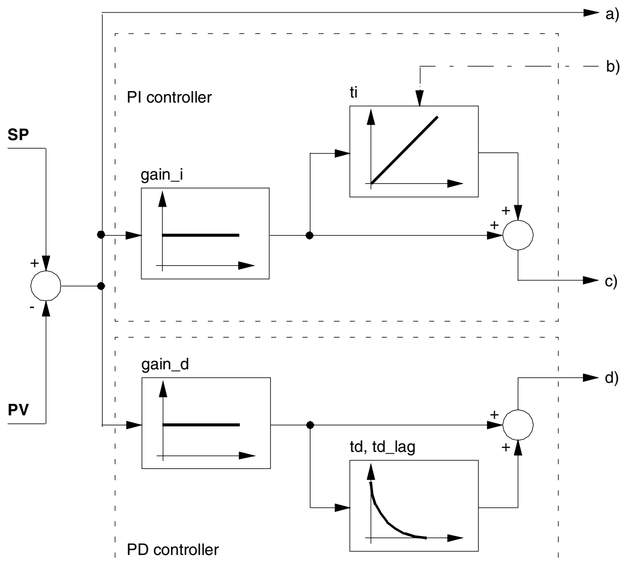
Data structure description

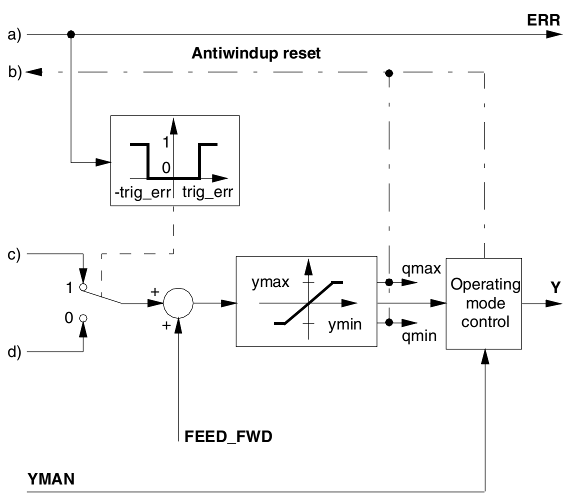
Element	Data type	Meaning
qmax	BOOL	"1" = Y reached upper limit
qmin	BOOL	"1" = Y reached lower limit

PD_or_PI function block structure diagram

Structure diagram

There follows now the structure diagram of the PD_or_PI block:





Detailed description

Determination of switching value

The parameterization of the function block begins with the determination of switching value trig_err. This parameter fixes the automatic changeover point of the function block from PDPI operation.

When the absolute value of system deviation ERR = SP - PV is smaller than the switching value trig_err, the controller switches automatically from PD operation into PI operation.

When the absolute value of system deviation is larger than the switching value trig_err, the controller switches automatically form PI operation into PD operation.

It then follows that:
- PD controller: ERR > trig_err
- PI controller: ERR ≤ trig_err

Each controller type is linked to a parameter set, which must be projected as well. The control algorithm changeover is practically a switch from one parameter set to the other. The changeover is bumpless.

PD controller

PD controller parameterization is accomplished by projection of the proportional action coefficient gain_d and rate time td.

For PD controller operation the D component is delayed by the time constant value td_lag. The td/td_lag ratio is termed the differential gain, and is generally selected between 3 and 10. The D component directly determined by the system deviation ERR, such that for reference variable fluctuations (variations at input SP) a jump attributed to the D component is produced.

The D component can be disabled by setting td = 0.

PI controller

PI controller parameterization is accomplished by projection of the proportional action coefficient gain_i and reset time ti.

In general during run-up with the PD algorithm, the proportional action coefficient is set considerably higher, than in the practically stationary operation in PI algorithm thereafter. This circumstance is conceded to by the designation of two independent proportional action coefficients.

The I component can be disabled by setting ti = 0.

Limiting of manipulated variable

The limits ymax and ymin retain the manipulated variable within the prescribed range.

It therefore holds that: ymin ≤ Y ≤ ymax

The outputs qmax and qmin signal that the manipulated variable has reached a limit, and thus been capped:
- QMAX = 1 if Y ≥ YMAX
- QMIN = 1 if Y ≤ YMIN

Upper limit ymax, limiting the manipulated variable, is to be set higher than lower limit ymin.

Antiwindup Reset

Should limiting of the manipulated variable take place while the PI control algorithm is active, the antiwindup reset should ensure that the I component "cannot go berserk". Antiwindup measures are taken only for I component values other than 0. Antiwindup limits are identical to those for the manipulated variable.

The antiwindup-reset measures correct the I component such that:
- YI ≥ ymin - gain_i * (SP - PV) - FEED_FWD
- YI ≤ ymax - gain_i * (SP - PV) - FEED_FWD

Operating mode

There are three operating modes selectable via the man and halt parameter inputs:

Operating mode	man	halt	Meaning
Automatic	0	0	The manipulated variable output Y is determined through the discrete PI or PD closed-loop control algorithms, based on the controlled variable PV and reference variable SP. The manipulated variable is limited by ymax and ymin. The controller output limits also serve as limits for the antiwindup reset.
Manual	1	0 or 1	The manual manipulated value YMAN is passed on directly to the manipulated variable Y. The manipulated variable is limited by ymax and ymin. Internal variables will be so manipulated, that the controller changeover from manual to automatic can be bumpless.
Halt	0	1	The manipulated variable remains unchanged, the block does not influence the manipulated variable Y. Internal variables will be manipulated in such a manner that the controller can be driven smoothly from it's current position. Manipulated variable limits and antiwindup measures are as those in automatic mode Halt operating mode is also useful for allowing an external operator device to adjust the manipulated variable Y; the internal components in the controller are then tracked correctly.

Detailed formulas

Explanation of the formula sizes

Significance of the size in the following formulas:

Size	Meaning
dt	Present sample time
ERR	System deviation
$ERR_{(old)}$	System deviation value from the previous sampling step
FEED_FWD	Disturbance variable
Y	Current output (halt operating mode) or YMAN (manual mode)
YD	D component
$YD_{(old)}$	Value of the D-component from the previous sampling step
YI	I component
$YI_{(old)}$	Value of the I component from the previous sampling step
YP	P component

System deviation

The system deviation will be determined as follows:

$$ERR = SP - PV$$

Manipulated variable

The manipulated variable consists of different partial sizes which are dependent on the operating mode.

$$Y = YP + YI + YD + FEED\_FWD$$

After the summation of the components a manipulated variable limiting takes place at the output of the sub controller, which means:

$$ymin \le Y \le ymax$$

Overview to calculate the control components

Following this an overview on the different calculations of the control components in relation to the elements trig_err can be found:

Controller type	Controller components
PI-Controller (ERR ≤ trig_err)	YP and YD for manual, halt and automatic modes YI for automatic operating mode YI for manual and halt operating mode
PD-Controller (ERR > trig_err)	YP and YI for manual, halt and automatic modes YD for automatic mode YD for manual and halt operating mode

PI controller: YP and YD for all operating mode

YP and YD for manual, halt, automatic and cascade modes are located as follows:

$$YP = gain\_i \times ERR$$

$$YD = 0$$

PI controller: I component for automatic mode

YI for automatic mode is determined as follows (ti > 0):

$$YI = YI_{(old)} + gain\_i \times \frac{dt}{ti} \times \frac{ERR + ERR_{(old)}}{2}$$

The I-component is formed according to the trapezoid rule.

PI controller: I component YI for manual and halt modes

YI for manual and halt are located as follows

$$YI = Y - (YP - FEED\_FWD)$$

PD controller: YP and YI for all modes

YP and YI for manual, halt, and automatic modes are determined as follows

$$YP = gain\_d \times ERR$$

$$YI = 0$$

PD controller: D component for automatic mode

YD for automatic mode is determined as follows:

$$YD = \frac{YD_{(old)} \times td\_lag + td \times gain\_d \times (ERR - ERR_{(old)})}{dt + dt\_lag}$$

PD controller: D component for manual and halt operating mode

YD for manual, halt and automatic modes are determined as follows:

$$YD = 0$$

Runtime error

Error message

There is an Error message, if

- an unauthorized floating point number is placed at the input PV
- or ymax < is ymin

PDM: Pulse duration modulation

31

Overview

At a glance

This chapter describes the PDM block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Using the block

Actuators are driven not only by analog quantities, but also through binary actuating signals. The conversion of analog values into binary output signals is achieved for example, through pulse width modulation (PWM) or pulse duration modulation (PDM).

The actuator adjusted average energy (actuator energy) should be in accord with the modulation block's analog input value (X).

Function description

The Function block PDM is used to convert analog values into digital output signals.

In the function block PD a "1" signal of fixed persistence is output within a variable cycle time proportional to the analog value X. The adjusted average energy corresponds to the quotient of the fixed duty cycle t_on and the variable cycle period.

In order that the adjusted average energy also corresponds to the analog input variable X, the following must apply:

$$T_{period} \sim \frac{1}{X}$$

EN and ENO can be configured as additional parameters.

General information about the actuator drive

In general, the binary actuator drive is performed by two Boolean signals Y_POS and Y_NEG. On a motor the output Y_POS corresponds to the signal "clockwise rotation" and the output Y_NEG the signal "counter-clockwise rotation". For an oven the outputs Y_POS and Y_NEG could be interpreted as corresponding to "heating" and "cooling".

Should the actuating drive in question be a motor, it is possible that to avoid overtravel for non-self-locking gearboxes, a brake pulse must be output after the engage signal.

In order to protect the power electronics, there must be a pause time t_pause after switching on t_on and before the brake pulse t_brake so as to avoid short circuits.

Formula

For correct operation the following rules should be observed:

$$t\_on + 2 \times t\_pause + t\_brake \geq \frac{pos\_}{neg\_} \times t\_min$$

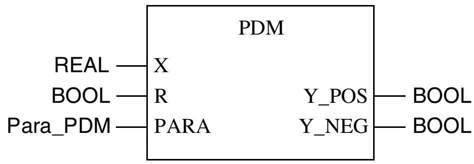
and

$$\frac{pos\_}{neg\_} \times t\_min < \frac{pos\_}{neg\_} \times t\_max$$

Representation

Symbol

Block representation



Parameter description PDM

Block parameter description

Parameter	Data type	Meaning
X	REAL	Input variable
R	BOOL	Reset mode
PARA	Para_PDM	Parameter
Y_POS	BOOL	Positive X value output
Y_NEG	BOOL	Negative X value output

Parameter description Para_PDM

Data structure description

Element	Data type	Meaning
t_on	TIME	Pulse duration (in s)
t_pause	TIME	Pause time (in s)
t_brake	TIME	Braking time (in s)
pos_up_x	REAL	Upper limit for positive X
pos_t_min	TIME	Minimum cycle time for Y_POS (where x = pos_up_x) (in s)
pos_lo_x	REAL	Lower limit for positive X
pos_t_max	TIME	Maximum cycle time for Y_POS (where x = pos_lo_x) (in s)
neg_up_x	REAL	Upper limit for negative X
neg_t_min	TIME	Minimum cycle time for Y_NEG (where x = -neg_up_x) (in s)
neg_lo_x	REAL	Lower limit for negative X
neg_t_max	TIME	Maximum cycle time for Y_NEG (where x = -neg_lo_x) (in s)

Detailed description

Block mode of operation

The pulse duration t_on determines the time span in which the output Y_POS resp. Y_NEG has "1" signal. For a positive input signal X the output Y_POS is set, on negative the output Y_NEG is set. Only one output can carry "1" signal. It is advisable to perform a freely definable pause time of t_pause = 10 or 20 ms between the actuating and brake pulses to protect the power electronics (hopefully preventing simultaneous firing of the antiparallel connected thyristors).

A possible brake pulse of duration time t_brake follows the output pulse duration after a pause time t_pause. Within the pause time both outputs carry "0" signal. During the braking time the output opposite that carrying the previous pulse goes to "1" signal. A pause time of t_pause = 20 ms (t_pause =0.02) corresponds to an interruption of the firing angle control for two half waves. That should guarantee a sufficiently large safety margin for the prevention of short-circuits resp. triggering of the suppressor circuitry as a consequence of antiparallel thyristors firing.

Thereafter follows a period in which both outputs carry "0" signal (wait timeout).

Period t_{period}

This wait timeout, together with the pulse, pause and brake times, all makeup a period t_{period}, which depending on lo_x and t_min, is calculated according to the following formulas:

Requirements	Equation	Explanation of formula variables
lo_x <> 0	$t_{period} = t_0 + \frac{K}{X}$	$K = (t\_max - t\_min) \times \frac{up\_x \times lo\_x}{up\_x - lo\_x}$ $t_0 = t\_max - \frac{K}{lo\_x}$
lo_x = 0 t_min > 0	$t_{period} = \frac{K}{X - X0}$	$X0 = \frac{t\_max \times lo\_x - t\_min \times up\_x}{t\_max - t\_min}$ $K = t\_min \times (up\_x - X0)$
lo_x = 0 t_min = 0	$t_{period} = t\_max \times \left(1 - \frac{X}{up\_x}\right)$	

The following holds for all three cases:

Assuming	lo_x	up_x	t_min	t_max
$X \geq \lvert pos\_lo\_x \rvert$	pos_lo_x	pos_up_x	pos_t_min	pos_t_max
$X \geq -\lvert neg\_lo\_x \rvert$	neg_lo_x	neg_up_x	neg_t_min	neg_t_max

Note: From the parameters up_x (-pos/-neg) and lo_x (-pos/-neg) only the (absolute) value is evaluated.

Cycle time

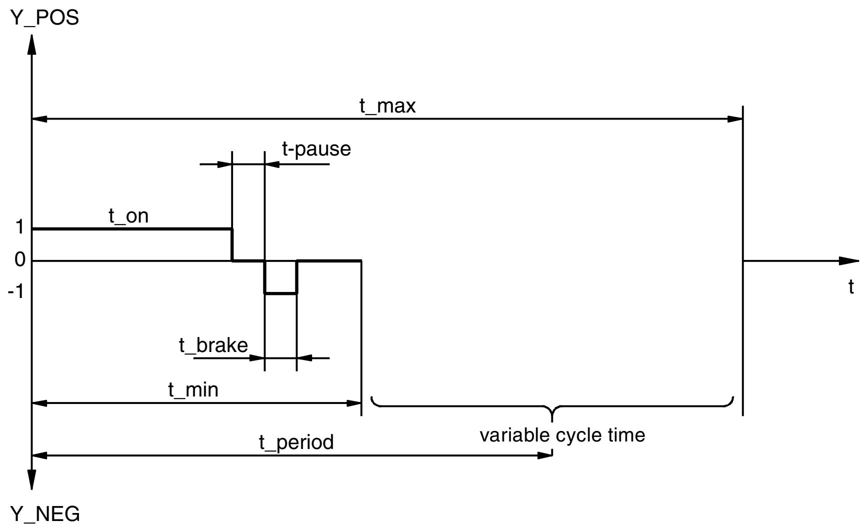
The parameter t_min _ for every output there is a separate value _ gives the minimum period, i.e. the time span, which passes from the beginning of one actuating pulse until the start of the next. This time span appears when input X goes beyond value up_x _ this time there is a separate value for each sign.

The parameter t_max places an upper limit on the maximum period. Should the input cross below the value pos_lo_x or neg_lo_x, the actuating pulse output is terminated until the until the input exceeds the value pos_lo_x or neg_lo_x again. The values pos_lo_x and neg_lo_x define what is in principle a dead zone, in which the function block outputs are not activated.

The parameters (pos_t_min, pos_up_x) and (pos_t_max, pos_lo_x) apply for positive X input signals, whereby output Y_POS is set. In the same way the parameters (neg_t_min, neg_up_x) and (neg_t_max, neg_lo_x) are valid for negative X input signals. Output Y_NEG is set.

Time ratios display

An overview of the ratios between times is shown in the following diagram:

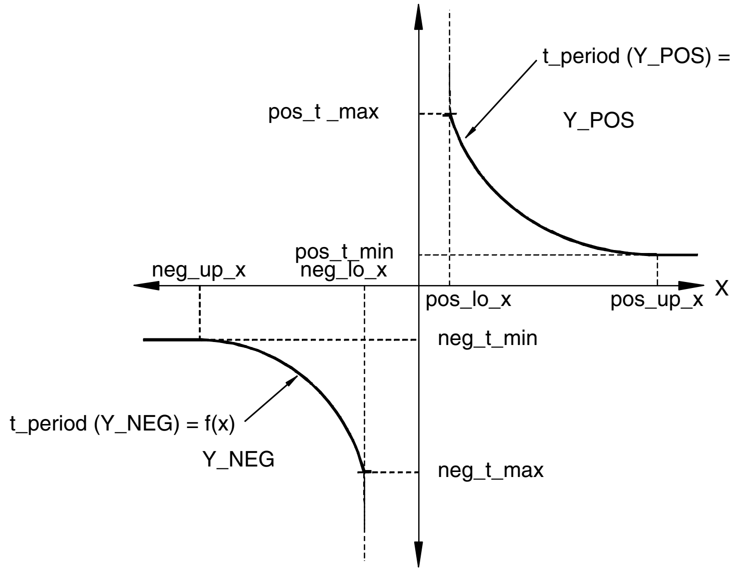


Time-span dependency

The time-span dependency from the input variable X, in which the output Y_POS (Y_NEG) carries 1-Signal, is displayed in the picture "*Output dependency on X, p. 261*" and the picture "*Output dependency on X (Special case), p. 261*".

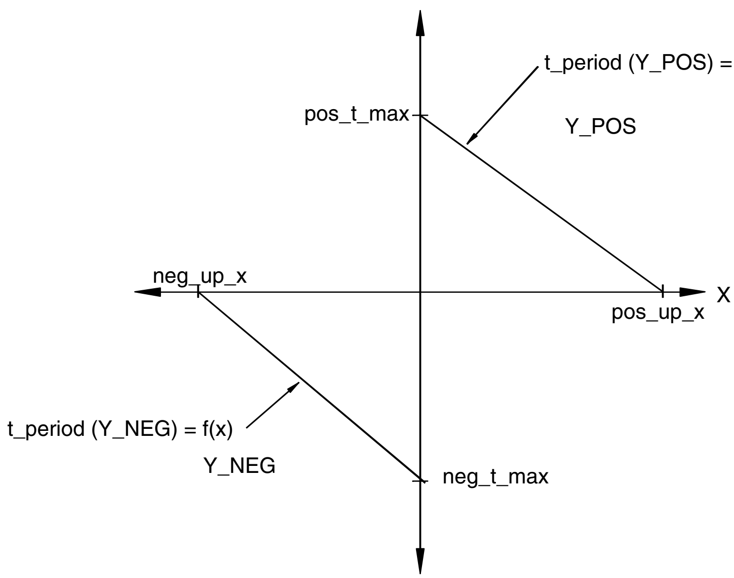
Output dependency on X

In the following picture the dependency of the output on X is shown:



Output dependency on X (Special case)

In the following picture the special case t_min = 0, lo_x = 0 is shown:



Operating mode

In reset mode R = "1", outputs Y_POS and Y_NEG are set to "0" signal. The internal time meters are also standardized, so that the function block begins the transfer to R=0 with the output of a new 1 signal on the associated output.

Boundary conditions

If the PDM function block is operated together with a PID controller, then the maximum period t_max should be so selected, that it corresponds to the PID controller's scan time. It is then guaranteed that every new actuating signal from the PID controller within the period time can be fully processed.

The PDM scan time should be in proportion with period vs. pulse time, Though this, the smallest possible actuating pulse is be determined.

The following ratio is recommended:

t_max/scan time (PDM) $\geq$ 10

Runtime error

Error message

An Error message appears, if
- |up_x| $\leq$ |lo_x|
- t_max $\leq$ t_min

EFB Descriptions (PI to Z)

III

Overview

Introduction

The EFB descriptions are arranged in alphabetical order.

Note: The number of inputs of some EFBs can be increased by vertically resizing the FFB symbol up to a maximum of 32. For information on which EFBs have this capability, please see the descriptions of the individual EFBs.

What's in this Part?

This part contains the following chapters:

PI: PI controller

32

Overview

At a glance

This chapter describes the PI block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block represents a simple PI controller.

A system deviation ERR is formed by the difference between the reference variable SP and the controlled variable PV. This deviation causes the manipulated variable Y to change.

EN and ENO can be configured as additional parameters.

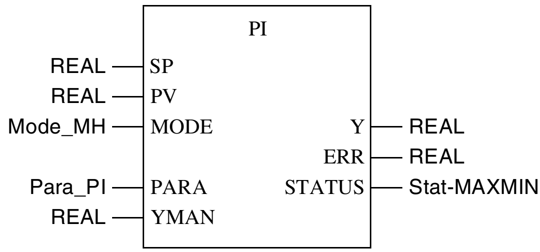
Properties

The function block has the following properties:
- Manual, Halt, Automatic modes
- bumpless manual/automatic mode changeover
- Manipulated variable limiting
- Antiwindup reset (only for an active I component)

Representation

Symbol

Block representation:



Parameter description PI

Block parameter description

Parameter	Data type	Meaning
SP	REAL	Setpoint input / reference variable
PV	REAL	Process variable / controlled variable
MODE	Mode_MH	Operating mode
PARA	Para_PI	Parameter
YMAN	REAL	Manual value
Y	REAL	Manipulated variable
ERR	REAL	System deviation
STATUS	Stat_MAXMIN	Y output status

Parameter description Mode_MH

Data structure description

Element	Data type	Meaning
Man	BOOL	"1": Manual mode
Halt	BOOL	"1": Halt mode

Parameter description Para_PI

Data structure description

Element	Data type	Meaning
gain	REAL	Proportional action coefficient (gain)
ti	TIME	Reset time
ymax	REAL	Upper limit
ymin	REAL	Lower limit

Parameter description Stat_MAXMIN

Data structure description

Element	Data type	Meaning
qmax	BOOL	"1" = Y has reached upper limit
qmin	BOOL	"1" = Y has reached lower limit

Formulae

Transfer function

The transfer function is:

$$G(s) = gain \times \left(1 + \frac{1}{ti \times s}\right)$$

Calculation formulae

The calculation formulae are:

$$YP = gain \times ERR$$

$$YI_{(new)} = YI_{(old)} + gain \times \frac{dt}{ti} \times \frac{ERR_{(new)} + ERR_{(old)}}{2}$$

Output signal Y

The output signal Y is then:

$$Y = YP + YI$$

The I component is formed according to the trapezoid rule.

Explanation of formula variables

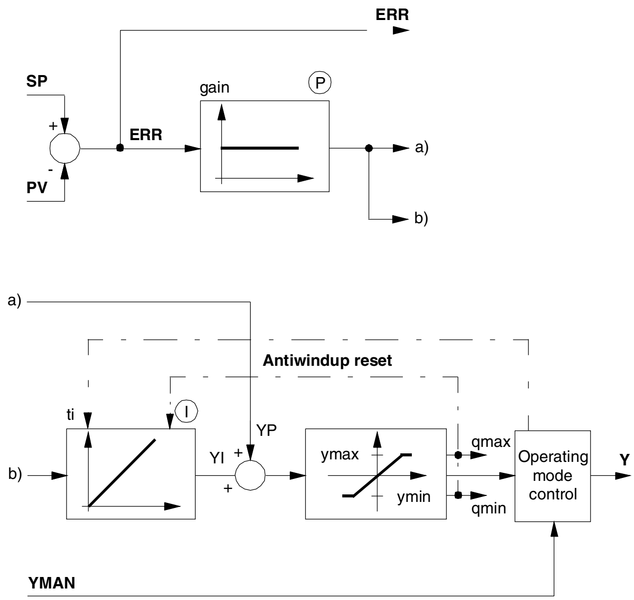
The meaning of the formula variables is given in the following table:

Variable	Meaning
dt	Current scan time
ERR	System deviation (SP - PV)
$ERR_{(old)}$	System deviation value from the previous sampling step
YI	I component
YP	P component

Parametering

Structure diagram

The following is the structure diagram of the PI controller:



Parametering

The structure of the PI controller is represented in the *Structure diagram, p. 270* above. The parametering of the function block takes place first of all for the elemental PI parameters: the proportional action coefficient gain and reset time ti.

The I component can be disabled by setting ti = 0.

The values ymax and ymin limit the upper and lower values of the output. Hence, ymin ≤ Y ≤ ymax.

The outputs qmax and qmin signal that the output has reached a limit, and thus been capped.

- qmax = 1 if Y ≥ ymax
- qmin = 1 when Y ≤ ymin

Manipulated variable limiting

After summation of the components a variable limiting takes place, so that: ymin ≤ Y ≤ ymax

Antiwindup Reset

Should limiting of the manipulated variable take place, the antiwindup reset should ensure that the integral component "cannot go berserk". Antiwindup measures are only taken if the controller I component is not switched off. Antiwindup limits are identical to those for the manipulated variable. The antiwindup reset measures correct the I component such that: ymin - YP ≤ YI ≤ ymax - YP

Operating modes

Selecting the operating modes

There are three operating modes, which are selected via the elements Man and Halt.

Operating mode	Man	Halt
Automatic	0	0
Manual	1	1 or 0
Halt	0	1

Automatic mode

In automatic mode the control output Y is determined through the closed-loop control based on the controlled variable PV and reference variable SP. The manipulated variable is limited by ymax and ymin. The manipulated variable limits also serve as limits for the Antiwindup reset.

The changeover from automatic to manual is normally not bumpless, since output Y can take on any value between ymax and ymin, and yet goes directly to YMAN at the changeover.

If the changeover from automatic to manual is to be bumpless in spite of these problems, there are two exemplary possibilities shown for a PID controller (see *Switching from automatic to manual, p. 302*).

Manual mode

In manual mode the manually manipulated value YMAN is passed on directly to the control output Y. But the manipulated variable is still limited by ymax and ymin. Internal variables will be manipulated in such a manner that the controller changeover from manual to automatic (with I component enabled) can be bumpless. The manipulated variable limits also serve as limits for the Antiwindup reset

Halt operating mode

In halt mode the control output remains unchanged; the function block does not influence the control output Y, i.e. Y = Y(old). Internal variables will be manipulated in such a manner that the component sum corresponds with the manipulated variable, thus allowing the controller to be driven smoothly from its current position. The manipulated variable limits also serve as limits for the Antiwindup reset. Halt mode is also useful in allowing an external operator device to adjust control output Y, whereby the controller's internal components are given the chance to continuously react to the external influence.

PI controller example

Example

The jump response of the PI controller is shown in the following Diagram (see *PI controller jump response, p. 273*) as an example.

In the first part of the figure the function block response to manual operating mode can be seen: The.output Y jumps to the YMAN value.

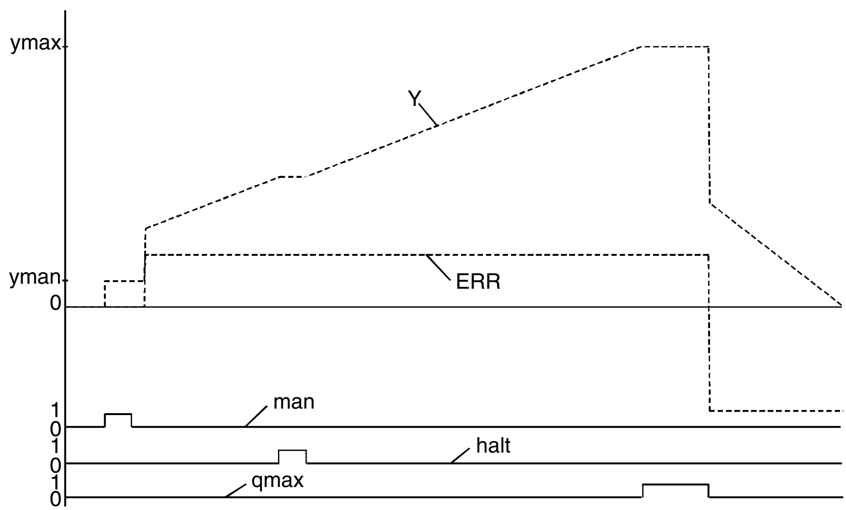
The second part of the diagram shows the reaction of the function block in automatic mode (MAN = 0 and HALT= 0) both with a positive ERR system deviation and with a negative ERR system deviation. For constant positive system deviation, Y ramps upward until the upper output limit is reached.

Y is then limited to the value ymax. Limiting at ymax being signaled in qmax. The system deviation then jumps to a negative value whose absolute value is greater than the previous positive value.

The input jumps to the value $\text{gain} \times (\text{ERR}_{(\text{new})} - \text{ERR}_{(\text{old})})$; through the P component, then there is a ramp decrease in Y. The absolute value of the gradient is greater than under the previous positive system deviation. This can be attributed to the now greater absolute value of the system deviation.

PI controller jump response

Presentation of the jump response of the PI controller



Runtime error

Error message

There is an Error message, if
- an unauthorized floating point number is placed at input YMAN or X,
- is ymax < ymin.

PI1: PI controller

33

Overview

At a glance

This chapter describes the PI1 block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block represents a simple PI controller.

A system deviation ERR is formed between the setpoint SP and the process value PV. This deviation brings about a change of the manipulated variable Y.

As additional parameters, EN and ENO can be projected.

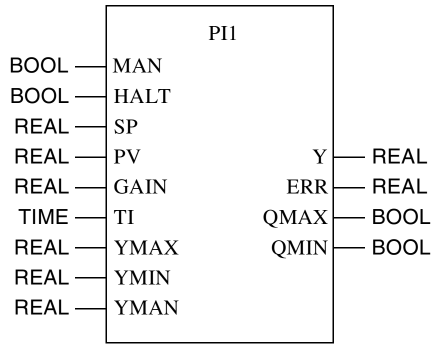
Properties

The function block has the following properties:
- Manual, Halt, Automatic operating modes
- bumpless changeover between manual and automatic
- Manipulated variable limiting
- Antiwindup reset
- Antiwindup measures taken only for an active I component

Presentation

Symbol

Representation of the Block:



Parameter description

Block parameter description

Parameter	Data type	Meaning
MAN	BOOL	"1": Manual mode
HALT	BOOL	"1": Halt mode
SP	REAL	Setpoint input
PV	REAL	Input variable
GAIN	REAL	Proportional action coefficient (gain)
TI	TIME	Reset time
YMAX	REAL	Upper limit
YMIN	REAL	Lower limit
YMAN	REAL	Manual value
Y	REAL	Manipulated variable
ERR	REAL	Output system deviation
QMAX	BOOL	"1" = Output Y has reached upper limit
QMIN	BOOL	"1" = Output Y has reached lower limit

Formulae

Transfer function

The transfer function is:

$$G(s) = GAIN \times \left(1 + \frac{1}{TI \times s}\right)$$

The I component can be disabled by setting TI = zero.

Calculation formulae

The calculation formulae are:

$$YP = GAIN \times ERR$$

$$YI_{(new)} = YI_{(old)} + GAIN \times \frac{dt}{TI} \times \frac{ERR_{(new)} + ERR_{(old)}}{2}$$

Output signal Y

The output signal Y is then:

$$Y = YP + YI$$

The I component is formed according to the trapezoid rule.

Explanation of formula sizes

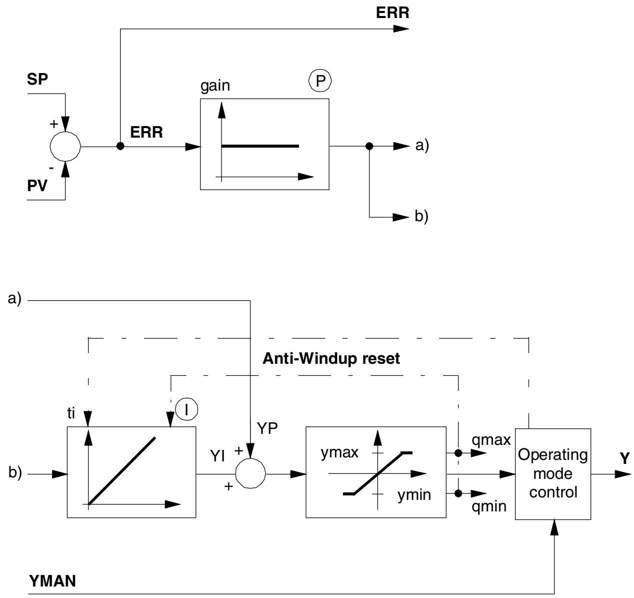
The meaning of the formula sizes is given in the following table:

Size	Meaning
dt	Current scan time
ERR	System deviation (SP - PV)
$ERR_{(old)}$	System deviation value from the previous sampling step
YI	I component
YP	P component

Parametering

Structure diagram

The following is the structure diagram of the PI1 controller:



Parametering

The structure of the PI1 controller is represented in the *Structure diagram, p. 279* above. The parametering of the function block takes place first of all for the elemental PI parameters: the proportional action coefficient GAIN and the reset time TI.

The limits YMAX and YMIN retain the output within the prescribed range. Hence, YMIN ≤ Y ≤ YMAX.

The outputs QMAX and QMIN signal that the output has reached a limit, and thus been capped.

- QMAX = 1 if Y ≥ YMAX
- QMIN = 1 if Y ≤ YMIN

Manipulated variable limiting

After the summation of the components a manipulated variable limiting takes place at the output of the sub controller, which means: YMIN ≤ Y ≤ YMAX

Antiwindup reset

Should limiting of the manipulated variable take place, the antiwindup reset should ensure that the integral component "cannot go berserk". Antiwindup measures are taken only for an active I component. Antiwindup limits are identical to those for manipulated variable limiting. The antiwindup reset measures correct the I component such that: $YMIN - YP \leq YI \leq YMAX - YP$

Operating modes

Selecting the operating modes

There are three operating modes, which are selected via the inputs MAN and HALT.

Operating mode	MAN	HALT
Automatic	0	0
Manual	1	1 or 0
Halt	0	1

Automatic mode

In automatic mode the control output Y is determined through the closed-loop control based on the controlled variable PV and reference variable SP. The control output is limited with YMAX and YMIN. The manipulated variable limits also serve as limits for the Antiwindup reset.

The changeover from automatic to manual is normally not bumpless, since output Y can take on any value between YMAX and YMIN, and Y goes directly to YMAN at the changeover.

If the changeover from automatic to manual is to be bumpless nevertheless, there are two possibilities, which are explained as an example for a PID1 Controller (see *Switching from automatic to manual, p. 316*).

Manual mode

In manual mode the manually manipulated value YMAN is passed on directly to the control output Y. The control output is however limited with YMAX and YMIN. Internal variables will be manipulated in such a manner that the controller changeover from manual to automatic (with I component enabled) can be bumpless. The manipulated variable limits also serve as limits for the Antiwindup reset

Halt mode

In halt mode the control output remains unchanged; the function block does not influence the control output Y, i.e. Y = Y(old). Internal variables will be manipulated in such a manner that the component sum corresponds with the manipulated variable, thus allowing the controller to be driven smoothly from its current position. The manipulated variable limits also serve as limits for the Antiwindup reset.

PI1 controller example

Example

The jump response of the PI1 controller is shown in the following Diagram (see *The jump response of the PI1 controller, p. 281*) as an example.

In the first part of the figure the function block response to manual operating mode can be seen: The.output Y jumps to the YMAN value.

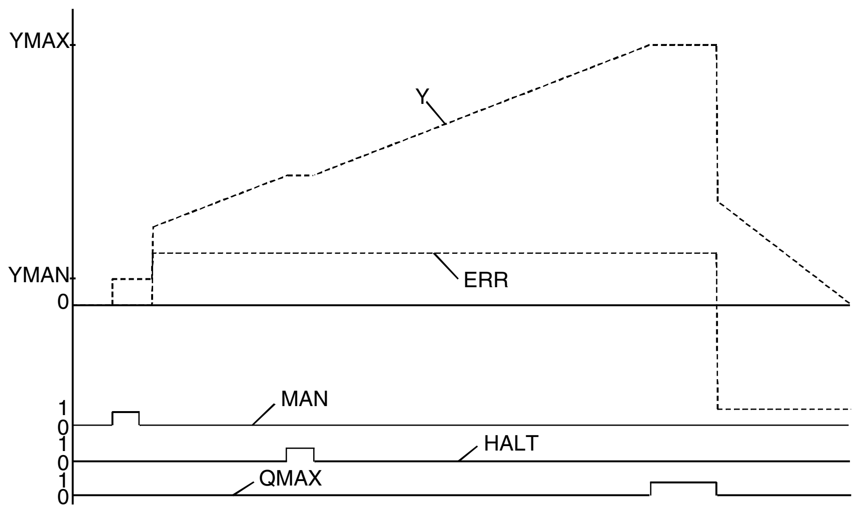
The second part of the diagram shows the reaction of the function block in automatic mode (MAN = 0 and HALT= 0) both with a positive ERR system deviation and with a negative ERR system deviation. For constant positive system deviation, Y ramps upward until the upper output limit is reached.

The output is subsequently limited to the YMAX value. The limit is signaled in the QMAX output. The system deviation then jumps to a negative value whose absolute value is greater than the previous positive value.

Under influence of the P component, the output jumps by the value gain $\mathrm{GAIN} \times (\mathrm{ERR}_{(new)} - \mathrm{ERR}_{(old)})$; thereafter Y ramps downward. The absolute value of the gradient is greater than under the previous positive system deviation. This can be attributed to the now greater absolute value of the system deviation.

The jump response of the PI1 controller

Presentation of the jump response of the PI1 controller



Runtime error

Error message

For YMAX < YMIN an Error message appears.

PI_B: Simple PI controller

34

Overview

At a glance

This chapter describes the PI_B block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block PI_B depicts a PI-algorithm with a mixed structure (series/ parallel). Its functions derive from function block PIDFF (see *PIDFF: Complete PID controller, p. 341*). These functions enable the function block to perform most classical control applications, without compromising user friendliness or using too many system resources. However, for difficult control tasks requiring extended control functions, the PIDFF block should be used.

As additional parameters, EN and ENO can be projected.

Functions

The most important functions of function block PI_B are as follows:
- Calculation of the proportional and integral component in incremental form
- Process value, setpoint value, and default value in physical units
- direct or inverse action
- Possibility of upgrading a block-external I component (RCPY input)
- Dead zone on deviation
- Incremental value and absolute value default
- Upper and lower limit value of the default signal
- Default offset
- Selecting the operating mode manual/automatic
- Tracking mode
- Upper and lower limit of the setpoint value

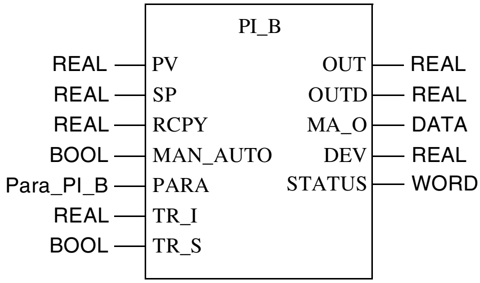
Extended functions

As is the case with PIDFF these functions can be extended by using various additional function blocks:
- Automatic control setting via the block AUTOTUNE
- Internal or external setpoint value selection via the block SP_SEL
- Control over manual operation of the scanned control cycles (see *Scanning, p. 35*) using the function block MS

Representation

Symbol

Representation of the Block:



Parameter description PI_B

Block parameter description

Parameter	**Data type**	**Meaning**
PV	REAL	Process value
SP	REAL	Setpoint
RCPY	REAL	Copy of the effective actuator position
MAN_AUTO	BOOL	Control operating mode: "1" : Automatic mode "0" : Manual mode
PARA	Para_PI_B	Parameter
TR_I	REAL	Initiating input
TR_S	BOOL	Initiating command
OUT	REAL	Actuator output
OUTD	REAL	Differential output Difference between the output of the current and previous execution
MA_O	BOOL	Current operating mode of the function block: "1" : Automatic mode "0" : other operation mode (i.e. manual or tracking mode)
DEV	REAL	Deviation value (PV – SP)
STATUS	WORD	Status word

Parameter description Para_PI_B

Data structure description

Element	Data type	Meaning
id	UINT	Reserved for autotuning
pv_inf	REAL	Lower limit of the process value range
pv_sup	REAL	Upper limit of the process value range
out_inf	REAL	Lower limit of the output value range
out_sup	REAL	Upper limit of the output value range
rev_dir	BOOL	"1" : direct action of the PID controller "0" : inverse action of the PID controller
en_rcpy	BOOL	"1" : the RCPY input is used
kp	REAL	Proportional action coefficient (gain)
ti	TIME	Reset time
dband	REAL	Dead zone on deviation
outbias	REAL	Manual adjustment of static deviation

Formulae

Transfer function

The transfer function is:

$$OUT = kp \times \left(1 + \frac{1}{ti \times p}\right) \times IN$$

Calculation formulae

The formulae actually used vary, depending on whether the function block uses the incremental or the absolute algorithm.

In a simplified form the function block can use one of the following formulae:

Algorithm	ti	Forms
Absolute	0	$OUT = TermP + outbias$ $OUTD = OUT(new) - OUT(old)$
Incremental	>0	$OUTD = TermP + TermI$ $OUT = OUT(old) + OUTD(new)$

Explanation of formula sizes

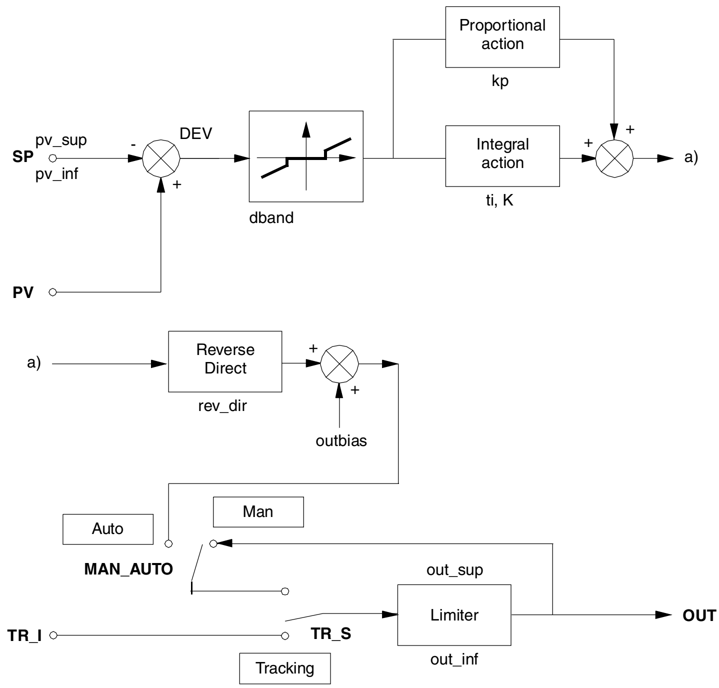
The meaning of the formula sizes is given in the following table:

Size	Meaning
(new)	Value which is calculated on current execution of the function block
(old)	Value which is calculated on previous execution of the function block
OUT	Absolute value output
OUTD	Incremental value output
TermI	Value of the integral component (depending on algorithm)
TermP	Value of the proportional component (depending on algorithm)

Parametering

Structure display of PI_B controller

Structure display of PI_B controller



Absolute algorithm

The absolute algorithm is used if no I component is available (when ti =0) In this case the output OUT is calculated first, and the output modification will then be deducted from this.

Incremental algorithms

Incremental algorithms are used when an I component is available (i.e. when ti > 0). The particularities of this algorithm are that the output alteration OUTD is calculated first and then an absolute value output via the following formulae is determined:

$$\mathrm{OUT(new)} = \mathrm{OUT(old)} + \mathrm{OUTD}$$

For this algorithm, a SERVO function block can be switched to the controller, enabling a static control.

In addition to this the incremental algorithm offers the projection of a block-external integral component for control applications, where the actually upgraded conduct diverts from the conduct calculated by the controller (during open control cycle). In this case it is advantageous to use this for the calculation of the real value. If this is available, the RCPY input must be upgraded and the parameter en_rcpy must be switched to 1. For calculation, therefore, the equation

$$\mathrm{OUT(new)} = \mathrm{OUT(old)} + \mathrm{OUTD}$$

to

$$\mathrm{OUT(new)} = \mathrm{RCPY} + \mathrm{OUTD}$$

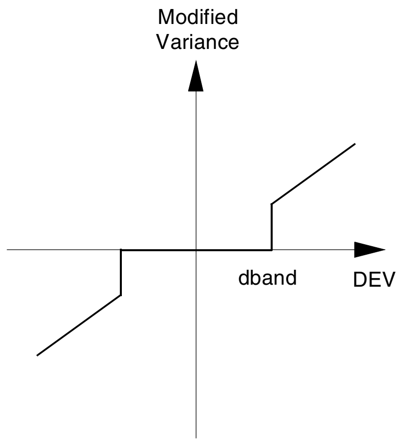
This is particularly useful for cascades or cascade-like controls.

> **Note:** The output OUT is not limited for upgrading an external integral component (en_rcpy=1).

Dead zone on deviation (dband)

Once the work point has been reached, the dead zone is used to limit slight alignments regarding the value of the control element. as long as the deviation lies below dband (in absolute values), the calculation of the function block is based on the value zero.

Display of dead zone on deviation (dband)



Further properties

The block contains the following properties:
- The use of the parameter outbias allows for a precise setting of the work point when no integral componenet is available (ti=0).
- The output OUT is limited to the area between out_inf and out_sup for all operation modes. If a value calculated by the function block (or a written value entered by the user in manual mode) exceeds these limits, the value is cut. The incremental output OUTD, however, never takes this cut into consideration. This enables the PI_B to control a SERVO function block without having to revert the position of the control element (continuous control).
- The choice between direct/inverse action (parameter rev_dir) allows for the adjustment of the control direction of the link control element/measuring process.
- Limiting the setpoint between pv_inf and pv_sup.
- The function block can operate in a purely integral mode (with kp=0).

Operating modes

Function block PI_B has three operating modes: Automatic, Manual and Tracking. The tracking mode is given preference over the other operating modes.

The operating modes are selected via the inputs MAN_AUTO and TR_S:

Operating mode	TR_S	MAN_AUTO	Meaning
Automatic	0	1	The OUT and OUTD outputs correspond to the result of the calculations made by the function block.
Manual	0	0	The output OUT is not set by the function block so that the user can change the value directly.
Tracking	1	0 or 1	The input TR_1 is transferred to the output OUT.

Switching operating modes

The switch manual → automatic or tracking → automatic is carried out as follows:
- The changeover is smooth for the incremental algorithm (ti > 0).
- The changeover is bumpy for the absolute algorithm (ti=0).

Detailed equations

Convention

The following equations use different variables and functions. The variables corresponding with block parameters are not rewritten at this point.

The most important inter-variables and the applied functions will however be described in the following table:

Inter-variables / function	Meaning
dt	Time interval since last function block execution
(new)	Value which is calculated on current function block execution
(old)	Value which was calculated on previous function block execution
TermI	Value of the integral component (depending on algorithm)
TermP	Value of the proportional component (depending on algorithm)
sense	Control sense with the following effect directions: ● +1 This is a direct action (rev_dir = 1) i.e. a positive deviation (PV - SP) generates a higher output value ● -1 This is a inverse action (rev_dir = 0) i.e. a positive deviation (PV - SP) generates a lower output value
Function Δ	$\Delta(x(t)) = x(t) - x(t-1)$
Function 'Limit'	Limit function of block output

Absolute algorithm

The following equations apply for proportional controllers (ti = 0),

$$OUT = TermP + outbias$$
$$OUTD = OUT(new) - OUT(old)$$
$$OUT = limiter(OUT)$$

$$TermP = sense \times kp \times DEV$$

Incremental algorithm

The following equations apply for controllers of type PI > 0);

$$OUTD = TermP + TermI$$

$$OUT = limiter(OUT)$$

If en_rcpy = 0, then

$$OUT = OUT(old) + OUTD(new)$$

If en_rcpy = 1, then

$$OUT = RCPY + OUTD(new)$$

Value of the proportional component TermP

$$TermP = sense \times kp \times [\Delta(DEV)]$$

Value of the integral component TermI, if kp > 0:

$$TermI = sense \times kp \times \frac{dt}{ti} \times DEV$$

Value of the integral component TermI if kp = 0 (pure integral mode):

$$TermI = sense \times \frac{out\_sup - out\_inf}{pv\_sup - pv\_inf} \times \frac{dt}{ti} \times DEV$$

Runtime error

Status word

The following messages are displayed in the status word:

Bit	Meaning
Bit 0 = 1	Error in a calculation with floating point values
Bit 1 = 1	Invalid value recorded at one of the floating point value inputs
Bit 2 = 1	Division by zero for a calculation with floating point values
Bit 3 = 1	Capacity overflow for a calculation with floating point values
Bit 4 = 1	The following behavior is displayed: ● The SP input lies outside the area [pv_inf, pv_sup] : for calculation, the function block uses value pv_inf or pv_sup. ● The kp or dband parameter is negative. the function block uses the value 0 outside the incorrect parameter value. ● The parameter outbias lies outside the area [(out_inf - out_sup), (out_sup - out_inf)]. For calculation, the function block uses the value (out_inf - out_sup) i.e. (out_sup - out_inf).
Bit 5 = 1	The output OUT has reached the lower limit value out_min (see Note)
Bit 6 = 1	The output OUT has reached the upper limit value out_max (see Note)
Bit 7 = 1	The limit values pv_inf and pv_sup are identical.

Note on output OUT

Note: In manual mode these bits stay at 1 for only one program cycle. When the user enters a value for OUT that exceeds one of these limit values, the function block sets Bit 5 or 6 to 1and cuts the value entered by the user. During the next execution of the function block, the value of OUT no longer lies outside the area and bits 5 and 6 are set again at zero.

Error message

An error is displayed when a non-floating point is caught at an input, when a problem occurs during a calculation with floating points or when the limit values pv_inf and pv_sup are identical. The outputs OUT, OUTD, MA_O and DEV remain unchanged.

Warning

In the following cases a warning is given:
- One of the kp or dband parameters is negative. the function block uses the value 0 instead of the incorrect parameter value.
- The parameter outbias is not in the range [(out_inf - out_sup), (out_sup - out_inf)]. For calculation, the function block uses the value (out_inf - out_sup) i.e. (out_sup - out_inf).

PID: PID controller

35

Overview

At a glance

This chapter describes the PID block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block produces a PID controller.

Due to the reference variable SP and the controlled variable PV, a system deviation, ERR, is formed. This ERR system deviation modifies manipulated variable Y.

The parameters EN and ENO can be additionally projected.

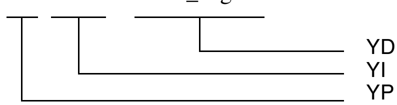
Properties

The Function Block has the following properties:
- real PID controller with independent gain, ti, td setting
- Manual, Halt, Automatic operating modes
- bumpless changeover between manual and automatic
- Manipulated variable limitation in automatic mode
- Separately enabled P, I and D component
- Anti-Windup reset
- Anti-Windup measures taken only for an active I component
- definable delay of the D-component
- D component can be switched to controlled variable PV or system deviation ERR

Transfer function

The transfer function is:

$$G(s) = \text{gain} \times \left(1 + \frac{1}{\text{ti} \times s} + \frac{\text{td} \times s}{1 + \text{td\_lag} \times s}\right)$$



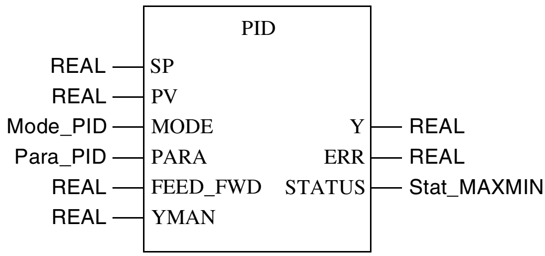
Explanation of the sizes:

Variable	Meaning
YD	D component (only when en_d = 1)
YI	I component (only when en_i = 1)
YP	P component (only when en_p = 1)

Presentation

Symbol

Block display:



Parameter description PID

Block parameter description

Parameter	Data type	Meaning
SP	REAL	Reference variable
PV	REAL	Controlled variable
MODE	Mode_PID	Operating mode
PARA	Para_PID	Parameter
FEED_FWD	REAL	Disturbance variable
YMAN	REAL	Manual manipulation
ERR	REAL	System deviation
Y	REAL	Manipulated variable
STATUS	Stat_MAXMIN	Status of output Y

Parameter description Mode_PID

Data structure description

Element	Data type	Meaning
man	BOOL	"1": Manual mode
halt	BOOL	"1": Halt operating mode
en_p	BOOL	"1": P-component in
en_i	BOOL	"1": I-component in
en_d	BOOL	"1": D-component in
d_on_pv	BOOL	"1": D component in relation to the controlled variable "0": D component in relation to the system deviation

Parameter description Para_PID

Data structure description

Element	Data type	Meaning
gain	REAL	Proportional action coefficient (gain)
ti	TIME	Reset time
td	TIME	Retaining time
td_lag	TIME	Delay of the D-component
ymax	REAL	Upper limit
ymin	REAL	Lower limit

Parameter description Stat_MAXMIN

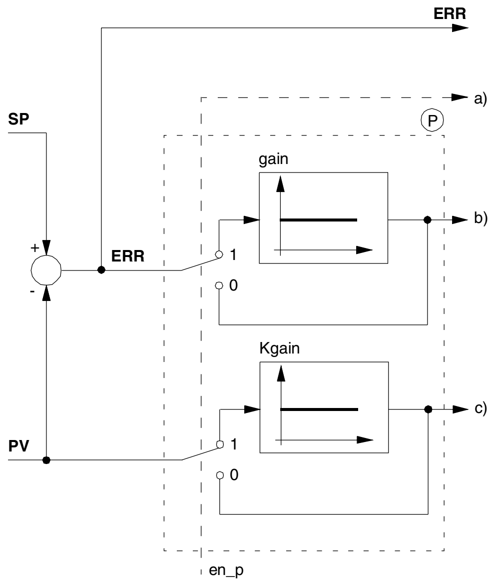
Data structure description

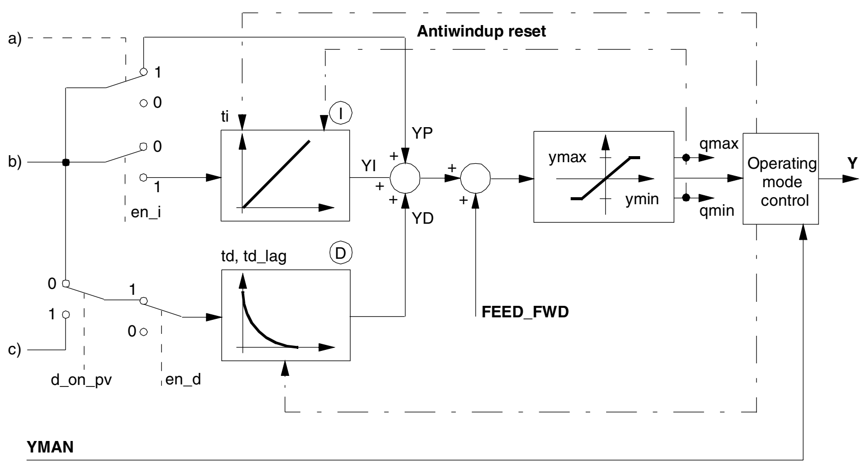
Element	Data type	Meaning
qmax	BOOL	"1" = Y reached upper limit
qmin	BOOL	"1" = Y reached lower limit

PID function block structure diagram

Structure diagram

There follows a structure diagram of the PID block:





Parametering of the PID controller

Parametering

The PID control structure is displayed in *Structure diagram, p. 299*.

The parametering of the function block is first performed by the pure PID parameter, i.e. the proportional action coefficient gain, the reset time ti and the restraining time td.

The D component is delayed by the time td_lag. The relation between td/td_lag is called differential time amplification and is generally selected between 3 and 10. The D component can either be formed based on the system deviation ERR (d_on_pv = 0) or based on the controlled variable PV (don_pv = 1). If the D component is determined based on the controlled variable PV, then no jump occurs during reference variable changes (changes in the SP input) due to the D component. In principle the D component only influences disturbances and process changes.

Reversing the control sense

A reversed behavior of the controller can be achieved by reversing the sign of gain. A positive value on gain causes the increase of the output value, for a positive error variable. A positive value on gain causes the increase of the output value, for a positive error variable.

Limiting of manipulated variable

The limits ymas and ymin limit the upper output as well as the lower output. So that means ymin ≤ Y ≤ ymax.

The outputs qmax and qmin signal that the limit value has been reached, i.e. that the output signal is limited.

- qmax = 1 when Y ≥ ymax
- qmin = 1 when Y ≤ ymin

The upper limit ymax for limiting the manipulated variable must be set higher than the lower limit ymin, otherwise the function block reports an error and does not function.

Antiwindup-Reset

If manipulated variable limiting takes place, the antiwindup reset should make sure that the integral component cannot exceed all limits. The antiwindup measure is only implemented if the I component of the controller is not disabled. The limits for antiwindup are the same here as they are for the manipulated variable limiting. The D component is not taken into consideration for antiwindup measures, so that peaks, caused by the D component, are not capped by the antiwindup-measure.

The antiwindup reset measure corrects the I component in the form, which means:

$$ymin - YP - FEED\_FWD \le YI \le ymax - YP - FEED\_FWD$$

Selecting the control types

There are four different control types, which are selected via the elements en_p, en_i and en_d:

Control type	**en_p**	**en_i**	**en_d**
P controller	1	0	0
PI controller	1	1	0
PD controller	1	0	1
PID controller	1	1	1
I controller	0	1	0

The I-component can also be switched off with ti = 0..

Operating mode

Selecting the operating mode

There are three operating mode, which are selected via the elements man and halt:

Operating mode	man	halt
Automatic	0	0
Manual	1	0 or 1
Halt	0	1

Automatic mode

In automatic mode, the manipulated variable Y is determined by discretized PID algorithm, in relation to the controlled variable PV and the reference variable SP. The manipulated variable is limited by ymax and ymin. The control limits are also limits for the Antiwindup reset.

Manual mode

In manual mode the manual manipulated value YMAN is passed on directly to the manipulated variable Y. The manipulated variable is however limited through ymax and ymin. The internal sizes are tracked in such a way that the controller (on connecting to the I component) can be switched bumplessly from manual to automatic. The control limits are also limits for the Antiwindup reset.

In this operating mode the D component is automatically set to 0.

Halt operating mode

The control output remains as it is found, the function block does not change the manipulated variable Y (controller remains), i.e. Y = Y(old). The internal sizes are tracked in such a way that the controller (on connecting to the I component) bumplessly proceeds from its current position. The control limits are also limits for the Antiwindup reset. The halt operating mode is also useful for setting the control output Y via an external operator device, whereby the internal components are tracked correctly in the controller.

In this operating mode the D component is automatically set to 0.

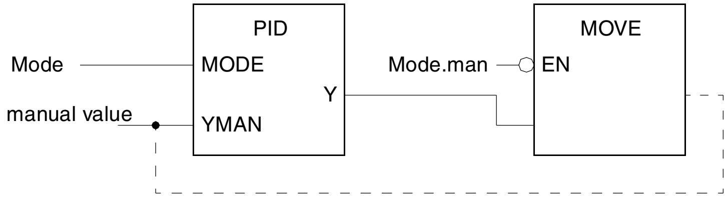
Switching from automatic to manual

The changeover from automatic to manual is normally not bumpless, since output Y can take on any value between ymax and ymin, and yet goes directly to YMAN at the changeover.

There are two possibilities if, nevertheless, a bumpless changeover from automatic to manual is required:
- Switching with the help of the MOVE function
- Switching with the help of the function block increase limit VLIM

Switching via MOVE

Using Function MOVE set the value of YMAN to the value of Y:

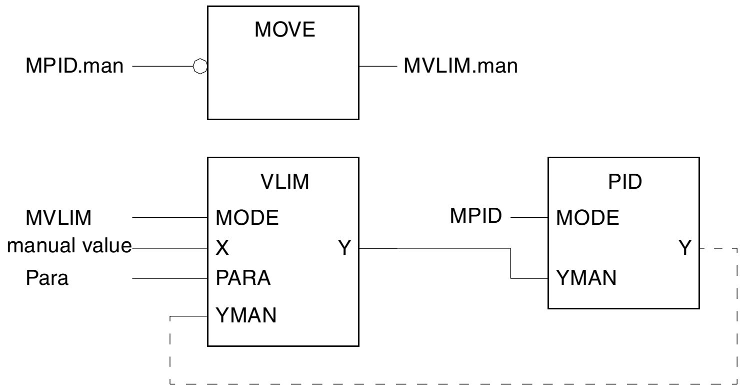


> **Note:** This type of display was selected purely to facilitate comprehension. The links represented by a dotted line can not be programmed as Links (link objects), as they forme unauthorized (in Concept) loops. During programming the links must be implemented through changes.

The MOVE function is only performed when the PID controller is in automatic mode (mode. man = 0). If only one changeover from automatic to manual takes place it is bumpless, as the value of YMAN is equal to the value of Y in this cycle. In the manual mode the value of YMAN can slowly be changed.

Switching via VLIM

Should you not wish to manipulate YMAN, perhaps because it happens to be a constant, then, the previous solution can be implemented using a slew rate limiter (Function block VLIM):



Note: This type of display was selected purely to facilitate comprehension. The links represented by a dotted line can not be programmed as Links (link objects), as they forme unauthorized (in Concept) loops. In programming, the links must be established using variables.

In automatic mode (MPID.man = 0) the slew rate limiter is in manual mode (MOVE function). That way the PID controller manual value (YMAN from PID) can be set to the Y value via the slew rate limiter manual value (YMAN from VLIM). If only one changeover from automatic to manual takes place, it is bumpless, as the value of YMAN(of the PID) is equal to the value of Y (of the PID) in this cycle. The PID controller YMAN value, starting at your adjustment value (Para.rate), are compared with the actual manual value (on VLIM) beginning with the next cycle.

Detailed formulas

Explanation of the formula sizes

Significance of the size in the following formulas:

Size	Meaning
dt	Time differential between the current cycle and the previous cycle
ERR	System deviation (SP - PV)
$ERR_{(new)}$	System deviation value from the current sampling step
$ERR_{(old)}$	System deviation value from the previous sampling step
FEED_FWD	Disturbance variable
$PV_{(new)}$	System deviation value from the current sampling step
$PV_{(old)}$	System deviation value from the prveious sampling step
Y	current output (Halt operating mode) or YMAN (manual mode)
YD	D component
YI	I component
YP	P component

Manipulated variable

The manipulated variable consists of different partial sizes which are dependent on the operating mode.

$$Y = YP + YI + YD + FEED\_FWD$$

After the summation of the components a manipulated variable limiting takes place at the output of the sub controller, which means:

$$ymin \le Y \le ymax$$

Overview to calculate the control components

The following section provides an overview on the different calculations of the control components in relation to the elements en_-, en_I and en_d can be found
- P component YP for manual, Halt and automatic mode
- I component YI for automatic mode
- I component YI for manual and Halt operating mode
- D component YD for automatic mode
- I component YD for manual and Halt operating mode

P component YP for all operating mode

YP for manual, Halt and automatic are located as follows

For en_p = 1 the following applies

$$YP = gain \times ERR$$

For en_p = 0 the following applies

$$YP = 0$$

I component YI for automatic mode

YI for automatic mode is located as follows:

For en_i = 1 the following applies

$$YI_{(new)} = YI_{(old)} + gain \times \frac{dt}{ti} \times \frac{ERR_{(new)} + ERR_{(old)}}{2}$$

For en_i = 0 the following applies

$$YI = 0$$

The I-component is formed according to the trapezoid rule.

I component YI for manual and Halt operating mode

YI for manual, Halt and automatic are located as follows

For en_i = 1 the following applies

$$YI = Y - (YP - FEED\_FWD)$$

For en_i = 0 the following applies

$$YI = 0$$

D component YD for automatic mode

YD for automatic mode is located as follows:

For en_d = 1 and d_on_pv = 0 the following applies:

$$YD_{(new)} = \frac{YD_{(old)} \times td\_lag + td \times gain \times (ERR_{(new)} - ERR_{(old)})}{dt + dt\_lag}$$

For en_d = 1 and d_on_pv = 1 the following applies:

$$YD_{(new)} = \frac{YD_{(old)} \times td\_lag + td \times gain \times (PV_{(old)} - PV_{(new)})}{dt + dt\_lag}$$

For en_d = 0 the following applies

$$YD = 0$$

D component YD for manual and Halt operating mode

YD for manual, Halt and automatic modes are located as follows

YD = 0

Runtime error

Error message

There is an Error message, if
- an invalid floating point number appears at input YMAN or PV,
- or ymax < is ymin

PID1: PID controller

36

Overview

At a glance

This chapter describes the PID1 block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block produces a PID controller.

Due to the reference variable SP and the controlled variable PV, a control difference ERR is formed. This ERR system deviation modifies the Y manipulated variable.

EN and ENO can be projected as additional parameters.

Properties

The function block contains the following properties:
- real PID controller with independent GAIN, TI, TD setting
- Operating mode, Manual, Halt, Automatic
- smooth changeover between manual and automatic
- Limited manipulated variable in automatic mode
- Separately enabled P, I and D component
- Antiwindup Reset
- Antiwindup measure with an active I component only
- definable delay of the D-component
- D component connectable to controlled variable PV or system deviation EER

Transmission function

The transmission function says:

$$G(s) = GAIN \times \left(\underbrace{1}_{YP} + \underbrace{\frac{1}{TI \times s}}_{YI} + \underbrace{\frac{TD \times s}{1 + TD\_LAG \times s}}_{YD} \right)$$

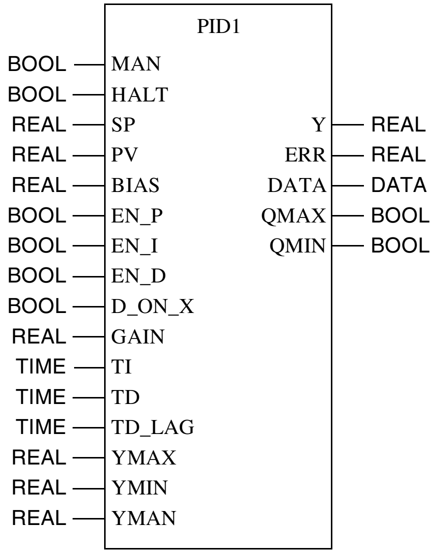
Explaining the sizes:

Size	Meaning
YD	D component (only for EN_D = 1)
YI	I component (only for EN_I = 1)
YP	P component (only for EN_P = 1)

Display

Symbol

Block display:



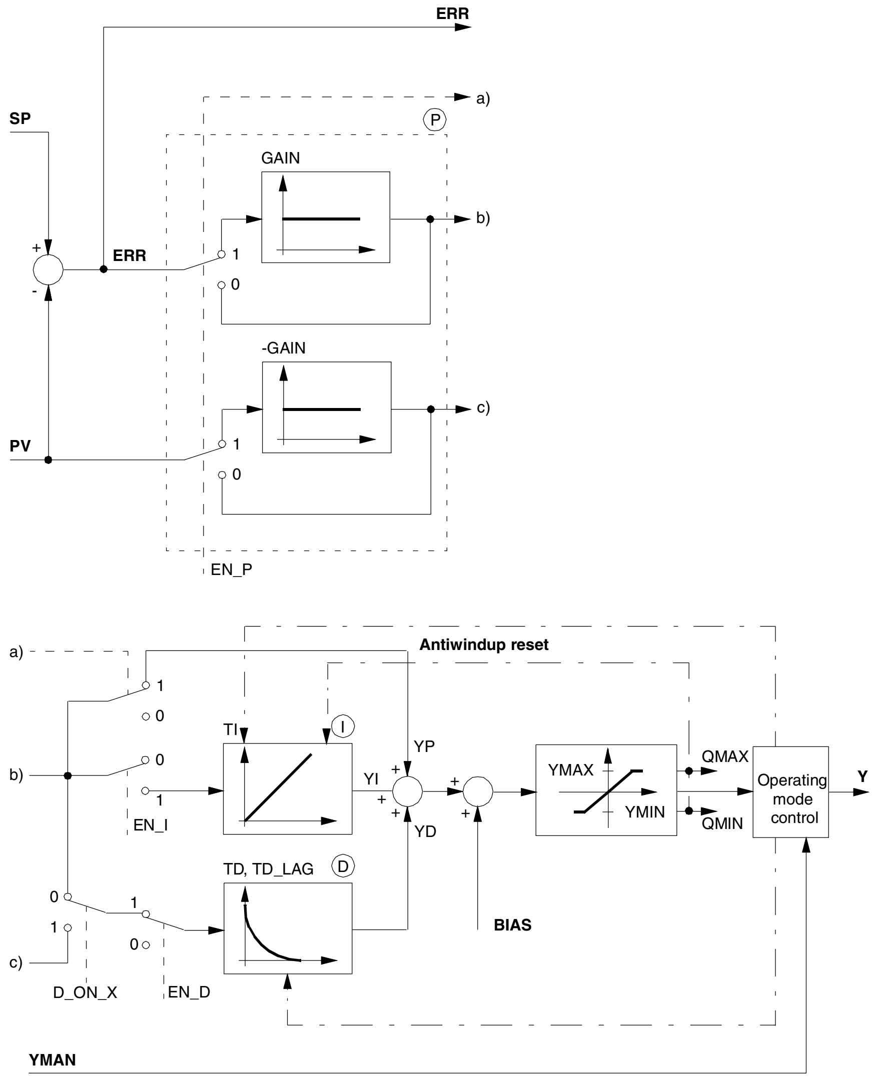
Parameter description

Block parameter description

Parameter	Data type	Meaning
MAN	BOOL	"1": Manual mode
HALT	BOOL	"1": HALT mode
SP	REAL	Setpoint input
PV	REAL	Process variable
BIAS	REAL	Disturbance input
EN_P	BOOL	"1": P component in
EN_I	BOOL	"1": I component in
EN_D	BOOL	"1": D component in
D_ON_X	BOOL	"1": D component on controlled variable "0": D component on system deviation
GAIN	REAL	Proportional action coefficient (gain)
TI	TIME	Reset time
TD	TIME	Retaining time
TD_LAG	TIME	Time lag, D component
YMAX	REAL	Upper limit
YMIN	REAL	Lower limit
YMAN	REAL	Manual manipulation
ERR	REAL	Output system deviation
Y	REAL	Manipulated variable
QMAX	BOOL	1 = Output Y has reached upper limit
QMIN	BOOL	1 = Output Y has reached lower limit

PID1 function block structure

Structure display The following is the structure display of the PIDP1 module:



Parametering the PID1 controller

Parametering

The PID1 controller structure is displayed in *Structure display, p. 313*.

The parametering of the function block is first carried out by the pure PID parameter, i.e. the proportional action coefficient GAIN, the reset time TI and the restraining time TD.

The D component is delayed by the lag time TD_LAG. The ratio between TD/ TD_LAG is called differential gain VD. The D component can either be formed by the system deviation ERR (D_ON_X = 0) or the controlled variable PV (D_ON_X = 1). Should the D component be determined by the controlled variable PV, then the D component will not be able to cause jumps when reference variable fluctuations (changes in input SP) take place. Generally, the D component only affects disturbances and process modifications.

Control direction reversal

A reversed behavior of the controller can be achieved by reversing the sign on GAIN. A positive value on GAIN causes the increase of the output value, for a positive disturbance value. A negative value on gain causes the decrease of the output value, for a positive disturbance value.

Manipulated variable limiting

The limits YMAX and YMIN retain the output within the prescribed range. Hence, YMIN ≤ Y ≤ YMAX.

The outputs QMAX and QMIN signal that the output has reached a limit, and thus been capped.
- QMAX = 1 if Y ≥ YMAX
- QMIN = 1 if Y ≤ YMIN

The upper limit YMAX, limiting the manipulated variable, is to be set higher than the lower limit YMIN.

Antiwindup reset

Should limiting of the manipulated variable take place, the antiwindup reset should ensure that the integral component cannot exceed all limits. Antiwindup measures are only taken if the controller I component is not switched off. Antiwindup limits are identical to those for manipulated variable limiting. The antiwindup measure disregards the D component, to avoid the capping of the D component peaks through the antiwindup measure.

The antiwindup measures correct the I component in such a way that:

$$\mathrm{YMIN} - \mathrm{YP} - \mathrm{BIAS} \leq \mathrm{YI} \leq \mathrm{YMAX} - \mathrm{YP} - \mathrm{BIAS}$$

Selecting the controller types

There are various controller types, which can be selected via the EN_P, EN_I and EN_D parameters.

Controller type	EN_P	EN_I	EN_D
P controller	1	0	0
PI controller	1	1	0
PD controller	1	0	1
PID controller	1	1	1
I controller	0	1	0

The I component can also be disabled with TI = 0.

Operating modes

Selecting the operating modes

There are three operating modes, which can be selected via the MAN and HALT parameters:

Operating mode	MAN	HALT
Automatic	0	0
Manual	1	0 or 1
Halt	0	1

Automatic mode

In automatic mode the manipulated variable Y is determined through the discrete PID algorithm depending on the controlled variable PV and the reference variable SP. The manipulated variable is limited by ymax and ymin. The control limits are also limits for the Antiwindup reset.

Manual mode

In manual mode the manual manipulated value YMAN is passed on directly to the control output Y. The control output is, however, limited by YMAX and YMIN. Internal variables will be manipulated in such a manner that the controller changeover from manual to automatic (with I component enabled) can be bumpless. The control limits are also limits for the Antiwindup reset.

In this operating mode the D component is automatically set to 0.

Halt mode

In halt mode the control output remains unchanged; the function block does not modify the controller output Y (controller remains), i.e. Y = Y(old). Internal variables will be manipulated in such a manner that the component sum corresponds to the control output, thus allowing the controller to be driven smoothly from its current position (when the I component is enabled). The control limits are also limits for the Antiwindup reset.

In this operating mode the D component is automatically set to 0.

Switching from automatic to manual

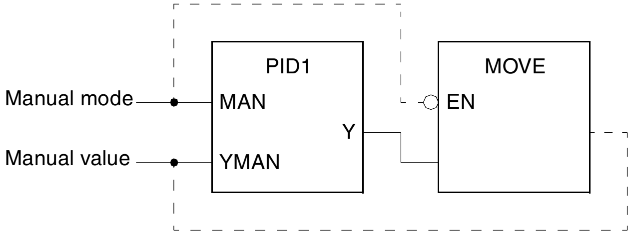
The changeover from automatic to manual is normally not bumpless, since output Y can take on any value between ymax and ymin, and yet goes directly to YMAN at the changeover.

If the changeover from automatic to manual is to be bumpless despite these problems, there are two possibilities:

- Switching with the help of the MOVE function
- Switching with the help of the velocity limiter function block LIMV

Switching via MOVE

With the help of Function MOVE set the value of YMAN to the value of Y:

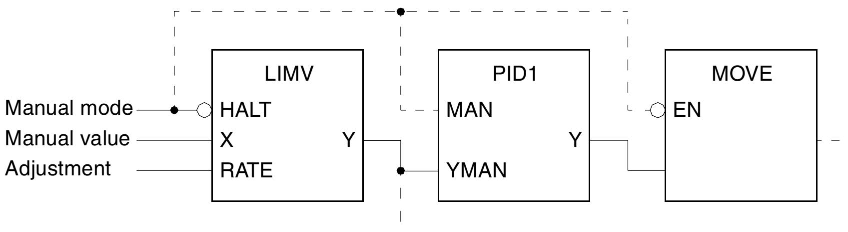


Note: This type of display as selected purely to facilitate comprehension. The links represented by a dotted line can not be programmed as Links (link objects), as they form unauthorized (in Concept) loops. In programming, the links must be established using variables.

The MOVE function is only executed when the PID controller is in automatic or halt mode (MAN = 0). Any subsequent changeover from automatic to manual is bumpless, as the values of YMAN and Y are identical within the same cycle. Now the YMAN value can be slowly changed in manual mode.

Switching with LIMV

Should you not wish to modify YMAN, e.g. because it is a constant, then the previous solution must be replace by a velocity limiter (Function block LIMV (see *LIMV: Velocity limiter: 1st order, p. 203*)):



Note: This type of display was selected purely to facilitate comprehension. The links represented by a dotted line cannot be programmed as Links (link objects), as they form unauthorized (in Concept) loops. In programming, the links must be established using variables.

The MOVE function is only executed when the PID controller is in automatic or halt mode (MAN = 0). If a changeover from automatic to manual is carried out, it is bumpless, as the values of YMAN (PID1) and Y (PID1) are identical within this cycle. The YMAN value (of PID1) together with your adjustment value (RATE), are compared with the actual manual value (on LIMV) beginning with the next cycle.

Detailed formulae

Explanation of formula variables

Significance of variables in the following formulas:

Variable	Meaning
dt	Time differential between the current cycle and the previous cycle
ERR	System deviation (SP - PV)
$ERR_{(new)}$	System deviation value from the current sampling step
$ERR_{(old)}$	System deviation value from the previous sampling step
BIAS	Disturbance variable
$PV_{(new)}$	Value of controlled variable from the current sampling step
$PV_{(old)}$	Value of controlled variable from the previous sampling step
Y	current output (Stop mode) or YMAN (manual mode)
YD	D component
YI	I component
YP	P component

Manipulated variable

The manipulated variable consists of various terms, which are dependent on the operating modes:

$Y = YP + YI + YD + BIAS$

After summation of the components variable limiting takes place, so that:

$YMIN \leq Y \leq YMAX$

Overview to calculate the control components

Following this an overview on the different calculations of the control components in relation to the inputs EN_P, EN_I and EN_D can be found

- P component YP for manual, halt and automatic modes
- I component YI for automatic mode
- I component YI for manual and halt modes
- D component YD for automatic mode
- D component YD for manual and halt modes

P component YP for all operating modes

YP for manual, halt and automatic modes are located as follows

For EN_P = 1 the following applies

$$YP = GAIN \times ERR$$

For EN_P = 0 the following applies

$$YP = 0$$

I component YI for automatic mode

YI for automatic mode is determined as follows:

For EN_I = 1 the following applies

$$YI_{(new)} = YI_{(old)} + GAIN \times \frac{dt}{TI} \times \frac{ERR_{(new)} + ERR_{(old)}}{2}$$

For EN_I = 0 the following applies

$$YI = 0$$

The I-component is formed according to the trapezoid rule.

I component YI for manual and halt modes

YI for manual, halt and automatic modes are determined as follows

For EN_I = 1 the following applies

$$YI = Y - (YP - BIAS)$$

For EN_I = 0 the following applies

$$YI = 0$$

D component YD for automatic mode

YD for automatic mode and cascade is determined as follows:

For EN_D = 1 and D_ON_X = 0 the following applies:

$$YD_{(new)} = \frac{YD_{(old)} \times TD\_LAG + TD \times GAIN \times (ERR_{(new)} - ERR_{(old)})}{dt + TD\_LAG}$$

For EN_D = 1 and D_ON_X = 1 the following applies:

$$YD_{(new)} = \frac{YD_{(old)} \times TD\_LAG + TD \times GAIN \times (PV_{(old)} - PV_{(new)})}{dt + TD\_LAG}$$

For EN_D = 0 the following applies

$$YD = 0$$

D component YD for manual and halt modes

YD for manual, halt and automatic modes are determined as follows:

YD = 0

Runtime error

Error message

For YMAX < YMIN an Error message appears.

PID_P: PID controller with parallel structure

37

Overview

At a glance

This chapter describes the PID_P block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block replicates a PID controller in parallel structure.

A system deviation ERR is formed by the difference between the reference variable SP and the controlled variable PV. This deviation brings about a change of the manipulated variable Y.

EN and ENO can be projected as additional parameters.

Properties

The function block has the following properties:
- PID controller in pure parallel structure
- Independent gains for P-. I and D component
- Each component P, I and D can be individually enabled
- Limiting control limits in automatic mode
- Antiwindup measure with an active I component only
- Antiwindup reset
- Manual, halt and automatic modes
- bumpless manual/automatic mode changeover
- D component can be based on input variable PV or system deviation ERR
- D component with variable delay

Transfer function

The transfer function is:

$$G(s) = \underbrace{kp}_{YP} + \underbrace{\frac{ki}{s}}_{YI} + \underbrace{\frac{kd \times s}{s + \frac{1}{td\_lag}}}_{YD}$$

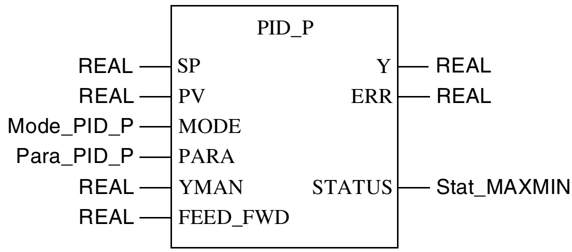
Explanation of the variables:

Variable	Meaning
YD	D component
YI	I component
YP	P component

Representation

Symbol

Block representation:



Parameter description PID_P

Block parameter description

Parameter	Data type	Meaning
SP	REAL	Reference variable
PV	REAL	Controlled variable
MODE	Mode_PID_P	Operating modes
PARA	Para_PID_P	Parameter
YMAN	REAL	Manually manipulated value
FEED_FWD	REAL	Disturbance input
Y	REAL	Manipulated variable
ERR	REAL	System deviation
STATUS	Stat_MAXMIN	Y output status

Parameter description Mode_PID_P

Data structure description

Element	Data type	Meaning
Man	BOOL	"1": Manual mode
Halt	BOOL	"1": Halt mode
d_on_pv	BOOL	"1": D component in relation to the controlled variable, "0": D component in relation to the system deviation
reverse	BOOL	"1": Output reversed

Parameter description Para_PID_P

Data structure description

Element	Data type	Meaning
kp	REAL	Proportional action coefficient (gain = P component)
ki	REAL	Integral action coefficient (gain = I component) [1/s]
kd	REAL	Rate of differentiation (gain = D component) [s]
td_lag	TIME	D component delay time (unit = s)
ymax	REAL	Upper limit
ymin	REAL	Lower limit

Parameter description Stat_MAXMIN

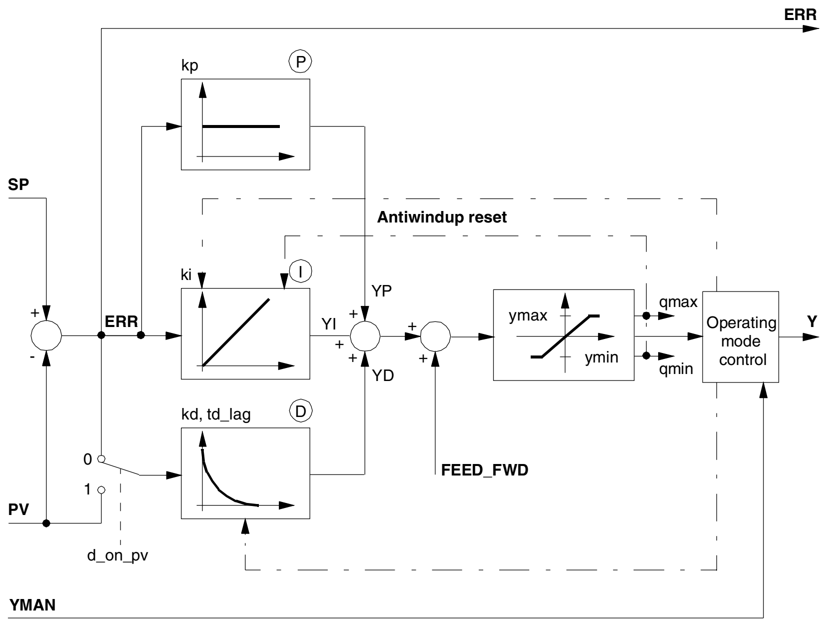
Data structure description

Element	Data type	Meaning
qmax	BOOL	"1" = Y has reached upper limit
qmin	BOOL	"1" = Y has reached lower limit

Parametering of the PID_P controller

Structure diagram

There follows a structure diagram of the PID_P block:



Parametering

The PID_P control structure is displayed in the*Structure diagram, p. 326*.

The parameterization of the PID_P controller takes place first of all for the pure PID parameters, that is to say, the proportional action coefficient kp, the integral action coefficient ki and rate of differentiation kd.

The P, I and D components can be disabled individually by setting the corresponding input (kp, ki oder kd) to 0.

The D component is delayed by the delay time td_lag. The D component can either be based upon the system deviation ERR (d_on_pv = "0") or the controlled variable PV (d_on_pv = "1"). Should the D component be determined by the controlled variable PV, then the D component will not be able to cause jumps when reference variable fluctuations (changes in input SP) take place. In principle, the D component only affects disturbances and process variances.

Control direction reversal

Reversed behavior by the controller can be obtained by setting the reverse input. reverse = 0 has the effect that the output value increases with a positive disturbance. reverse = 1 has the effect that the output value decreases with a positive disturbance.

Manipulated variable limiting

The limits ymax and ymin retain the output within the prescribed range. Hence ymin ≤ Y ≤ ymax.

The outputs qmax and qmin signal that the limit value has been reached, i.e. that the output signal is limited.
- qmax = 1 if Y ≥ ymax
- qmin = 1 when Y ≤ ymin

Upper limit ymax, limiting the manipulated variable, is to be selected greater than lower limit ymin, otherwise the function block reports an error and refuses to function.

Antiwindup reset

Should limiting of the manipulated variable take place, the antiwindup reset should ensure that the I component "cannot go berserk". Antiwindup measures are taken only for an active I component. Antiwindup limits are identical to those for manipulated variable limiting. The antiwindup measures disregard D component values, to avoid being falsely triggered by D component peaks.

The antiwindup measures correct the I component in such a way that:

$$ymin - YP - FEED\_FWD \leq YI \leq ymax - YP - FEED\_FWD$$

Selecting the controller types

Several controller variants can be selected over the parameters kp, ki and kd:

Controller type	kp	ki	kd
P controller	> 0	= 0	= 0
PI controller	> 0	> 0	= 0
PD controller	> 0	= 0	> 0
PID controller	> 0	> 0	> 0
I controller	= 0	> 0	= 0

Operating modes

Selecting the operating modes

There are three operating modes, which are selected via the elements Man and Halt:

Operating mode	Man	Halt
Automatic	0	0
Manual	1	0 or 1
Halt	0	1

Automatic mode

In automatic mode, the manipulated variable Y is determined through the discrete PID closed-loop control algorithm subject to controlled variable PV and reference variable SP. The manipulated variable is limited by ymax and ymin. The control limits are also limits for the Antiwindup reset.

The changeover from automatic to manual is normally not bumpless, since output Y can take on any value between ymax and ymin, and yet goes directly to YMAN at the changeover.

If the changeover from automatic to manual is to be bumpless in spite of this, there are two exemplary possibilities shown for a PID controller (see *Switching from automatic to manual, p. 302*).

Manual mode

In manual mode the manually manipulated value YMAN is passed on directly to the manipulated variable Y. But the manipulated variable is still limited by ymax and ymin. Internal variables will be manipulated in such a manner that the controller changeover from manual to automatic (with I component enabled) can be bumpless. The control limits are also limits for the Antiwindup reset.

In this operating mode the D component is automatically set to 0.

Halt mode

In halt mode the control output remains unchanged; the function block does not influence the manipulated variable Y, i.e. Y = Y(old). Internal variables will be manipulated in such a manner that the controller (with I component enabled) can be driven smoothly from its current position. The control limits are also limits for the Antiwindup reset. Halt mode is also useful in allowing an external operator device to adjust control output Y, and the controller's internal components are given the chance to continuously react to the external influence.

In this operating mode the D component is automatically set to 0.

Detailed formulas

Explanation of formula variables

Meaning of the variables in the formulas:

Variable	Meaning
dt	Time differential between the current cycle and the previous cycle
ERR	System deviation (SP - PV)
$ERR_{(new)}$	System deviation value from the current sampling step
$ERR_{(old)}$	System deviation value from the previous sampling step
FEED_FWD	Disturbance variable
$PV_{(new)}$	Value of controlled variable from the current sampling step
$PV_{(old)}$	Value of controlled variable from the previous sampling step
Y	current output (halt mode) or YMAN (manual mode)
YD	D component
YI	I component
YP	P component

Manipulated variable

The manipulated variable is composed of various terms:

$$Y = YP + YI + YD + FEED\_FWD$$

After the summation of the components a manipulated variable limiting takes place at the output of the sub controller, which means:

$$ymin \le Y \le ymax$$

System deviation

The system deviation is determined as follows:

ERR = SP - PV, if reverse = 0

ERR = PV - SP, if reverse = 1

Overview to calculate the control components

Following this an overview on the different calculations of the control components in relation to the gains kp, ki and kd can be found:
- P component YP for manual, halt and automatic modes
- I component YI for automatic mode
- I component YI for manual and halt modes
- D component YD for automatic mode
- D component YD for manual and halt modes

P component YP for all operating modes

YP for manual, halt and automatic modes are located as follows

$$YP = kp \times ERR$$

I component YI for automatic mode

YI for automatic mode is determined as follows:

For ki > 0 applies:

$$YI_{(new)} = YI_{(old)} + ki \times dt \times \frac{ERR_{(new)} + ERR_{(old)}}{2}$$

For ki = 0 the following applies

$$YI = 0$$

The I-component is formed according to the trapezoid rule.

I component YI for manual and halt modes

YI for manual, halt and automatic modes is determined as follows

For ki > 0 applies:

$$YI = Y - (YP - FEED\_FWD)$$

For ki = 0 the following applies

$$YI = 0$$

D component YD for automatic mode

YD for automatic mode and cascade is determined as follows:

For kd > 0 and d_on_pv = 0 applies:

$$YD_{(new)} = \frac{td\_lag}{dt + td\_lag} \times (YD_{(old)} + kd \times (ERR_{(new)} - ERR_{(old)}))$$

For kd > 0 and d_on_pv = 1 applies:

$$YD_{(new)} = \frac{td\_lag}{dt + td\_lag} \times (YD_{(old)} + kd \times (PV_{(old)} - PV_{(new)}))$$

For kd = 0 the following applies

$$YD = 0$$

D component YD for manual and halt modes

YD for manual, halt and automatic modes are determined as follows:

$$YD = 0$$

Runtime error

Error message

There is an Error message, if:
- an invalid floating point number appears at input YMAN, or if
- is ymax < ymin.

PID_PF: PID controller with parallel structure

38

Overview

At a glance

This chapter describes the PID_PF block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block replicates a PID controller in parallel structure.

A system deviation ERR is formed by the difference between the reference variable SP and the controlled variable PV. This deviation brings about a change of the manipulated variable Y.

EN and ENO can be projected as additional parameters.

Properties

The function block has the following properties:
- PID controller in pure parallel structure
- Independent gains for P-. I and D component
- Each component P, I and D can be individually enabled
- Limiting control limits in automatic mode
- Antiwindup measure with an active I component only
- Antiwindup reset
- Manual, halt and automatic modes
- bumpless manual/automatic mode changeover
- D component can be based on input variable PV or system deviation ERR
- D component with variable delay

Transfer function

The transfer function is:

$$G(s) = \underbrace{kp}_{YP} + \underbrace{\frac{ki}{s}}_{YI} + \underbrace{\frac{kd \times s}{s + \frac{1}{td\_lag}}}_{YD}$$

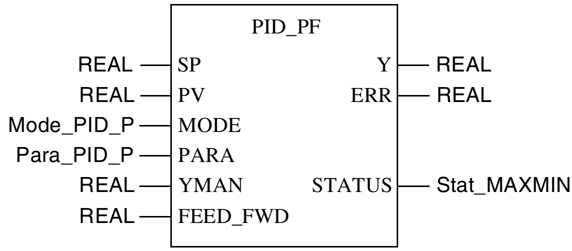
Explanation of the variables:

Variable	Meaning
YD	D component
YI	I component
YP	P component

Representation

Symbol

Block representation:



Parameter description PID_PF

Block parameter description

Parameter	Data type	Meaning
SP	REAL	Reference variable
PV	REAL	Controlled variable
MODE	Mode_PID_P	Operating modes
PARA	Para_PID_P	Parameter
YMAN	REAL	Manually manipulated value
FEED_FWD	REAL	Disturbance input
Y	REAL	Manipulated variable
ERR	REAL	System deviation
STATUS	Stat_MAXMIN	Y output status

Parameter description Mode_PID_P

Data structure description

Element	Data type	Meaning
Man	BOOL	"1": Manual mode
Halt	BOOL	"1": Halt mode
d_on_pv	BOOL	"1": D component in relation to the controlled variable, "0": D component in relation to the system deviation
reverse	BOOL	"1": Output reversed

Parameter description Para_PID_P

Data structure description

Element	Data type	Meaning
kp	REAL	Proportional action coefficient (gain = P component)
ki	REAL	Integral action coefficient (gain = I component) [1/s]
kd	REAL	Rate of differentiation (gain = D component) [s]
td_lag	TIME	D component delay time
ymax	REAL	Upper limit
ymin	REAL	Lower limit

Parameter description Stat_MAXMIN

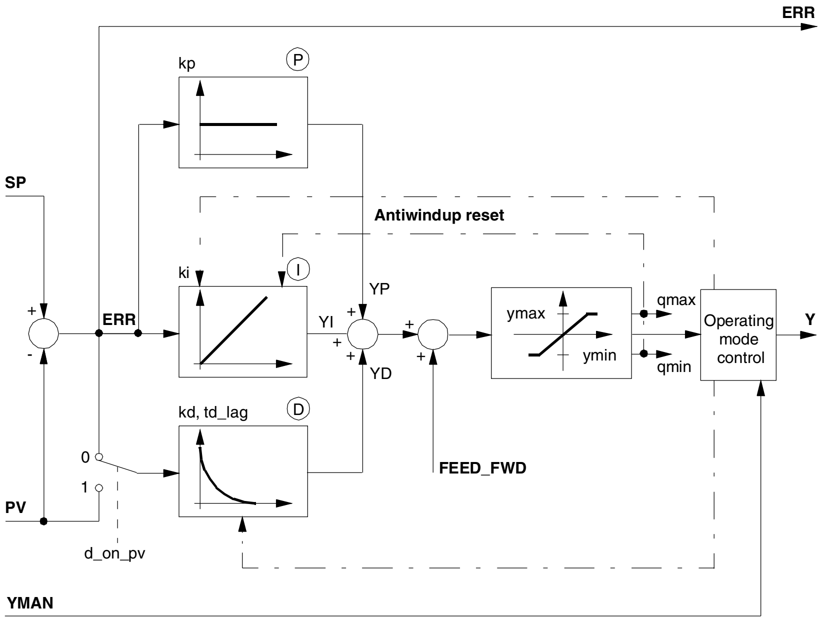
Data structure description

Element	Data type	Meaning
qmax	BOOL	"1" = Y has reached upper limit
qmin	BOOL	"1" = Y has reached lower limit

Parametering of the PID_PF controller

Structure diagram

There follows a structure diagram of the PID_PF block:



Parametering

The PID_PF control structure is displayed in the*Structure diagram, p. 335*.

The parameterization of the PID_PF controller takes place first of all for the pure PID parameters, that is to say, the proportional action coefficient kp, the integral action coefficient ki and rate of differentiation kd.

The P, I and D components can be disabled individually by setting the corresponding input (kp, ki oder kd) to 0.

The D component is delayed by the delay time td_lag. The D component can either be based upon the system deviation ERR (d_on_pv = "0") or the controlled variable PV (d_on_pv = "1"). Should the D component be determined by the controlled variable PV, then the D component will not be able to cause jumps when reference variable fluctuations (changes in input SP) take place. In principle, the D component only affects disturbances and process variances.

Control direction reversal

Reversed behavior by the controller can be obtained by setting the reverse input. reverse = 0 has the effect that the output value increases with a positive disturbance. reverse = 1 has the effect that the output value decreases with a positive disturbance.

Manipulated variable limiting

The limits ymax and ymin retain the output within the prescribed range. Hence ymin ≤ Y ≤ ymax.

The outputs qmax and qmin signal that the limit value has been reached, i.e. that the output signal is limited.

- qmax = 1 if Y ≥ ymax
- qmin = 1 when Y ≤ ymin

Upper limit ymax, limiting the manipulated variable, is to be selected greater than lower limit ymin, otherwise the function block reports an error and refuses to function.

Antiwindup reset

Should limiting of the manipulated variable take place, the antiwindup reset should ensure that the I component "cannot go berserk". Antiwindup measures are taken only for an active I component. Antiwindup limits are identical to those for manipulated variable limiting. The antiwindup measures disregard D component values, to avoid being falsely triggered by D component peaks.

The antiwindup measures correct the I component in such a way that:

$$\mathrm{ymin} - \mathrm{YP} - \mathrm{FEED\_FWD} \le \mathrm{YI} \le \mathrm{ymax} - \mathrm{YP} - \mathrm{FEED\_FWD}$$

Selecting the controller types

Several controller variants can be selected over the parameters kp, ki and kd:

Controller type	kp	ki	kd
P controller	> 0	= 0	= 0
PI controller	> 0	> 0	= 0
PD controller	> 0	= 0	> 0
PID controller	> 0	> 0	> 0
I controller	= 0	> 0	= 0

Operating modes

Selecting the operating modes

There are three operating modes, which are selected via the elements Man and Halt:

Operating mode	Man	Halt
Automatic	0	0
Manual	1	0 or 1
Halt	0	1

Automatic mode

In automatic mode, the manipulated variable Y is determined through the discrete PID closed-loop control algorithm subject to controlled variable PV and reference variable SP. The manipulated variable is limited by ymax and ymin. The control limits are also limits for the Antiwindup reset.

The changeover from automatic to manual is normally not bumpless, since output Y can take on any value between ymax and ymin, and yet goes directly to YMAN at the changeover.

If the changeover from automatic to manual is to be bumpless in spite of this, there are two exemplary possibilities shown for a PID controller (see *Switching from automatic to manual, p. 302*).

Manual mode

In manual mode the manually manipulated value YMAN is passed on directly to the manipulated variable Y. But the manipulated variable is still limited by ymax and ymin. Internal variables will be manipulated in such a manner that the controller changeover from manual to automatic (with I component enabled) can be bumpless. The control limits are also limits for the Antiwindup reset.

In this operating mode the D component is automatically set to 0.

Halt mode

In halt mode the control output remains unchanged; the function block does not influence the manipulated variable Y, i.e. Y = Y(old). Internal variables will be manipulated in such a manner that the controller (with I component enabled) can be driven smoothly from its current position. The control limits are also limits for the Antiwindup reset. Halt mode is also useful in allowing an external operator device to adjust control output Y, and the controller's internal components are given the chance to continuously react to the external influence.

In this operating mode the D component is automatically set to 0.

Detailed formulas

Explanation of formula variables

Meaning of the variables in the formulas:

Variable	Meaning
dt	Time differential between the current cycle and the previous cycle
ERR	System deviation (SP - PV)
$ERR_{(new)}$	System deviation value from the current sampling step
$ERR_{(old)}$	System deviation value from the previous sampling step
FEED_FWD	Disturbance variable
$PV_{(new)}$	Value of controlled variable from the current sampling step
$PV_{(old)}$	Value of controlled variable from the previous sampling step
Y	current output (halt mode) or YMAN (manual mode)
YD	D component
YI	I component
YP	P component

Manipulated variable

The manipulated variable is composed of various terms:

$$Y = YP + YI + YD + FEED\_FWD$$

After the summation of the components a manipulated variable limiting takes place at the output of the sub controller, which means:

$$ymin \le Y \le ymax$$

System deviation

The system deviation is determined as follows:

ERR = SP - PV, if reverse = 0

ERR = PV - SP, if reverse = 1

Overview to calculate the control components

Following this an overview on the different calculations of the control components in relation to the gains kp, ki and kd can be found:
- P component YP for manual, halt and automatic modes
- I component YI for automatic mode
- I component YI for manual and halt modes
- D component YD for automatic mode
- D component YD for manual and halt modes

P component YP for all operating modes

YP for manual, halt and automatic modes are located as follows

$$YP = kp \times ERR$$

I component YI for automatic mode

YI for automatic mode is determined as follows:

For ki > 0 applies:

$$YI_{(new)} = YI_{(old)} + ki \times dt \times \frac{ERR_{(new)} + ERR_{(old)}}{2}$$

For ki = 0 the following applies

$$YI = 0$$

The I-component is formed according to the trapezoid rule.

I component YI for manual and halt modes

YI for manual, halt and automatic modes is determined as follows

For ki > 0 applies:

$$YI = Y - (YP - FEED\_FWD)$$

For ki = 0 the following applies

$$YI = 0$$

D component YD for automatic mode

YD for automatic mode and cascade is determined as follows:

For kd > 0 and d_on_pv = 0 applies:

$$YD_{(new)} = \frac{td\_lag}{dt + td\_lag} \times (YD_{(old)} + kd \times (ERR_{(new)} - ERR_{(old)}))$$

For kd > 0 and d_on_pv = 1 applies:

$$YD_{(new)} = \frac{td\_lag}{dt + td\_lag} \times (YD_{(old)} + kd \times (PV_{(old)} - PV_{(new)}))$$

For kd = 0 the following applies

$$YD = 0$$

D component YD for manual and halt modes

YD for manual, halt and automatic modes are determined as follows:

$$YD = 0$$

Runtime error

Error message

There is an Error message, if:
- an invalid floating point number appears at input YMAN, or if
- is ymax < ymin.

PIDFF: Complete PID controller

39

Overview

At a glance

This chapter describes the PIDFF block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The PIDFF Function block is based on a PID algorithm with parallel or mixed structure (series / parallel).

EN and ENO can be configured as additional parameters.

Functions

It displays numerous functions:
- Calculating the proportional, integral and differential component in its incremental form
- 2 antiwindup measures
- Process value, setpoint and output in physical units
- Direct or inverse action
- Differential component to process value or deviation
- Parametering the transfer gain of the differential component
- Weight of the setpoint in the proportional component (reducing the overrun)
- Possibility of upgrading a block external integral component (RCPY input)
- Feed forward component for disturbance compensation (FF input)
- Dead zone on deviation
- Incremental value and absolute value output
- Upper and lower limit on the output signal (according to operating mode)
- Gradient limitation of the output signal
- Output offset
- Selecting manual/automatic mode
- Tracking mode
- Upper and lower setpoint limit

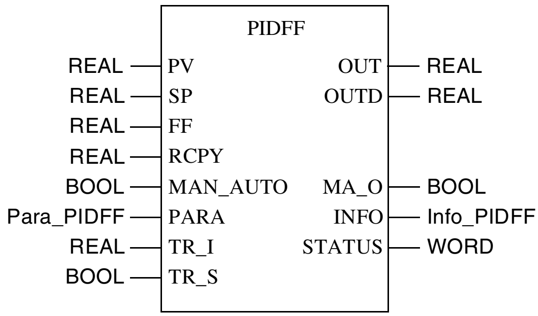
Complementary functions

Other function blocks complement these functions when used in conjunction with the PIDFF block:
- Autotuning via the AUTOTUNE-Function block
- Selecting an internal or external setpoint via the function block SP_SEL
- Controlling manual operation of the sampled control loops (see *Scanning, p. 35*) using the function block MS

Representation

Symbol

Block representation:



PIDFF Parameter Description

Block parameter description

Parameter	Data type	Meaning
PV	REAL	Process value
SP	REAL	Setpoint
FF	REAL	Feed forward input
RCPY	REAL	Copy of the current manipulated variable
MAN_AUTO	BOOL	Controller operating mode: "1": Automatic mode "0": Manual mode
PARA	Para_PIDFF	Parameter
TR_I	REAL	Initialization input
TR_S	BOOL	Initialization command
OUT	REAL	Absolute value output
OUTD	REAL	Incremental value output: Difference between the output of the current and previous cycle
MA_O	BOOL	Current operating mode of the function block: "1": Automatic operating mode "0": other operating mode (i.e. manual or tracking mode)
INFO	Info_PIDFF	Information
STATUS	WORD	Status word

Parameter description Para_PIDFF

Data structure description

Element	Data type	Meaning
id	UINT	Reserved for autotuning
pv_inf	REAL	Lower limit of the process value range
pv_sup	REAL	Upper limit of the process value range
out_inf	REAL	Lower limit of the output value range
out_sup	REAL	Upper limit of the output value range
rev_dir	BOOL	"0": direct action of the PID controller "1": inverse action of the PID controller
mix_par	BOOL	"1": PID controller with parallel structure "0": PID controller with mixed structure
aw_type	BOOL	"1": Anti-windup halt is filtered
en_rcpy	BOOL	"1": the RCPY input is used
kp	REAL	Proportional contribution (gain)
ti	TIME	Integral time
td	TIME	Derivative time
kd	REAL	Differential gain
pv_dev	BOOL	Type of differential contribution: "1": Differential contribution in relation to system deviation "0": Differential contribution in relation to regulating variable (process value)
bump	BOOL	"1": Transition to automatic mode with bump "0": Bumpless transition to automatic mode
dband	REAL	Dead zone on deviation
gain_kp	REAL	Reducing the proportional contribution within the dead zone dband
ovs_att	REAL	Reducing the overrun
outbias	REAL	Manual compensation for the static deviation
out_min	REAL	Lower limit of the output
out_max	REAL	Upper limit of the output
outrate	REAL	Limit for output modification in units per second (≥ 0)
ff_inf	REAL	Lower limit of the FF range
ff_sup	REAL	High limit of the FF range
otff_inf	REAL	Low limit of the out_ff range
otff_sup	REAL	High limit of the out_ff range

Parameter description Info_PIDFF

Data structure description

Element	Data type	Meaning
dev	REAL	Deviation value (PV – SP)
out_ff	REAL	Value of the feed forward contribution

Formulae

Transfer function

Depending on whether the mixed or parallel structure is being used, the transfer function is as follows:

Structure	Formulae
Mixed	$OUT = kp \times \left[1 + \frac{1}{ti \times p} + \frac{td \times p}{1 + \left(\frac{td}{kd}\right) \times p}\right] \times IN$
Parallel	$OUT = \left[kp + \alpha \times \frac{1}{ti \times p} + \alpha \times \frac{td \times p}{1 + \left(\frac{td}{kd}\right) \times p}\right] \times IN$
with α = scaling factor	$OUT = \frac{out\_sup - out\_inf}{pv\_sup - pv\_inf}$

Calculation formulae

The formulae actually used vary, depending on whether the function block uses the incremental or absolute form of the algorithm.

In a simplified form the function block can use one of the following formulae:

Algorithm	ti	Formulae
Absolute	0	$OUT = TermP + TermD + TermFF + outbias$ $OUTD = OUT(new) - OUT(old)$
Incremental	>0	$OUTD = TermP + TermI + TermD + TermFF$ $OUT = OUT(old) + OUTD(new)$

Explanation of formula variables

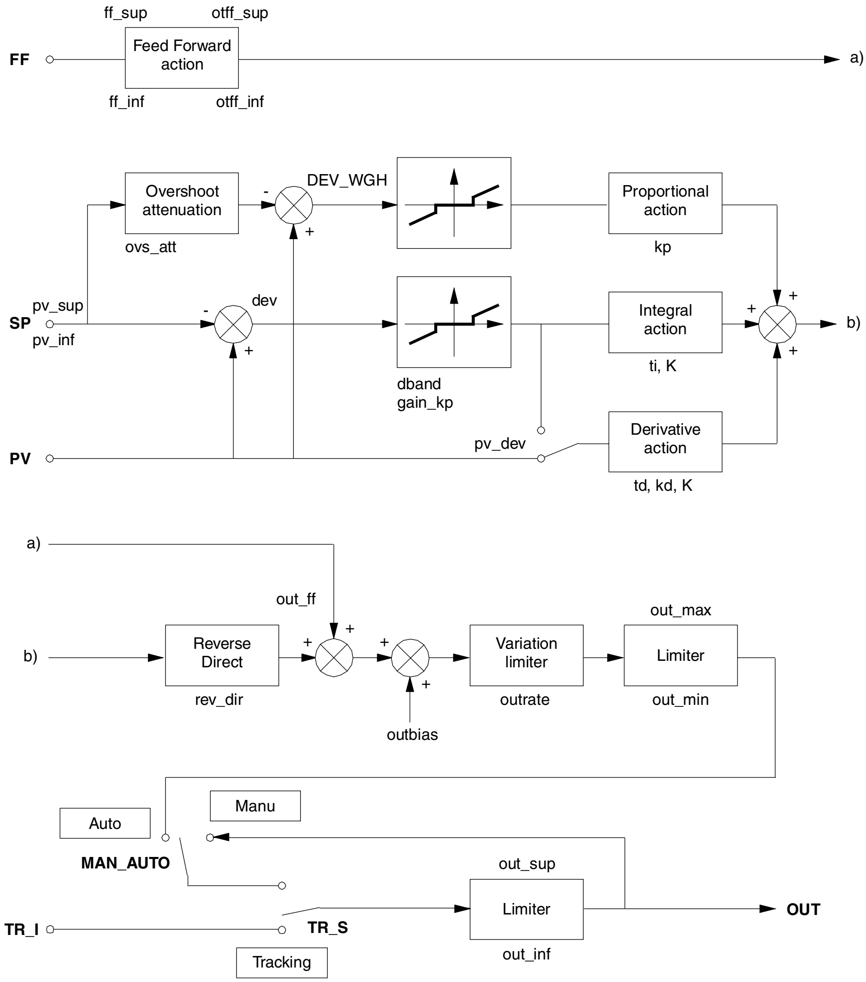
The meaning of the formula sizes is given in the following table:

Variable	Meaning
(new)	Value which is calculated on current function block execution
(old)	Value which was calculated on previous function block execution
OUT	Absolute value output
OUTD	Incremental value output
TermD	Value of the differential component
TermFF	Value of the feed forward component (disturbance compensation)
TermI	Value of the integral component
TermP	Value of the proportional component

Structure diagram of the PIDFF controller

Structure diagram

Structure diagram of the PIDFF controller



Parametering

Mixed/parallel structure (mix_par)

Structure selection takes place via the mix_par parameter :

If…	Then …
mix_par = 0	there is a mixed structure, i.e. the proportional component is set up in the connection to the integral and differential component. The gain K set up for the components (see *Structure diagram, p. 347*) corresponds to kp.
mix_par = 1	the structure is parallel, i.e. the proportional coefficient is set up parallel to the integral and differential coefficient. In this case, the gain kp does not related to the integral and differential component. In this case, gain K corresponds to the relationship between the output zone and the range.

Absolute algorithms (ti = 0)

Absolute algorithms are used when no integral component is set up (ti = 0). In this case the output OUT is calculated first, and then the output alteration is deducted.

Incremental algorithms (ti > 0)

Incremental algorithms are used when an integral component is present (i.e. when ti > 0). The special feature of this algorithm is that the output alteration OUTD is calculated first and then an absolute value output is determined according to the following formula:

$$\mathrm{OUT(new)} = \mathrm{OUT(old)} + \mathrm{OUTD}$$

This algorithm form makes it possible to switch a SERVO-function block to the controller and thus to attain static control.

The incremental form also offers the following possibilities:

Possibility	**Explanation**
External block integral component (mit en_rcpy = 1)	If the real component deviates from the value calculated by the controller (with an open servoloop), the real value should be used as the basis for the calculation. If this value is available, it should be assigned to the RCPY input and the parameter en_rcpy must be switched to 1. In calculations done by the function block, the equation OUT(new) = OUT (old) + OUTD to OUT(new) = RCPY + OUTD This is particularly beneficial for cascades or cascade-like controls. **Note**: In this case the OUT output is not limited.
Expanded antiwindup measure	The incremental form of the PID controller offers as standard an antiwindup measure taken into account in the algorithm. This type is the basis when aw_type = 0. In this case the output can be saturated and suddenly leave its threshold, even if the sign of the deviation does not change (e.g. if it is affected by a brief disturbance during measuring). It is possible to use a second antiwindup measure (aw_type = 1) which prevents the output from exceeding its threshold as long as the deviation does not alter the sign.

Weight of the setpoint in the proportional component (reducing the overrun)

If an integral component is present (ti > 0), the ovs_att parameter makes the weight of the proportional component possible the calculation of the proportional component is based on the weighted deviation ($PV - (1 - ovs\_att) \times SP$).

This could have an influence in the case of an overrun, as can occur with setpoint modifications. The aim is to retain a control-intensive proportional component and therefore a dynamic response to disturbances without an overrun occurring during control.

The parameter ovs_att can fluctuate continually between:

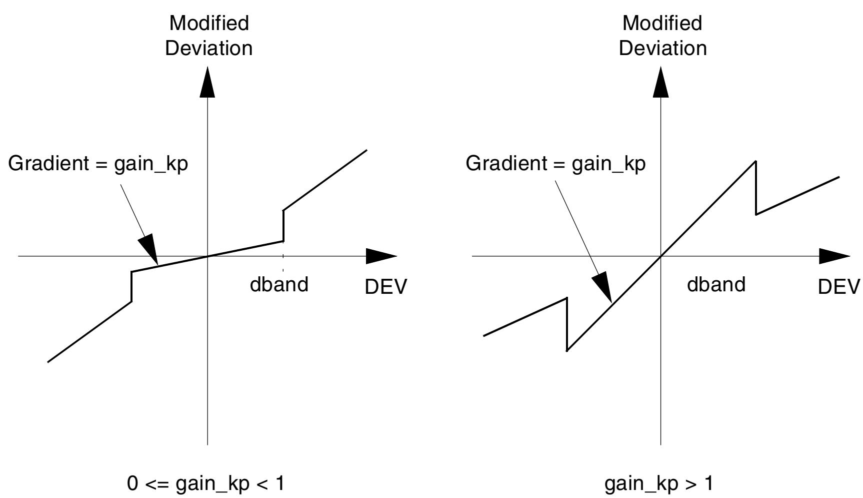
Value	Meaning
0	to the proportional component (classic case) assigned to the deviation (system deviation)
1	for the proportional component (with sensitive processes or processes with an integral effect) assigned to the measurement (controlled variable),.

Dead zone on deviation (dband)

When the work point is reached the dead zone can limit smaller values to the actuator’s value. as long as the deviation lies below dband, the calculation of the function block is based on the value zero.

The extended parameter gain_kp can be used to modify the deviation inside the dead zone. This is better than deleting it. The modified deviation (multiplied by gain_kp) is used to calculate the proportional and integral components.

Representation of the alteration of the deviation



Transfer gain with the differential component

The PIDFF function block contains a filter of the first order for the differential component. The filter gain kd can be parametered so that processes where the differential component must be very strongly filtered can be processed as well as processes where the filtering of the differential component can be removed because the signal is "pure" enough.

Feed forward component for disturbance compensation (FF input)

With classic PID control, the controller reacts to output modifications of the control process (closed servoloop). In the case of a disturbance, the controller only reacts if the process value deviates from the setpoint value. The feed-forward-function means that a measurable disturbance can be compensated for as soon as it arises. This function, conceived as an open servoloop, removes the effects of the disturbance. in this case the term disturbance size update (Feed Forward) is used.

The component of the feed forward input is updated directly/inversely to the manipulated variable of the controller after the control direction has been included.

The calculation proceeds according to the following formula:

$$out\_ff = \frac{(FF - ff\_inf) \times (otff\_sup - otff\_inf)}{(ff\_sup - ff\_inf)} + otff\_inf$$

A specific user example of this function is given in the section "*Application example of the feed forward function , p. 360*".

> **Note:** If ff_sup = ff_inf, the calculation of the feed_forward component is ignored.

Further properties

The block contains the following properties:

- The outbias parameter makes precision at the work point possible if the process contains no integral component (ti = 0).
- In automatic mode, the OUT output is limited to the range between out_min and out_max, and to the range between out_inf and out_sup in manual and tracking mode. If a value calculated by the function block (or a written value entered by the user in manual mode) exceeds one of these limits, the value is capped. The incremental output OUT_D, however, never takes this capping into consideration. This enables the PIDFF function block to control a SERVO function block without having to revert the position of the acuator (continuous control).
- The output speed increase is limited by the parameter outrate.
- The possibility of selecting between direct/inverse action (parameter rev_dir) allows for the adjustment of the control direction of the link actuator/ process.
- The differential component can affect both the process value (pv_dev = 0), and the deviation (pv_dev = 1).
- pv_inf and pv_sup correspond to the upper and lower threshold of the setpoint value.
- The function block can also have an effect in pure integral mode (with kp = 0).

Operating modes

Selecting the operating modes

There are 3 operating modes for the PIDFF function block: Automatic, Manual and Tracking. As the following table shows, the tracking mode takes priority over the other operating modes.

The operating modes are selected via the MAN_AUTO and TR_S inputs:

Operating mode	TR_S	MAN_AUTO	Meaning
Automatic	0	1	The OUT and OUTD outputs correspond to the result of the calculations made by the function block. The thresholds for the OUT output are out_min and out_max.
Manual	0	0	The output OUT is not set via the function block. Its value can be directly modified by the user. OUT remains limited however; this operating mode involves the thresholds out_inf and out_sup (instead of out_min and out_max in automatic mode).
Tracking	1	0 or 1	The input TR_1 is transferred to the output OUT. As in manual mode, OUT is between the thresholds out_inf and out_sup.

Switching from Manual -> Automatic or Tracking -> Automatic

The type of changeover depends on bump:

If…	Then …
bump = 0	the changeover is bumpless. **Note:** If ti = 0 the outbias parameter is re-calculated. The OUT values can thus re-start bumpless beginning with the last value of the previous operating mode.
bump = 1	the changeover may have a bump.

Detailed equations

Overview

The detailed equations are shown for the following situations are shown in this section:
- Convention for the most important Interim variables and Functions used in the equations
- *Absolute algorithm, p. 355*
- *Incremental algorithm PID controller, p. 356*
  - Normal incremental algorithms (aw_type = 0)
  - With bumpless antiwindup measure (aw_type = 1)
- *Incremental algorithms in integral mode, p. 358*
  - Normal incremental algorithms (aw_type = 0)
  - With bumpless antiwindup measure (aw_type = 1)

Convention

Various variables and functions are used in the following equations. The variables corresponding to the parameters of the function block are not re-described.

The most important Interim variables and the Functions used are described in the following tables.

Explanation of the interim variables

An explanation of the most important interim variables can be found here.

Interim variable	Meaning
DEV_WGH	DEV_WGH = PV - (1 - ovs_att) * SP
dt	Time elapsed since the last function block execution.
K	Gain of the integral and differential components. The gain varies according to the structure of the function block (mixed or parallel) and depends on whether the proportional component is assigned or not. • If mix_par = 0 (mixed structure) and kp <> 0, K = kp applies • If mix_par = 1 (parallel structure) or kp – 0, the following applies:) K = scaling factor = $\alpha = \frac{out\_sup - out\_inf}{pv\_sup - pv\_inf}$
(new)	Value which is calculated on current execution of the function block
(old)	Value which is calculated on previous execution of the function block
OUTc	Before limitation of calculated output value
sense	Control setting
TermAW	Value of the bumpless antiwindup measure
TermD	Value of the differential component
TermFF	Value of the feed forward component (disturbance compensation)
TermI	Value of the integral component
TermP	Value of the proportional component
VAR	To calculate the variable used by the differential component. Its value depends on the pv_dev parameter : • If pv_dev = 0, VAR = PV • If pv_dev = 1, VAR = dev

Explanation of the functions

An explanation of the most important functions can be found here.

Function	Meaning
Control setting	The control setting has the following directions of action: • +1 This is a direct action (rev_dir = 0,) i.e. a positive deviation (PV - SP) generates an increase in the output value • -1 This is an inverse action (rev_dir = 1,) i.e. a positive deviation (PV - SP) generates decrease in the output value.
Function Δ	$\Delta(x(t)) = x(t) - x(t-1)$
'Limit'	Limiting function for the function block output

Absolute algorithm

The following equations apply for PD controllers (ti = 0),

$$OUT = TermP + TermD + TermFF + outbias$$

$$OUTD = OUTP(new) - OUTP(old)$$

$$OUT = limiter(OUT)$$

Value of the proportional component TermP

$$TermP = sense \times kp \times dev$$

Value of the differential component TermD

$$TermD = sense \times \frac{td \times TermD_{(old)} + K \times td \times kd \times (VAR_{(new)} - VAR_{(old)})}{kd \times dt + td}$$

Value of the feed forward component TermFF

$$TermFF = \frac{(FF - ff\_inf) \times (otff\_sup - otff\_inf)}{ff\_sup - ff\_inf} + otff\_inf$$

Detailed equations: Incremental algorithm PID controller

Incremental algorithm PID controller

For the PID controller (ti > 0), the equations are divided into the following categories, depending on the aw_type element.

Element	Meaning
aw_type = 0	Normal incremental algorithms
aw_type = 1	With bumpless antiwindup measures

PID controller aw_type = 0

The following equations apply to normal incremental algorithms of PID controllers;

$$OUTD = TermP + TermI + TermD + TermFF$$

$$OUT = limiter(OUT)$$

If en_rcpy = 0, then

$$OUT = OUT(old) + OUTD(new)$$

If en_rcpy = 1, then

$$OUT = RCPY + OUTD(new)$$

Value of the proportional component TermP:

$$TermP = sense \times kp \times [\Delta(DEV\_WGH)]$$

Value of the integral component TermI:

$$TermI = sense \times kp \times \frac{dt}{ti} \times dev$$

Value of the differential component TermD

$$TermD = \Delta\left[sense \times \frac{td \times TermD_{(old)} + K \times td \times kd \times (VAR_{(new)} - VAR_{(old)})}{kd \times dt + td}\right]$$

Value of the feed forward component TermFF

$$TermFF = \Delta\left[\frac{(FF - ff\_inf) \times (otff\_sup - otff\_inf)}{(ff\_sup - ff\_inf)} + otff\_inf\right]$$

PID controller aw_type = 1

The following equations apply to incremental algorithms of PID controllers with bumpless antiwindup measures;

$$\text{OUTD} = \text{TermP} + \text{TermI} + \text{TermD} + \text{TermFF} + \text{TermAW}$$

$$\text{OUTc} = \text{OUTc(old)} + \text{OUTD(new)}$$

$$\text{OUT} = \text{limiter(OUTc)}$$

Value of the proportional component TermP:

$$\text{TermP} = \text{sense} \times \text{kp} \times [\Delta(\text{DEV\_WGH})]$$

Value of the integral component TermI:

$$\text{TermI} = \text{sense} \times \text{kp} \times \frac{\text{dt}}{\text{ti}} \times \text{dev}$$

Value of the differential component TermD

$$\text{TermD} = \Delta\left[\text{sense} \times \frac{\text{td} \times \text{TermD}_{(\text{old})} + \text{K} \times \text{td} \times \text{kd} \times (\text{VAR}_{(\text{new})} - \text{VAR}_{(\text{old})})}{\text{kd} \times \text{dt} + \text{td}}\right]$$

Value of the feed forward component TermFF

$$\text{TermFF} = \Delta\left[\frac{(\text{FF} - \text{ff\_inf}) \times (\text{otff\_sup} - \text{otff\_inf})}{(\text{ff\_sup} - \text{ff\_inf})} + \text{otff\_inf}\right]$$

Value of the bumpless antiwindup measure TermAW

If en_rcpy = 0, then

$$\text{TermAW} = \frac{\text{dt}}{\text{ti}}[\text{OUT(old)} - \text{OUTc(old)}]$$

If en_rcpy = 1, then

$$\text{TermAW} = \frac{\text{dt}}{\text{ti}}[\text{RCPY} - \text{OUTc(old)}]$$

Detailed equations: Incremental algorithms in integral mode

Incremental algorithms in integral mode

The controller can be set to a purely integral mode (with kp=0).

Here too, the equations are divided into the following categories, depending on the aw_type element:

Element	Meaning
aw_type = 0	Normal incremental algorithms
aw_type = 1	With bumpless antiwindup measures

Integral mode: aw_type = 0

The following equations apply to normal incremental algorithms of controllers in integral mode;

$$\mathrm{OUTD} = \mathrm{TermI} + \mathrm{TermFF}$$

$$\mathrm{OUT} = \mathrm{limiter(OUT)}$$

If en_rcpy = 0, then

$$\mathrm{OUT} = \mathrm{OUT(old)} + \mathrm{OUTD(new)}$$

If en_rcpy = 1, then

$$\mathrm{OUT} = \mathrm{RCPY} + \mathrm{OUTD(new)}$$

Value of the integral component TermI:

$$\mathrm{TermI} = \mathrm{sense} \times \alpha \times \frac{\mathrm{dt}}{\mathrm{ti}} \times \mathrm{dev}$$

Value of the feed forward component TermFF

$$\mathrm{TermFF} = \Delta\left[\frac{(\mathrm{FF} - \mathrm{ff\_inf}) \times (\mathrm{otff\_sup} - \mathrm{otff\_inf})}{(\mathrm{ff\_sup} - \mathrm{ff\_inf})} + \mathrm{otff\_inf}\right]$$

Integral mode: aw_type = 1

The following equations apply to incremental algorithms of integral controllers with bumpless antiwindup measures;

$$OUTD = TermI + TermFF + TermAW$$

$$OUTc = OUTc(old) + OUTD(new)$$

$$OUT = limiter(OUTc)$$

Value of the integral component TermI:

$$TermI = sense \times \alpha \times \frac{dt}{ti} \times dev$$

Value of the feed forward component TermFF

$$TermFF = \Delta\left[\frac{(FF - ff\_inf) \times (otff\_sup - otff\_inf)}{(ff\_sup - ff\_inf)} + otff\_inf\right]$$

Value of the bumpless antiwindup measure TermAW

If en_rcpy = 0, then

$$TermAW = \frac{dt}{ti}[OUT(old) - OUTc(old)]$$

If en_rcpy = 1, then

$$TermAW = \frac{dt}{ti}[RCPY - OUTc(old)]$$

Example for the PIDFF block

Example-overview

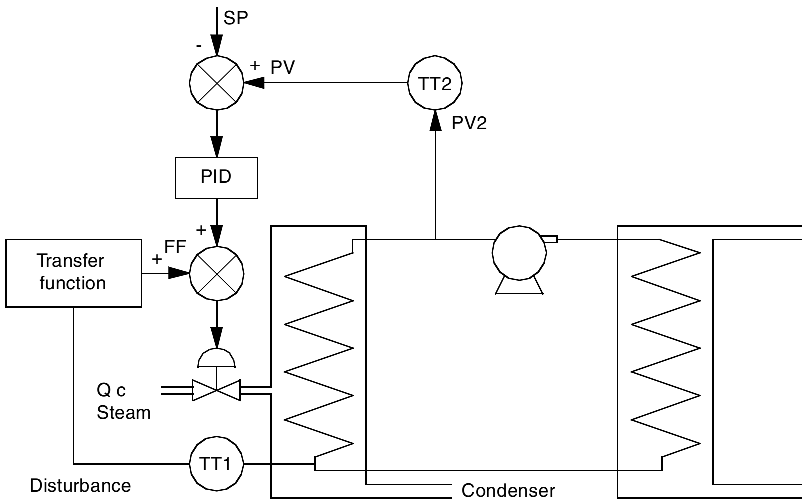
This chapter contains the following examples:
- *Application example of the feed forward function , p. 360*
- Classic control examples programmed via the PIDFF function block:
  - *Example of the cascaded arrangement of two controllers, p. 362*
  - *Example of cascade-like control, p. 364*

Application example of the feed forward function

With a heat exchanger, the temperature PV2 should be regulated at the output of the secondary circulation. A PID controller controls the inflow valve for warm air depending on PV2 and the setpoint SP. The cold water temperature is regarded as a measurable disturbance variable in this control process.

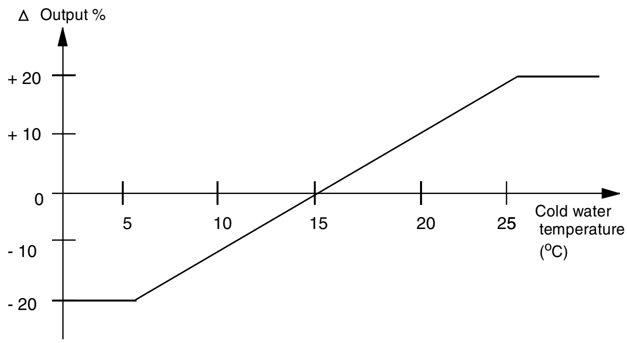
The feed forward function means a reaction can occur as soon as the cold water temperature changes without waiting for PV2 to decrease.

Presentation of the servo loop:



The following hypotheses are accepted:
- The condenser output temperature (cold water temperature) varies between 5 C and 25 C, with a mean value of 15 C.
- A DT temperature change has a full effect on the output temperature of the heat exchanger.
- To compensate for a temperature increase (or decrease) by 5 C at the output of the heat exchanger, the steam control valve must be closed (or opened) by 10 %.

The feed forward input parameters should be adjusted so that the cold water temperature has the following effect on the steam control valve:

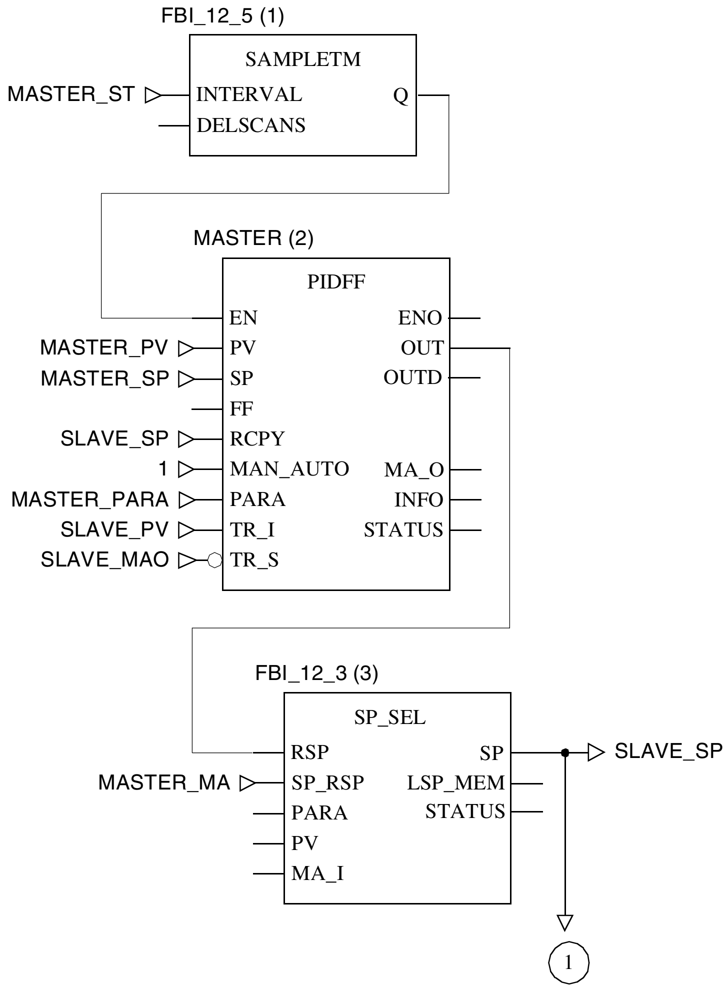
Temperature range	Effects
15 C	no effect
10% per 5 °C between 5 and 25 °C	

Adjustments to be pre-set

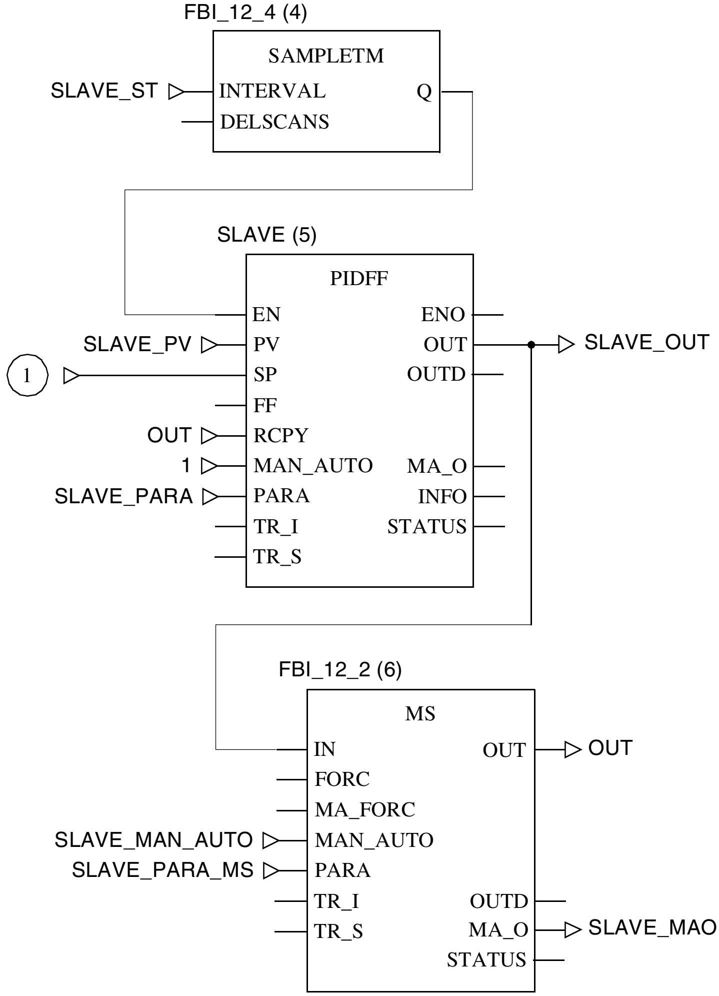
Element	Value
ff_sup	25 °C
ff_inf	5 °C
otff_sup	10 %
otff_inf	- 10 %

Example of the cascaded arrangement of two controllers

A representation of the function map, part 1, follows:

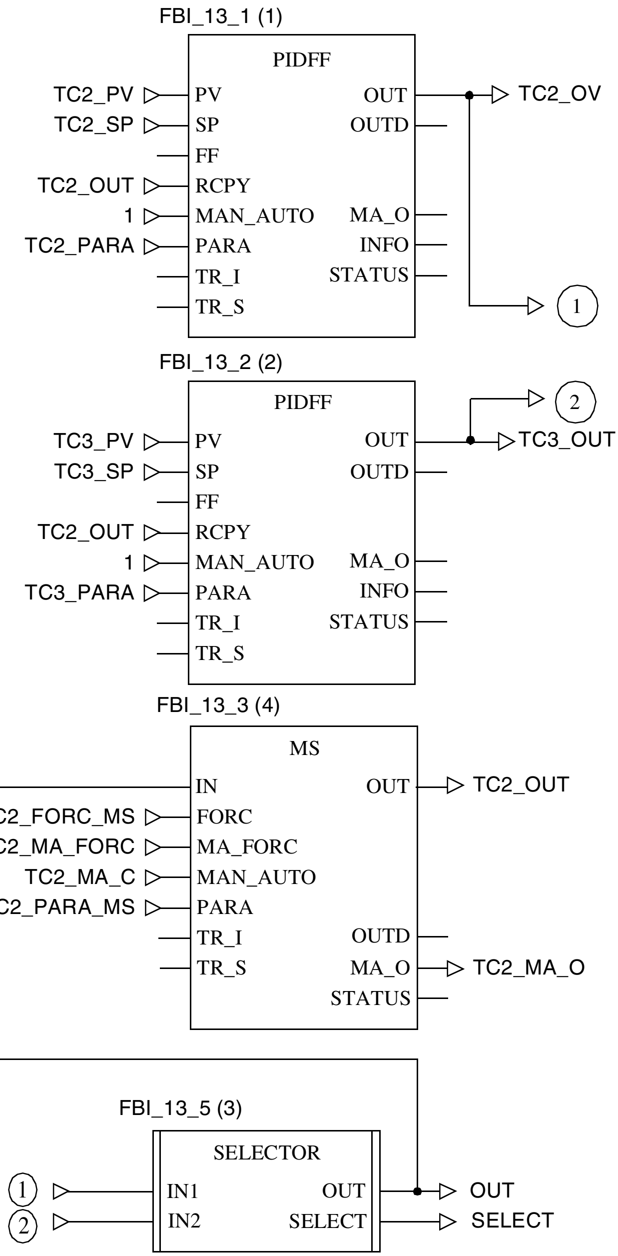


A representation of the function map, part 2, follows:



Example of cascade-like control

A representation of the function map follows:



Runtime error

Status word

The following messages are displayed in the status word:

Bit	Meaning
Bit 0 = 1	Error in a calculation in floating point values
Bit 1 = 1	Recording of an unauthorized value on a floating point value input
Bit 2 = 1	Division by zero with calculation in floating point values
Bit 3 = 1	Capacity overflow with a calculation in floating point values
Bit 4 = 1	The following behavior is displayed: • The SP input lies outside the area [pv_inf, pv_sup] : for calculation, the function block uses value pv_inf or pv_sup. • One of the kp, dband, gain _kp parameters outrate is negative. the function block uses the value 0 outside the incorrect parameter value. • kd < 1 (mit td <> 0) : the function block uses the value 1 instead of the faulty value of kd. • The parameter ovs_att is outside the [0, 1] range: for calculation, the function block uses the value 0 or 1. • One of the parameters out_min or out-max is outside the range [out_inf, out_sup]. For calculation, the function block uses the value out_inf or out sup. • One of the outbias, otff_inf or otff_sup parameters is outside the range [(out_min – out_max), (out_max – out_min)]. For calculation, the function block uses the value (out_min- out_max) i.e. (out_max - out_min).
Bit 5 = 1	The output OUT has reached the lower threshold out_min (see Note)
Bit 6 = 1	The output OUT has reached the upper threshold out_max (see Note)
Bit 7 = 1	The thresholds pv_inf and pv_sup are identical.

Note on output OUT

Note: In manual mode these bits stay at 1 for only one program cycle. When the user enters a value for OUT which exceeds one of the thresholds, the function block sets the Bit 5 or 6 to 1 and cuts them from the user entered value. During the next execution of the function block, the value of OUT no longer lies outside the range and bits 5 and 6 are set to zero again.

Error message

An error is displayed when a non-floating point has been recorded at an input, when a problem occurs during a calculation with floating points or when the thresholds pv_inf and pv_sup of the controller are identical. In this case the outputs OUT, OUTD, MA_O and INFO remain unchanged.

Warning

In the following cases a warning is given:

- One of the kp, dband, gain _kp parameters outrate is negative. The function block then uses the value 0 instead of the incorrect parameter value.
- kd < 1 (mit td <> 0) : the function block uses the value 1 instead of the faulty value of kd.
- The parameter ovs_att is outside the [0, 1] range: for calculation, the function block uses the value 0 or 1.
- The parameters out_min or out_max is outside the range [out_inf, out_sup]. For calculations, the function block uses the value out_inf or out_sup.
- One of the outbias, otff_inf or otff_sup parameters is outside the range [(out_min – out_max), (out_max – out_min)]. For calculation, the function block uses the value (out_min- out_max) i.e. (out_max - out_min).

PIDP1: PID controller with parallel structure

40

Overview

At a glance

This chapter describes the PIDP1 block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block replicates a PID controller in parallel structure.

A system deviation ERR is formed by the difference between the setpoint SP and the controlled variable PV. This deviation brings about a modification to the manipulated variable Y.

EN and ENO can be configured as additional parameters.

Properties

The function block has the following properties:
- PID controller in pure parallel structure
- Each component P, I and D can be individually enabled
- Limiting control limits in automatic mode
- Antiwindup measure with an active I component only
- Antiwindup reset
- Operating modes, Manual, Halt, Automatic
- bumpless changeover between manual and automatic
- D component can be based on input variable PV or system deviation ERR
- D component with variable delay

Transfer function

The transfer function is:

$$G(s) = \underbrace{KP}_{YP} + \underbrace{\frac{KI}{s}}_{YI} + \underbrace{\frac{KD \times s}{s + \frac{1}{TD\_LAG}}}_{YD}$$

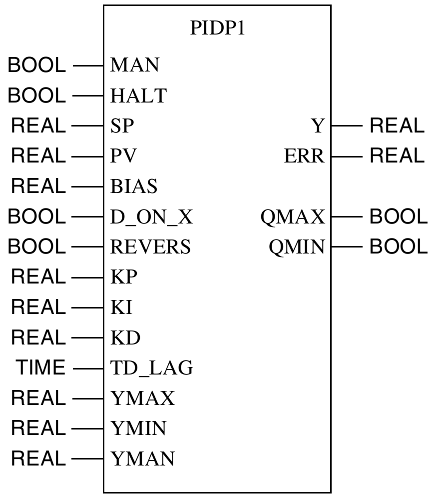
Explanation of the sizes:

Variable	Meaning
YD	D component
YI	I component
YP	P component

Representation

Symbol

Block representation:



Parameter description

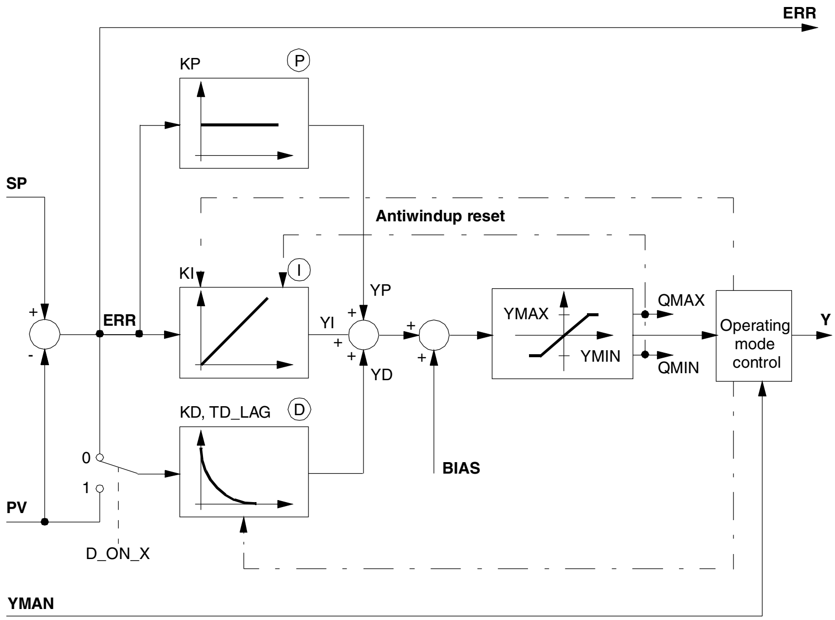
Block parameter description

Parameter	Data type	Meaning
MAN	BOOL	"1": Manual mode
HALT	BOOL	"1": Halt mode
SP	REAL	Setpoint input
PV	REAL	Input variable
BIAS	REAL	Disturbance input
D_ON_X	BOOL	"1": D component in relation to the controlled variable, "0": D component in relation to the system deviation
REVERSE	BOOL	"1": Output reversed
KP	REAL	Proportional action coefficient (gain)
KI	REAL	Integral action coefficient]
KD	REAL	Rate of differentiation]
TD_LAG	TIME	D component delay time
YMAX	REAL	Upper limit
YMIN	REAL	Lower limit
YMAN	REAL	Manually manipulated value
Y	REAL	Manipulated variable
ERR	REAL	System deviation
QMAX	BOOL	"1" = Y has reached upper limit
QMIN	BOOL	"1" = Y has reached lower limit

Parametering of the PIDP1 controller

Structure diagram

The following is the structure diagram of the PIDP1 block:



Parametering

The PIDP1 controller structure is displayed in the *Structure diagram, p. 371*.

The parametering of the PIDP1 controller initially occurs through the pure PID parameters, i.e. the proportional action coefficient KP, the integral action coefficient KI and the rate of differentiation KD.

The P, I and D components can be individually disabled while the corresponding input (KP, KI or KD) is set to 0.

The D component is delayed by the delay time TD_LAG. The D component can either be formed by the system deviation ERR (D_ON_X = "0") or the controlled variable PV (D_ON_X = "1"). Should the D component be determined by the controlled variable PV, then the D component does not cause jumps when reference variable fluctuations (changes in input SP) occur. In principle, the D component only affects disturbances and process variances.

Control direction reversal

The opposite behavior of the controller can be attained by setting input REVERSE to 1. REVERSE = 0 results in an increased output value when there is a positive disturbance. REVERSE = 1 results in an decreased output value when there is a positive disturbance.

Manipulated variable limiting

The limits YMAX and YMIN retain the output within the prescribed range. Hence, YMIN ≤ Y ≤ YMAX.

The outputs QMAX and QMIN signal that the output has reached a limit, and thus been capped.

- QMAX = 1 if Y ≥ YMAX
- QMIN = 1 if Y ≤ YMIN

The upper limit YMAX, limiting the manipulated variable, is to be set higher than the lower limit YMIN.

Antiwindup reset

If manipulated variable limiting takes place, the antiwindup reset should ensure that the I component "cannot go berserk". Antiwindup measures are taken only for an active I component. Antiwindup limits are identical to those for manipulated variable limiting. The antiwindup measures disregard D component values, to avoid being falsely triggered by D component peaks.

The antiwindup measures correct the I component in such a way that:

$$\mathrm{YMIN} - \mathrm{YP} - \mathrm{BIAS} \leq \mathrm{YI} \leq \mathrm{YMAX} - \mathrm{YP} - \mathrm{BIAS}$$

Selecting the controller types

Several controller variants can be selected via the parameters KP, KI and KD.

Controller type	KP	KI	KD
P controller	> 0	= 0	= 0
PI controller	> 0	> 0	= 0
PD controller	> 0	= 0	> 0
PID controller	> 0	> 0	> 0
I controller	= 0	> 0	= 0

Operating modes

Selecting the operating modes

There are three operating modes which are selected via the parameters MAN and HALT:

Operating mode	MAN	HALT
Automatic	0	0
Manual	1	0 or 1
Halt	0	1

Automatic mode

In automatic mode the control output Y is determined through the discrete PID closed-loop control algorithm, based on the controlled variable PV and reference variable SP. The control output is limited with YMAX and YMIN. The control limits are also limits for the Antiwindup reset.

The changeover from automatic to manual is normally not bumpless, since output Y can take on any value between YMAX and YMIN, and Y goes directly to YMAN at the changeover.

If the changeover from automatic to manual is to be bumpless in spite of this, there are two exemplary possibilities shown for a PID1 Controller (see *Switching from automatic to manual, p. 316*).

Manual mode

In manual mode the manually manipulated value YMAN is passed on directly to the control output Y. The control output is, however, limited by YMAX and YMIN. Internal variables will be manipulated in such a manner that the controller changeover from manual to automatic (with I component enabled) can be bumpless. The control limits are also limits for the Antiwindup reset.

In this operating mode the D component is automatically set to 0.

Halt mode

In halt mode the control output remains unchanged; the function block does not influence the manipulated variable Y, i.e. Y = Y(old). Internal variables will be manipulated in such a manner that the component sum corresponds to the control output, thus allowing the controller to be driven smoothly from its current position (when the I component is enabled). The control limits are also limits for the Antiwindup reset.

In this operating mode the D component is automatically set to 0.

Detailed formulas

Explanation of formula variables

Meaning of the variables in the formulae:

Variable	Meaning
dt	Time differential between the present cycle and the previous cycle
ERR	System deviation (SP - PV)
$ERR_{(new)}$	System deviation value from the current sampling step
$ERR_{(old)}$	System deviation value from the previous sampling step
BIAS	Disturbance
$PV_{(new)}$	Value of controlled variable from the current sampling step
$PV_{(old)}$	Value of controlled variable from the previous sampling step
Y	current output (halt mode) or YMAN (manual mode)
YD	D component
YI	I-component
YP	P-component

Manipulated variable

The manipulated variable is composed of various terms:

$$Y = YP + YI + YD + BIAS$$

After the summation of the components a manipulated variable limiting takes place at the output of the sub controller, which means:

$$YMIN \le Y \le YMAX$$

System deviation

The system deviation is determined as follows:

If	Then
REVERS = 0	ERR = SP - PV
REVERS = 1	ERR = PV - SP

Overview to calculate the control components

Following this an overview on the different calculations of the control components in relation to the gains KP, KI and KD can be found:
- P component YP for manual, halt and automatic modes
- I component YI for automatic mode
- I component YI for manual and halt modes
- D component YD for automatic mode
- D component YD for manual and halt modes

P component YP for all operating modes

YP for manual, halt and automatic modes are determined as follows

$$YP = KP \times ERR$$

I component YI for automatic mode

YI for automatic mode is determined as follows:

For KI > 0 applies:

$$YI_{(new)} = YI_{(old)} + KI \times dt \times \frac{ERR_{(new)} + ERR_{(old)}}{2}$$

For KI = 0 the following applies

$$YI = 0$$

The I-component is formed according to the trapazoid rule.

I component YI for manual and halt modes

YI for manual, halt and automatic modes is determined as follows:

For KI > 0 applies:

$$YI = Y - (YP - BIAS)$$

For KI = 0 the following applies

$$YI = 0$$

D component YD for automatic mode

YD for automatic mode and cascade is determined as follows:

For KD > 0 and D_ON_X = 0 the following applies:

$$YD_{(new)} = \frac{TD\_LAG}{dt + TD\_LAG} \times (YD_{(old)} + KD \times (ERR_{(new)} - ERR_{(old)}))$$

For KD > 0 and D_ON_X = 1 the following applies:

$$YD_{(new)} = \frac{TD\_LAG}{dt + TD\_LAG} \times (YD_{(old)} + KD \times (PV_{(old)} - PV_{(new)}))$$

For KD = 0 the following applies

$$YD = 0$$

D component YD for manual and halt modes

YD for manual, halt and automatic modes is determined as follows:

$$YD = 0$$

Runtime error

Error message For YMAX < YMIN an Error message appears.

PIP: PIP cascade controller

41

Overview

At a glance

This chapter describes the PIP block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block displays a cascade-controller, consisting of a PI-master controller and a P-sub controller.

The system deviation is formed between the SP reference variable and the PV controlled variable.

The master controller generates a sub controller setpoint value SP2 through this system deviation. Due to the difference between SP2 and PV2 the sub controller generates the manipulated variable Y.

The parameters EN and ENO can be additionally projected.

Properties

The function block contains the following properties:
- PI as master controller and P as sub controller
- Manipulated variable limiting
- Antiwindup-Reset for the PI controller
- Operating mode, fixed setpoint control, manual, halt, automatic

Transfer function

The transmission function for the controller says:

Controller	Transfer function
Master controller (PI-controller)	$G(s) = gain1 \times \left(1 + \frac{1}{ti \times s}\right)$
Sub controller (P controller)	$G(s) = gain2$

Proportional action coeffiecient

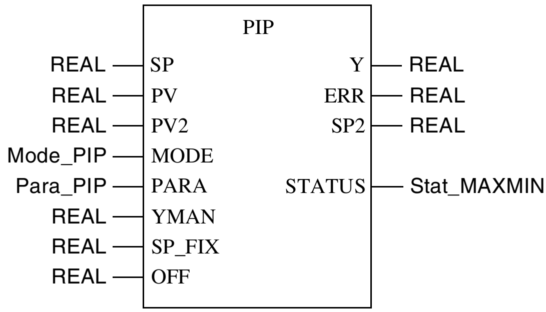
The proportional action coefficient of the master controller is determined as follows:

$$YP = gain1 \times ERR$$

Display

Symbol

Block display



PIP parameter description

Block parameter description

Parameter	Data type	Meaning
SP	REAL	Reference variable
PV	REAL	Controlled variable for the master controller
PV2	REAL	Controlled variable for the sub controller (auxiliary control variable)
MODE	Mode_PIP	Operating mode
PARA	Para_PIP	Parameter
YMAN	REAL	Manual value (of output Y)
SP_FIX	REAL	Fixed value (reference variable as manual value for the sub controller)
OFF	REAL	Offset at the output of the P-controller
Y	REAL	Manipulated variable
ERR	REAL	System deviation
SP2	REAL	Sub controller setpoint value
STATUS	Stat_MAXMIN	Status of output Y

Parameter description Mode_PIP

Data structure description

Element	Data type	Meaning
man	BOOL	"1": Manual mode
halt	BOOL	"1": Halt mode
fix	BOOL	"1": Fixed setpoint control

Parameter description Para_PIP

Data structure description

Element	Data type	Meaning
gain1	REAL	Proportional action coefficient (gain) for PI controller
ti	TIME	PI controller reset time
gain2	REAL	Proportional action coefficient (gain) for P controller
ymax	REAL	Upper limit
ymin	REAL	Lower limit

Parameter description Stat_MAXMIN

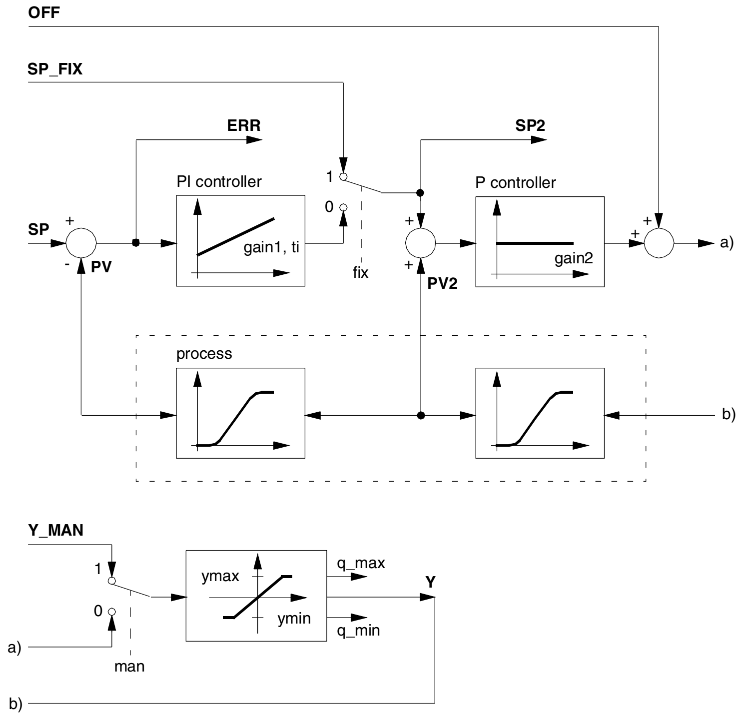
Data structure description

Element	Data type	Meaning
qmax	BOOL	"1" = Y reached upper limit
qmin	BOOL	"1" = Y reached lower limit

Structure diagram of the PIP function block

Structure diagram

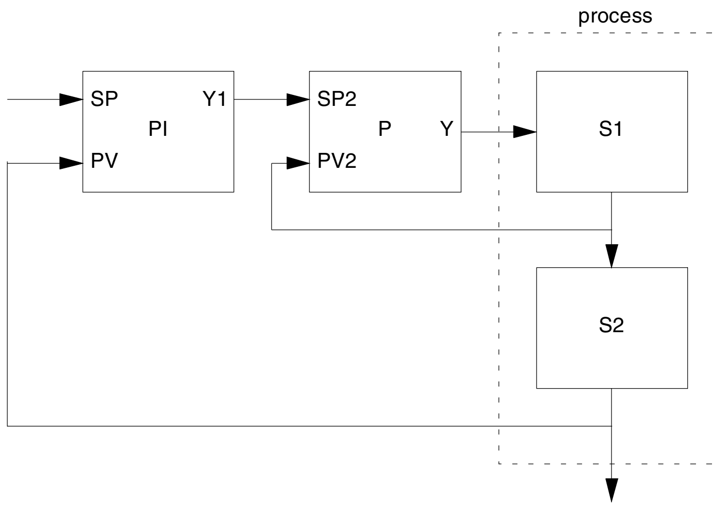
There follows now the structure diagram of the PIP block:



Parametering of the PIP-cascade controller

Modular mimic display

Modular mimic display of the PIP-cascade controller



Parametering

The structure of the PIP controller is demonstrated in the *Modular mimic display, p. 382* .

The parametering of the function block takes place firstly through the pure PI – parameter, that is to say the proportional correction value (gainl) and the reset time (ti).

The I-component can be disabled by setting the ti to zero.

Subsequently the parametering of the P controller takes place through the proportional correction value gain2.

Manipulated variable limiting

Manipulated variable limiting takes place at the output of the sub controller, which means:

ymin ≤ Y ≤ ymax

Antiwindup-Reset (PI controller)

If manipulated variable limiting takes place, the antiwindup reset should make sure that the integral component of the master controller "is not able to exceed all limits". The antiwindup measure can only be used if the I-component of the controller is not disabled.

The antiwindup limits for the PI master controller are adjusted dynamically to the present system deviation of the sub controller and the ymax and ymin limits.

If manipulated variable limiting takes place, the integral component will be limited as follows:

- on reaching the upper limit:

$$YI = \left(\frac{ymax - OFF}{gain2} + PV\right) - YP$$

- on reaching the lower limit:

$$YI = \left(\frac{ymin - OFF}{gain2} + PV\right) - YP$$

Operating mode

Choice of operating mode

There are four operating mode, which are selected via the elements man, halt and fix:

Operating mode	man	halt	fix
Automatic	0	0	0
Hand	1	0 or 1	0
Halt	0	1	0
Fixed setpoint control	0	0	1

Automatic mode

In the automatic mode, the control output Y is determined through the PI closed-loop control, based on the controlled variables PV, PV2 and the reference variables SP, SP2. The control output is limited through ymax and ymin.

The changeover from automatic to manual is normally not bumpless, since output Y can take on any value between ymax and ymin, and yet goes directly to YMAN at the changeover.

If the changeover from automatic to manual is to be bumpless despite these problems, there are two exemplary possibilities shown for a PID controller (see *Switching from automatic to manual, p. 302*).

Manual mode

The P controller works in manual mode. The PI controller I component is manipulated to permit bumpless switching.

In the manual mode the manual manipulated value YMAN is passed on directly to the control output Y. The control output is, however, limited through ymax and ymin. the integral component of the master controller is tracked in such a way that the controller (on connecting to the I-component) can be switched bumplessly from manual to automatic.

Halt mode

In halt mode the control output remains unchanged; the function block does not influence the control output Y. Halt mode is also useful in allowing an external operator device to adjust control output Y the internal components are so manipulated that the controller can be driven smoothly from it’s current position. The control output is, however, limited through ymax and ymin.

Fixed setpoint control

In fixed setpoint control mode the P controller works in automatic mode and the PI-controller works in halt mode.

The fixed setpoint SP_FIX is passed on directly to the control output of the PI controller Y1 (=SP2). The control output of the PIP controller Y is limited through ymax and ymin. The integral component of the master controller is tracked in such a way that the controller (on connecting to the I-component) can be switched smoothly from fixed setpoint control to automatic.

Detailed formulas

Explanation of the formula sizes

Significance of the size in the following formulas:

Size	Meaning
dt	Time differential between the present cycle and the previous cycle
ERR	System deviation (SP - PV)
$ERR_{(new)}$	System deviation value from the current sampling step
$ERR_{(old)}$	System deviation value from the current sampling step
OFF	Offset at the output of the P-controller
Y	Manipulated variable
Y1	Y of the master controller
YI	I-component
YP	P-component

Overview to calculate the control components

There now follows an overview of the varying calculations on control components and outputs for the various modes:
- YI, Y, SP2 in the automatic mode
- YI, Y, SP2 in the manual mode
- YI, Y, SP2 in the manual mode
- YI, Y, SP2 in the fixed setpoint control mode

Automatic mode

The output signal Y of the cascade controller is:

$$Y = (SP2 - PV2) \times gain2 + OFF$$

The input signal SP2 of the sub controller is:

$$SP2 = YP + YI$$

The integral component Y1 of the master controller for the automatic mode is determined as follows:

$$YI_{(new)} = YI_{(old)} + gain1 \times \frac{dt}{ti} \times \frac{ERR_{(new)} + ERR_{(old)}}{2}$$

The I-component is formed according to the trapazoid rule.

Manual mode

The output signal Y of the cascade controller is:

$$Y = YMAN$$

The input signal SP2 of the sub controller is:

$$SP2 = \frac{Y - OFF}{gain2} + PV2$$

The integral component Y1 of the master controller for the manual mode is determined as follows:

$$YI = SP2 - (SP - PV) \times gain1$$

Halt mode

The output signal Y of the cascade controller is:

$$Y = Y_{(old)}$$

The input signal SP2 of the sub controller is:

$$SP2 = \frac{Y - OFF}{gain2} + PV2$$

The integral component Y1 of the master controller for the halt mode is determined as follows:

$$YI = SP2 - (SP - PV) \times gain1$$

Fixed setpoint control

The output signal Y of the cascade controller is:

$$Y = (SP2 - PV2) \times gain2 + OFF$$

The input signal SP2 of the sub controller is:

$$SP2 = SP\_FIX$$

The integral component Y1 of the master controller for the fixed setpoint control mode is determined as follows:

$$YI = SP2 - (SP - PV) \times gain1$$

Runtime error

Error message

An error message, appears if

- an invalid floating point number lies at input PV, PV2, YMAN or SP_FIX.
- is ymax < ymin.

PPI: PPI cascade controller

42

Overview

At a glance

This chapter describes the PPI block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block displays a cascade-controller, consisting of a P-master controller and a PI-sub controller.

The system deviation is formed between the SP reference variable and the PV controlled variable.

The master controller generates a sub controller setpoint value SP2 through this system deviation. Due to the difference between SP2 and PV2 the sub controller generates the manipulated variable Y.

The parameters EN and ENO can be additionally projected.

Properties

The function block contains the following properties:
- P as master controller and PI as sub controller
- Manipulated variable limiting
- Antiwindup-Reset for the PI controller
- Operating mode, fixed setpoint control, manual, halt, automatic

Transfer function

The transmission function for the controller says:

Controller	Transfer function
Master controller (P-controller)	$G(s) = gain1$
Sub controller (PI controller)	$G(s) = gain2 \times \left(1 + \frac{1}{ti \times s}\right)$

Proportional action coeffiecient

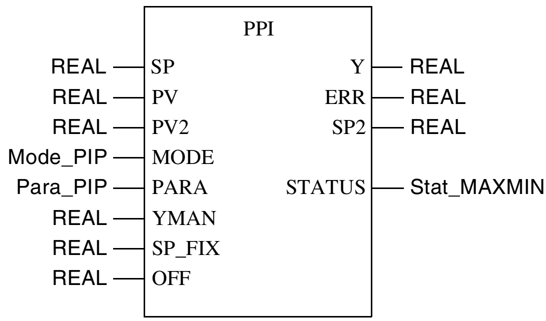
The proportional action coefficient is determined as follows:

$$YP = gain2 \times (SP2 - PV2)$$

Display

Symbol

Block display



PIP parameter description

Block parameter description

Parameter	Data type	Meaning
SP	REAL	Reference variable for the master controller
PV	REAL	Controlled variable for the master controller
PV2	REAL	Controlled variable for the sub controller (auxiliary control variable)
MODE	Mode_PPI	Operating mode
PARA	Para_PPI	Parameter
YMAN	REAL	Manual value (of output Y)
SP_FIX	REAL	Fixed value (reference variable as manual value for the sub controller)
OFF	REAL	Offset at the output of the P-controller
Y	REAL	Manipulated variable
ERR	REAL	System deviation
SP2	REAL	Sub controller setpoint value
STATUS	Stat_MAXMIN	Status of output Y

Parameter description Mode_PPI

Data structure description

Element	Data type	Meaning
man	BOOL	"1": Manual mode
halt	BOOL	"1": Halt mode
fix	BOOL	"1": Fixed setpoint control

Parameter description Para_PPI

Data structure description

Element	Data type	Meaning
gain1	REAL	Proportional action coefficient (gain) for P controller
ti	TIME	PI controller reset time
gain2	REAL	Proportional action coefficient (gain) for PI controller
ymax	REAL	Upper limit
ymin	REAL	Lower limit

Parameter description Stat_MAXMIN

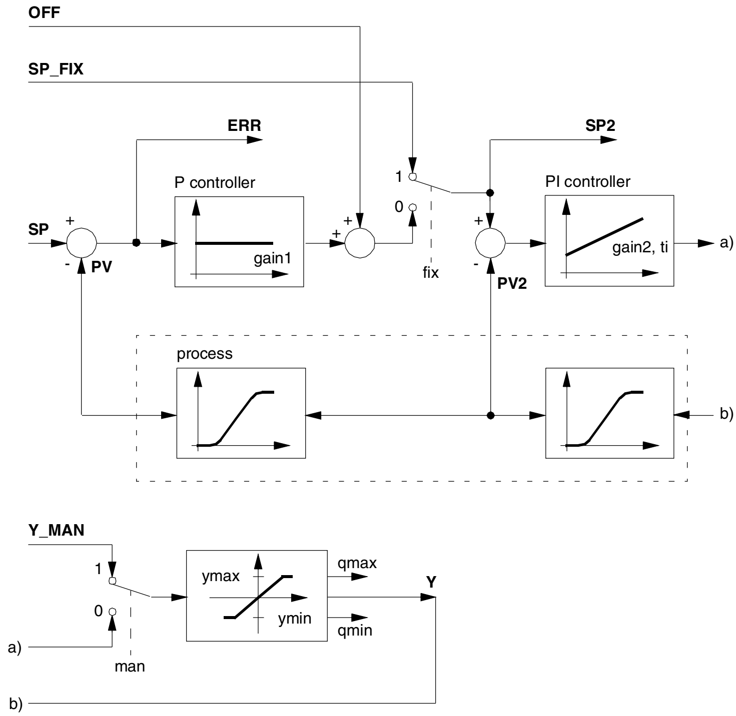
Data structure description

Element	Data type	Meaning
qmax	BOOL	"1" = Y reached upper limit
qmin	BOOL	"1" = Y reached lower limit

Structure diagram of the PPI function block

Structure diagram

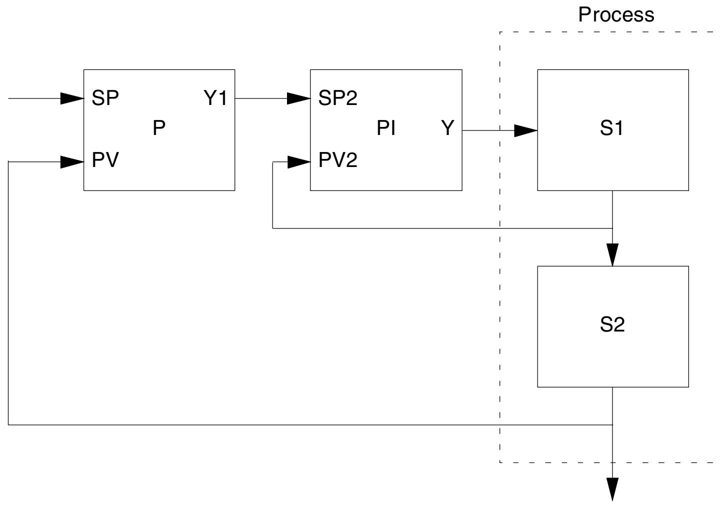
There follows now the structure diagram of the PPI block:



Parametering of the PPI-cascade controller

Modular mimic display

Modular mimic display of the PPI-cascade controller



Parametering

The structure of the PPI controller is demonstrated in the *Modular mimic display, p. 394*.

The parametering of the function block takes place firstly through the proportional correction value (gain1) and the offset for the output of the p-controller.

Subsequently the parametering of the PI controller takes place through the proportional correction value (gain2) and the reset time (ti).

The I-component can be disabled by setting the ti to zero.

The limits YMAX and YMIN limit the upper output as well as the lower output.

The outputs qmax and qmin signal that the output has reached a limit, and thus been capped.
- QMAX = 1 if Y ≥ YMAX
- QMIN = 1 if Y ≤ YMIN

Manipulated variable limiting

After the summation of the components a manipulated variable limiting takes place at the output of the sub controller, which means: ymin ≤ Y ≤ ymax

Antiwindup-Reset (PI controller)

If manipulated variable limiting takes place, the antiwindup reset should make sure that the integral component of the master controller "is not able to exceed all limits". The antiwindup measure can only be used if the I-component of the sub-controller is not disabled.

The antiwindup reset takes place if:

$Y \geq ymax$ or $Y \leq ymin$

In this case, it is:

$YI = Y - YP$

Operating mode

Choice of operating mode

There are four operating mode, which are selected via the elements man, halt and fix:

Operating mode	man	halt	fix
Automatic	0	0	0
Hand	1	0 or 1	0
Halt	0	1	0
Fixed setpoint control	0	0	1

Automatic mode

In the automatic mode, the control output Y is determined through the PI closed-loop control, based on the controlled variables PV, PV2 and the reference variables SP, SP2. The control output is limited through ymax and ymin.

The changeover from automatic to manual is normally not bumpless, since output Y can take on any value between ymax and ymin, and yet goes directly to YMAN at the changeover.

If the changeover from automatic to manual is to be bumpless despite these problems, there are two exemplary possibilities shown for a PID controller (see *Switching from automatic to manual, p. 302*).

Manual mode

In the manual mode the manual manipulated value YMAN is passed on directly to the control output Y. The control output is, however, limited through ymax and ymin. The integral sizes of the master controller are tracked in such a way that the controller (on connecting to the I-component) can be switched bumplessly from manual to automatic.

Halt mode

In halt mode the control output remains unchanged; the function block does not influence the control output Y. Halt mode is also useful in allowing an external operator device to adjust control output Y the internal components are so manipulated that the controller can be driven smoothly from it’s current position. The control output is, however, limited through ymax and ymin.

Fixed setpoint control

In this operating mode the fixed setpoint SP_FIX is passed on directly to the setpoint input of the PI controller (SP2). The PI controller works in the automatic mode.

Detailed formulas

Explanation of the formula sizes

Significance of the size in the following formulas:

Size	Meaning
dt	present sample time
ERR	System deviation (SP - PV)
$err2_{(new)}$	System deviation (SP2-PV2)
$err2_{(old)}$	System deviation value from the current sampling step
OFF	Offset at the output of the P-controller
Y	Manipulated variable
YI	I-component
YP	P-component

Master controller output

The output of the master controller is determined as follows:

$$Y1 = SP2 = gain1 \times ERR + OFF$$

Overview to calculate the control components

There now follows an overview of the varying calculations on control components and outputs based on the various modes:

- YI and Y in the automatic mode
- YI, Y, SP2 in the manual mode
- YI, Y and SP2 in the halt mode
- YI, YP, Y and SP2 in the fixed setpoint control mode

Automatic mode

The output signal Y of the cascade controller is:

$$Y = YP + YI$$

The integral component Y1 of the sub controller for the automatic mode is determined as follows:

$$YI_{(new)} = YI_{(old)} + gain2 \times \frac{dt}{ti} \times \frac{err2_{(new)} + err2_{(old)}}{2}$$

The I-component is formed according to the trapazoid rule.

Manual mode

The output signal Y of the cascade controller is:

$$Y = YMAN$$

The input signal SP2 of the sub controller is:

$$SP2 = gain1 \times (SP - PV) + OFF$$

The integral component Y1 of the sub controller for the manual mode is determined as follows:

$$YI = Y - (SP2 - PV2) \times gain2$$

Halt mode

The output signal Y of the cascade controller is:

$$Y = Y_{(old)}$$

The input signal SP2 of the sub controller is:

$$SP2 = gain1 \times (SP - PV) + OFF$$

The integral component Y1 of the sub controller for the halt mode is determined as follows:

$$YI = Y - (SP2 - PV2) \times gain2$$

Fixed setpoint control

The output signal Y of the cascade controller is:

$$Y = YP + YI$$

The input signal SP2 of the sub controller is:

$$SP2 = SP\_FIX$$

The integral component Y1 of the sub controller for the fixed setpoint control mode is determined as follows:

$$YI_{(new)} = YI_{(old)} + gain2 \times \frac{dt}{ti} \times \frac{err2_{(new)} + err2_{(old)}}{2}$$

The proportional action coefficient YP is determined as follows:

$$YP = gain2 \times (SP2 - PV2)$$

Runtime error

Error message

There is a Error message, if
- an invalid floating point number lies at input PV, PV2, YMAN or SP_FIX.
- is ymax < ymin.

PWM: Pulse width modulation

43

Overview

At a glance

This chapter describes the PWM block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Block usage

Actuators are driven not only by analog quantities, but also through binary actuating signals. The conversion of analog values into binary output signals is achieved for example, through pulse width modulation (PWM) or pulse duration modulation (PDM).

In this context, the preset mean energy level of the actuator is to correspond to the analog input value (X) of the block.

Function description

The function block PWM serves to convert analog values into digital output signals for Concept.

In pulse width modulation (PWM), a 1-signal is emitted, at a constant clock rate, for a duration that is a function of the analog value. The adjusted average energy corresponds to the quotient of the fixed duty cycle T_on and the variable cycle period.

In order that the adjusted average energy also corresponds to the analog input variable X, the following must apply:

$$T\_on \sim X$$

EN and ENO can be projected as additional parameters.

General information about the actuator drive

In general, the binary actuator drive is performed by two binary signals Y_POS and Y_NEG.

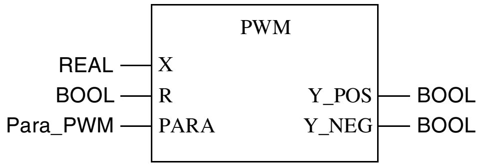
On a motor the output Y_POS corresponds to the signal "clockwise rotation" and the output Y_NEG the signal "counter-clockwise rotation". For an oven the outputs Y_POS and Y_NEG could be interpreted as corresponding to "heating" and "cooling".

Should the actuating drive in question be a motor, it is possible that to avoid overtravel for non-self-locking gearboxes, a brake pulse must be output after the engage signal. In order to protect the power electronics, there must be a pause time after switching on T_on and before the brake impulse t_brake so as to avoid short circuits.

Display

Symbol

Block display



PWM parameter description

Block parameter description

Parameter	Data type	Meaning
X	REAL	Input variable
R	BOOL	Reset mode ("1" = Reset)
PARA	Para_PWM	Parameter
Y_POS	BOOL	Positive X value output
Y_NEG	BOOL	Negative X value output

Parameter description Para_PWM

Data structure description

Element	Data type	Meaning
t_period	TIME	Length of period
t_pause	TIME	Pause time
t_brake	TIME	Braking time
t_min	TIME	Minimum actuating pulse time (in sec)
t_max	TIME	Maximum actuating pulse time (in sec)
up_pos	REAL	Upper limiting value for positive X values
up_neg	REAL	Upper limiting value for negative X values

Formulas

The pulse length for Y_POS and Y_NEG

The pulse length T_on for output Y_pos amd Y_neg is determined by the following equations:

Output	Formula	Condition
Y_POS	$T\_on = t\_period \times \frac{X}{up\_pos}$	$0 \le X \le up\_pos$
Y_NEG	$T\_on = t\_period \times \frac{\lvert X \rvert}{up\_neg}$	$up\_neg \le -X \le 0$

Parametering rules

For correct operation the following rules should be observed:

- $(2 \times t\_pause + t\_brake + t\_max) \le t\_period$
- From the parameters up_pos and up_neg only the value is evaluated.

Detailed description

Block mode of operation

The period determines the time, in which the actuating pulses ("1" signal on output Y_POS resp. Y_NEG) are regularly output, i.e. in a constant time-slot pattern.

The parameter t_min specifies the minimum pulse length, i.e. the shortest time span for which the output Y_POS and/or Y_NEG should carry "1" signal. If the length of impulse calculated according to the equation in the section "*Formulas, p. 402*" is shorter than t_min, then there will be no impulse throughout the whole period.

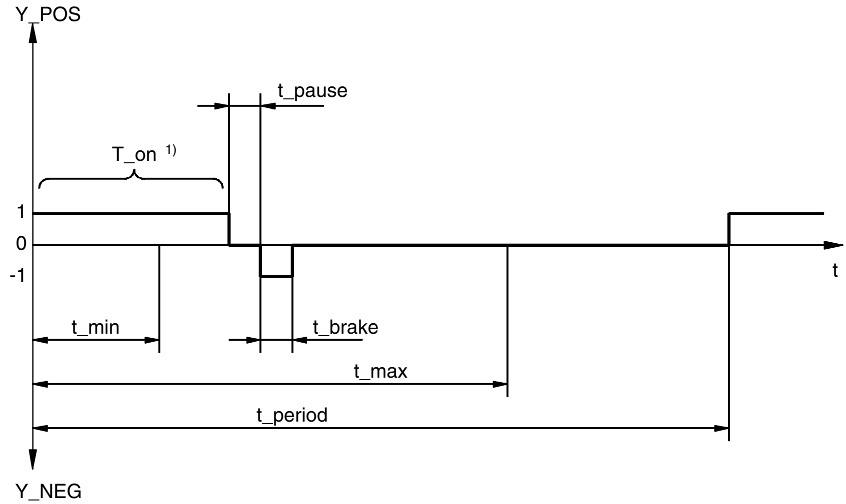
The parameter t_max specifies the minimum pulse length, i.e. the shortest time span in which the output Y_POS resp. Y_NEG should carry "1" signal. Pulse output length is then limited to t_max, should the pulse duration calculated by the above stated formula be greater. It is advisable to perform a freely definable pause time of t_pause = 10 or 20 ms between the actuating and brake pulses to protect the power electronics (hopefully preventing simultaneous firing of the antiparallel connected thyristors).

Parameter t_pause specifies the time interval that should be waited after the "1" signal on output Y_POS (Y_NEG), before the opposite output Y_NEG (Y_POS) goes to "1" signal for time span t_brake. The action in question here is a brake pulse, which should take place after the pause time. A pause time of t_pause = 20 ms (t_pause =0.02) corresponds to an interruption of the firing angle control for two half waves.

That should guarantee a sufficiently large safety margin for the prevention of short-circuits resp. triggering of the suppressor circuitry as a consequence of antiparallel thyristors firing.

Time ratios display

An overview of the ratios between times is shown in the following diagram:



1 Variable turn-on time

The parameter up_pos mark those positive values of input variable X, for which output Y_POS would continuously carry "1", assuming:

t_pause = t_brake = 0

and

t_max = t_period.

The parameter up_neg mark those positive values of input variable X, for which output Y_NEG would continuously carry "1", assuming:

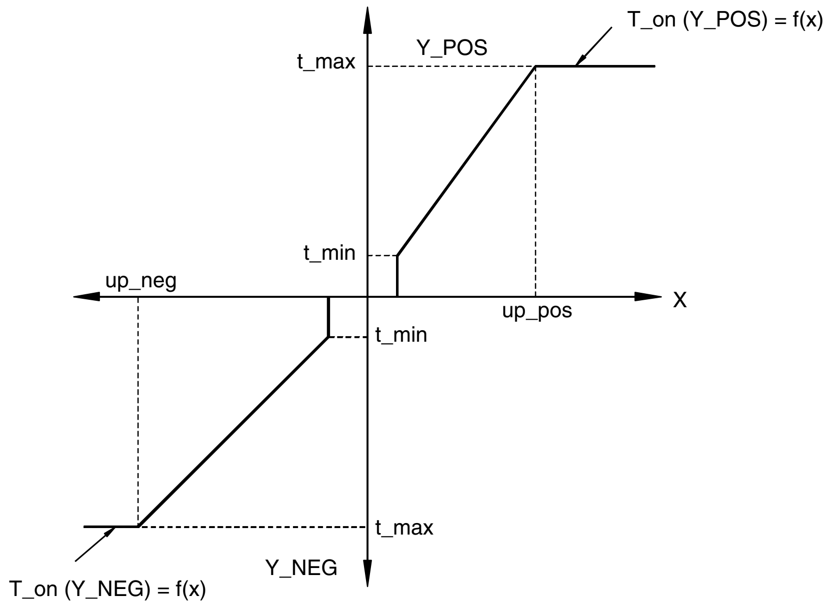
t_pause = t_brake = 0

and

t_max = t_period.

Time-span dependency

The dependency of the time duration in which the output Y_POS (Y_NEG) carries a 1-signal, on the input variable X is illustrated in the following diagram (again the figure has put t_pause = t_brake = 0)



Operating mode

In reset mode R = "1", outputs Y_POS and Y_NEG are set to "0" signal. The internal time meters are also standardized, so that the function block begins the transfer to R=0 with the output of a new 1 signal on the associated output.

Boundary conditions

If the PWM block is operated together with a PID controller, then the period t_period should be so selected, that it corresponds to the PID controller's scan time. It is then guaranteed that every new actuating signal from the PID controller within the period time can be fully processed.

The PDM scan time should be in proportion with the period vs. pulse time. Though this the smallest possible actuating pulse will be specified.

The following ratio is recommended:

t_period/scan time (PWM) $\geq$ 10

Example for the PWM block

Overview

In the examples, the signal sequences on the outputs Y_POS and Y_NEG are shown for various X input signal values. The examples differ with respect to their selected parameter assignments.

The following examples on the PMW function block are to be found in this section

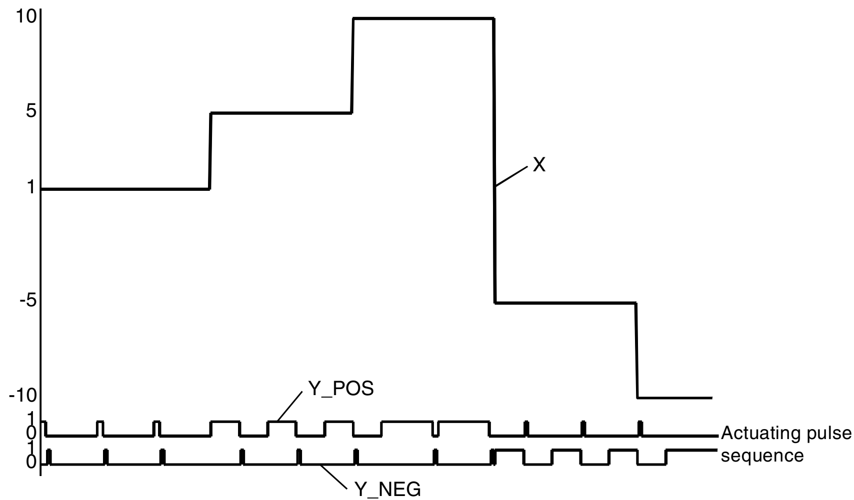
- Step Response 1
- Step Response 2

Step Response 1

The following parameter specifications apply to the step response 1 display:

Parameter	Settings
t_period	4 s
t_min	0,2 s
t_max	3,8 s
t_pause	0.1 s
t_brake	0.2 s
up_pos	10
up_neg	10

Step Response 1 timing diagram



X analog signal

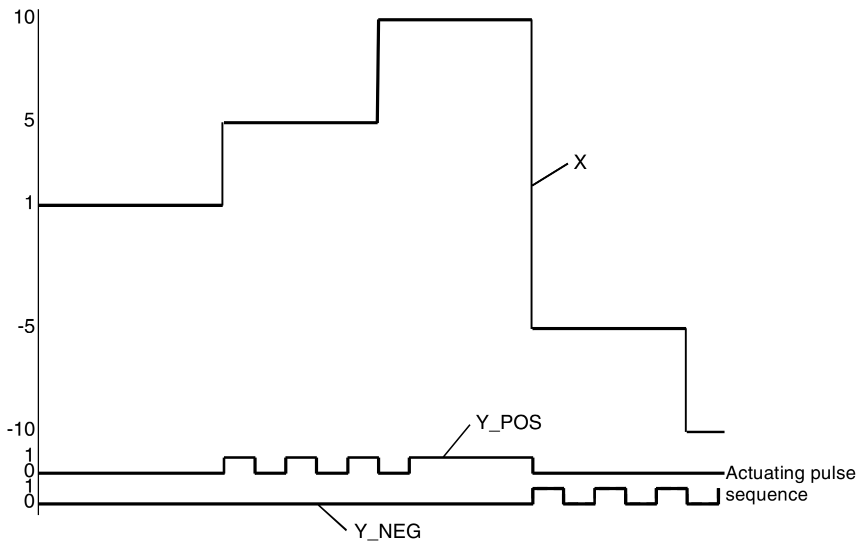
It is easily seen that the time span in which output Y_POS carries "1" signal is directly proportional to input signal X. In addition, it can be seen that a short Y_NEG-signal follows every Y_POS signal, and vice versa. This can be attributed to the non-"0" t_brake parameter. Y_NEG output time span is directly proportional to negative X input signal values. A short Y_POS pulse as brake pulse also follows the Y_NEG pulse here as well.

Step Response 2

The following parameter specifications apply to the step response 2 display:

Parameter	Settings
t_period	4 s
t_min	0.5 s
t_max	4 s
t_pause	0 s
t_brake	0 s
up_pos	10
up_neg	10

Step Response 2 timing diagram



X analog signal

The difference to the example "step response 1" is, that here the pause and brake pulses are dropped, as here the appropriate parameters were configured to "0". It is noticeable that pulses are no longer output for very small X input signals. This is directly attributable to the effect of time t_min. Moreover a continuous pulse is output for large X input signals (X = up_pos or up_neg). This is related to having selected t_max = t_period.

PWM1: Pulse width modulation

44

Overview

At a glance

This chapter describes the PWM1 block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Use of block

Actuators are driven not only by analog quantities, but also through binary actuating signals.

The actuator adjusted average energy (actuator energy) should be in accord with the modulation block's analog input value (IN).

Function description

The function block PWM1 serves to convert analog values into digital output signals for Concept.

In the pulse width modulation (PWM1) a "1" signal of variable persistence proportional to the analog value X is output within a fixed cycle period. The adjusted average energy corresponds to the quotient of the duty cycle T_on and the cycle time t_period.

In order that the adjusted average energy also corresponds to the analog input variable IN, the following must apply:

$$T\_on \sim IN$$

EN and ENO can be projected as additional parameters.

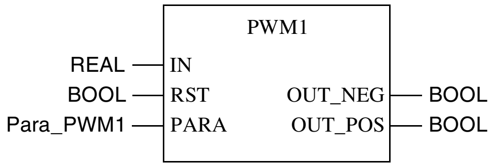
General information about the actuator drive

In general, the binary actuator drive is carried out by two binary signals OUT_POS and OUT_NEG. On a motor the output OUT_POS corresponds to the signal "clockwise rotation" and the output OUT_NEG the signal "counter-clockwise rotation". For an oven the outputs OUT_POS and OUT_NEG could be interpreted as corresponding to "heating" and "cooling".

Presentation

Symbol

Block display



PWM1 parameter description

Block parameter description

Parameter	Data type	Meaning
IN	REAL	Input variable
RST	BOOL	Reset mode ("1" = Reset)
PARA	Para_PWM1	Parameter
OUT_NEG	BOOL	Negative IN-value output
OUT_POS	BOOL	Positive IN value output

Parameter description Para_PWM1

Data structure description

Element	Data type	Meaning
t_period	TIME	Length of period
t_min	TIME	Minimum actuating pulse time
in_max	REAL	Upper limiting value for positive/negative IN values

Formulas

The pulse length for OUT_POS and OUT_NEG

The pulse length T_on for output OUT_pos and OUT_neg is determined by the following formulas:

Output	Equation	Condition
OUT_POS	$T\_on = t\_period \times \frac{IN}{in\_max}$	$0 \le IN \le in\_max$
OUT_NEG	$T\_on = t\_period \times \frac{\|IN\|}{in\_max}$	$0 \le -IN \le in\_max$

Parametering rules

For correct operation the following rules should be observed:

$t\_min \le t\_period$

Detailed description

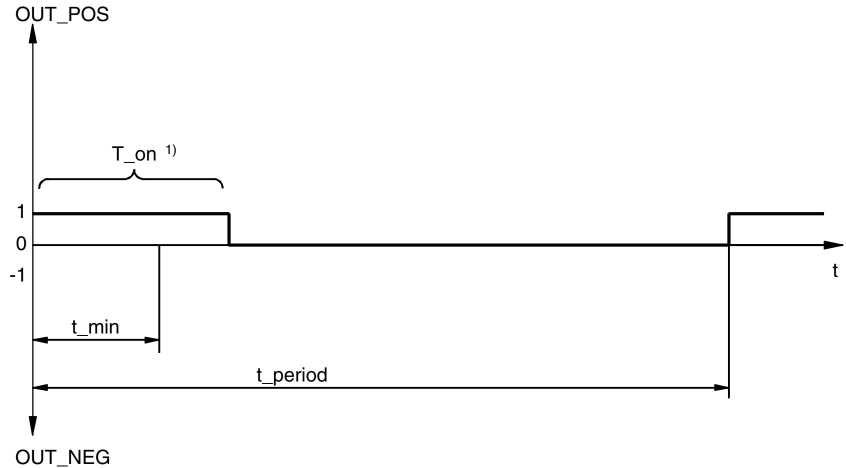
Block mode of operation

The period duration determines the time during which the actuating pulses (1-signals at the output OUT_POS or OUT_NEG) are output at regular intervals, i.e. within a constant time-slot pattern.

The parameter t_min specifies the minimum pulse length, i.e. the shortest time span for which the output Y_POS and/or Y_NEG should carry "1" signal. If the length of impulse calculated according to the equation in the section "*Formulas, p. 412*" is shorter than t_min, then there will be no impulse throughout the whole period.

Time ratios display

An overview of the ratios between times is shown in the following diagram:

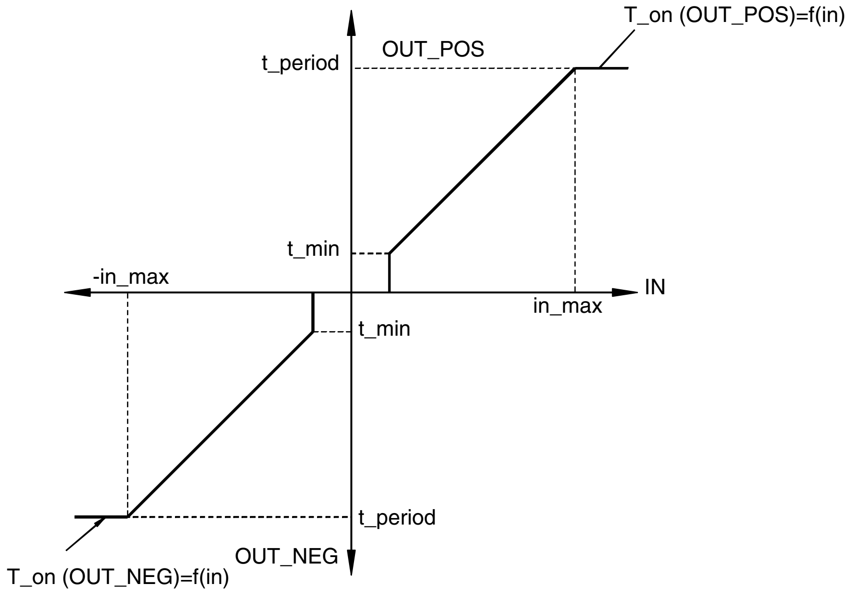


1 Variable turn-on time

The parameter in_max marks those positive values of input variable IN, for which output OUT_POS would continuously carry "1".

Time-span dependency

The dependency of the time duration in which the output OUT_POS (OUT_NEG) carries a 1-Signal, on the input variable IN is illustrated in the following diagram:



Operating mode

In reset mode RST = 1, outputs OUT_POS and OUT_NEG are set to "0" signal. The internal time meters are normalized as well so that the function block begins its transition to RST=0 with the output of a new 1-signal on the associated output.

Boundary conditions

If the PWM1 block is operated together with a PID controller, then the period t_period should be so selected, that it corresponds to the PID controller's scan time. It is then guaranteed that every new actuating signal from the PID controller within the period time can be fully processed.

The PWM1 scan time should be in proportion with the period vs. pulse time. Though this, the smallest possible actuating pulse is be determined.

The following ratio is recommended:

t_period/scan time (PWM1) $\geq$ 10

Example of the PWM1 block

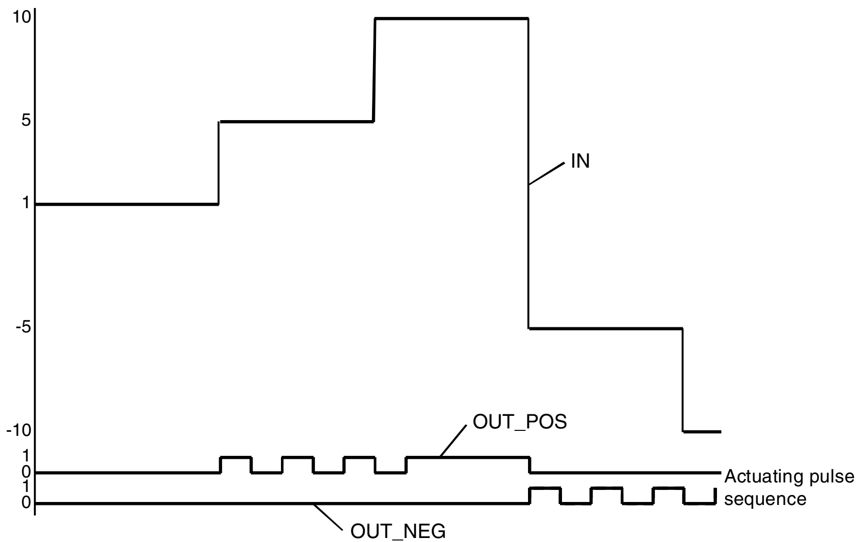
Step response

In the examples, the signal sequences on the outputs OUT_POS and OUT_NEG are shown for various IN input signal values.

The following parameter specifications apply to the step response display:

Parameter	Settings
t_period	4 s
t_min	0,5 s
in_max	10

Step response timing diagram



IN analog signal

It is noticeable that pulses are no longer output for very small IN input signals. This is directly attributable to the effect of time t_min. A continuous pulse is output for large IN (IN=in_max) signals.

QDTIME: Deadtime device

45

Overview

At a glance

This chapter describes the QDTIME block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

With this function block the input signal is delayed by deadtime.

The function block delays the signal IN by the deadtime T_DELAY, before it is transmitted to OUT again.

The function block has a delay-puffer for 128 elements (IN-VALUES), i.e. 128 IN-Values can be saved during the T_DELAY time. The puffer is used in such a way that it corresponds with the operating mode.

Whether the system is started cold or warm, the value of OUT remains unchanged. The internal values are set to the value of IN.

After the system has been started cold or warm or a change has been made to the deadtime T_DELAY, the READY will be "0". This means: that the Puffer is empty and not ready.

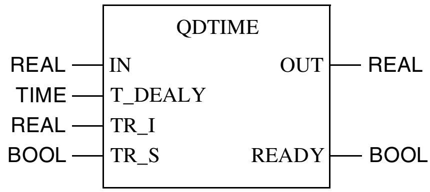
The function block has both a tracking and automatic mode.

EN and ENO can be projected as additional parameters.

Representation

Symbol

Block representation



Parameter Description

Block parameter description

Parameter	Data type	Meaning
IN	REAL	Input value
T_DELAY	TIME	Deadtime
TR_I	REAL	Initialization input
TR_S	BOOL	Initialization type "1" = Operating mode Tracking "0" = Automatic operating mode
OUT	REAL	Output
READY	BOOL	"1" = internal buffer is full "0" = internal buffer is not full (e.g. after warm/cold start or alteration to dead-time)

Detailed description

Selecting the operating modes

There are two operating modes, which can be selected via the input TR_S:

Operating mode	TR_S
Automatic	0
Tracking	1

Automatic mode

In the automatic operating mode, the function block works according to the following rules:

If…	Then…
Cycle time > T_DELAY/128	If the current IN-value is transferred to the buffer, the oldest IN-value will be displayed on the output OUT. In this case the solution is smaller than 128 and there is a systematic error, i.e. some IN values are saved twice (see also example).
Cycle time < T_DELAY/128	not all IN values can be contained in the buffer. In this case the IN value is not saved in some cycles and OUT remains unchanged in this cycle.

Example of cycle time > 128

The following values are accepted:

Cycle time = 100 ms

T_DELAY = 10 s

tin = T_DELAY / 128 = 78 ms

As tin (reading time) is shorter than the cycle time, every IN value is accepted in the buffer. On the fourth performance of the function block (after 400 ms) the IN value will be saved twice rather than once (because 3 x 78 = 312 and 4 x 78 = 390).

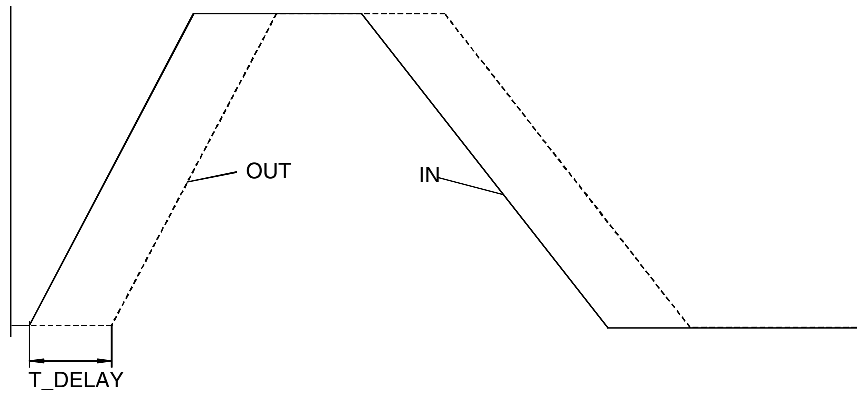
Tracking mode

In the tracking mode, the tracking value TR_I is transmitted permanently to the output OUT. The internal buffer is filled with the tracking value TR_1. The buffer is marked as full (READY =1).

Example of the behavior of the QDTIME

The diagram shows an example of the behavior of the function block. The input IN changes, in the form of a ramp, from one value to a new value and the output OUT follows the input IN, delayed by the deadtime T_DELAY.

Diagram of the QDTIME function block



QPWM: Pulse width modulation (simple)

46

Overview

At a glance

This chapter describes the QPWM block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Use of block

Actuators are driven not only by analog quantities, but also through binary actuating signals. The conversion of analog values into binary output signals is achieved for example, through pulse width modulation (QPWM) or pulse duration modulation (PDM).

The actuator adjusted average energy (actuator energy) should be in accord with the modulation block's analog input value (X).

Function description

The function block QPWM serves to convert analog values into digital output signals.

In the pulse width modulation (QPWM) a "1" signal of variable persistence proportional to the analog value X is output within a fixed cycle period. The adjusted average energy corresponds to the quotient of the duty cycle T_on and the cycle time t_period.

In order that the adjusted average energy also corresponds to the analog input variable X, the following must apply:

$T\_on \sim X$

As additional parameters, EN and ENO can be projected.

General information about the actuator drive

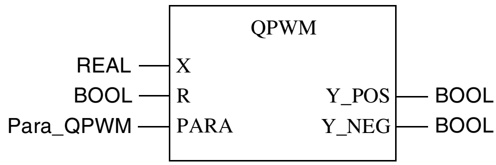
In general, the binary actuator drive is carried out by two binary signals Y_POS and Y_NEG.

On a motor the output Y_POS corresponds to the signal "clockwise rotation" and the output Y_NEG the signal "counter-clockwise rotation". For an oven the outputs Y_POS and Y_NEG could be interpreted as corresponding to "heating" and "cooling".

Representation

Symbol

Block representation



QPWM parameter description

Block parameter description

Parameter	Data type	Meaning
X	REAL	Input variable
R	BOOL	Reset mode ("1" = Reset)
PARA	Para_QPWM	Parameter
Y_POS	BOOL	Positive X value output
Y_NEG	BOOL	Negative X value output

Parameter description Para_QPWM

Data structure description

Element	Data type	Meaning
t_period	TIME	Period
t_min	TIME	Minimum actuating pulse time
x_max	REAL	Upper threshold for positive/negative X values

Formulae

The pulse length for Y_POS and Y_NEG

The pulse length T_on for output Y_pos amd Y_neg is determined by the following equations:

Output	Formula	Condition
Y_POS	$T\_on = t\_period \times \frac{X}{x\_max}$	$0 \leq X \leq x\_max$
Y_NEG	$T\_on = t\_period \times \frac{\lvert X \rvert}{x\_max}$	$0 \leq -X \leq x\_max$

Parametering rules

For correct operation the following rules should be observed:

$t\_min \leq t\_period$

Detailed description

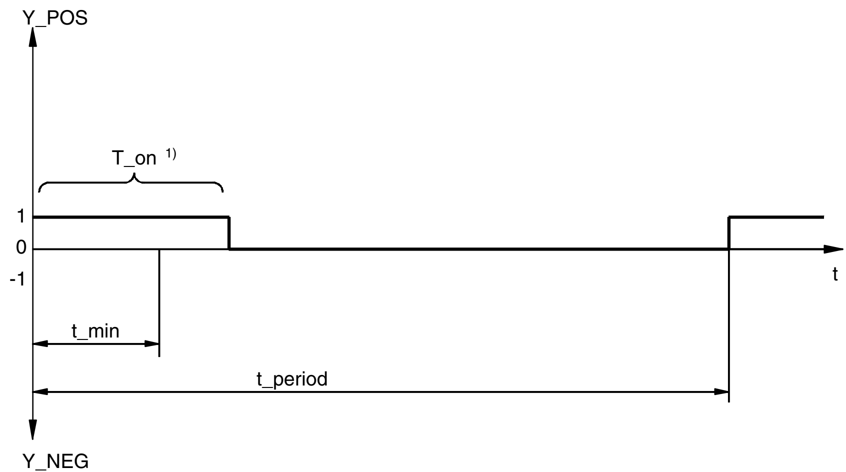
Block mode of operation

The period determines the time, in which the actuating pulses ("1" signal on output Y_POS resp. Y_NEG) are regularly output, i.e. in a constant time-slot pattern.

The parameter t_min specifies the minimum pulse length, i.e. the shortest time span for which the output Y_POS and/or Y_NEG should carry "1" signal. If the length of impulse calculated according to the equation in the section "*Formulae, p. 426*" is shorter than t_min, then there will be no impulse throughout the whole period.

Time ratios display

An overview of the ratios between times is shown in the following diagram:

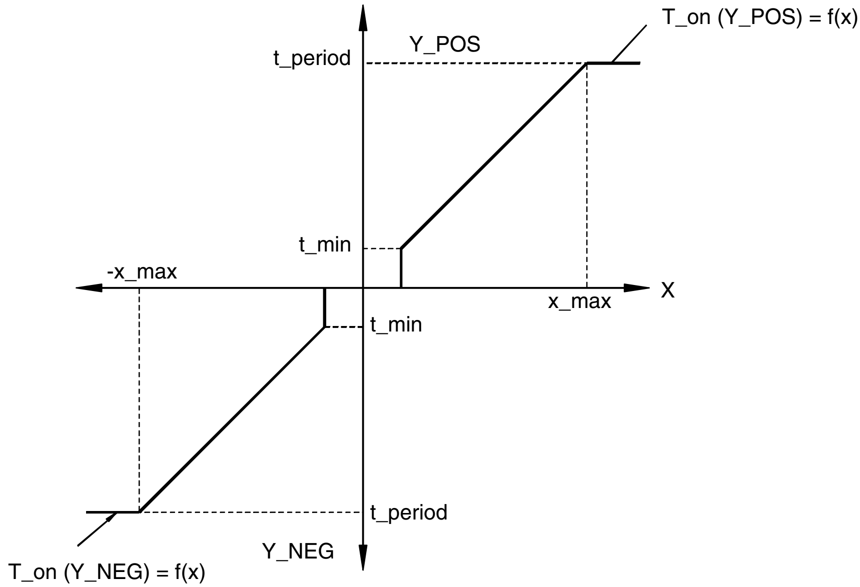


1 Variable turn-on time

The parameters x_max mark the point of input variable X, with which the output Y_POS would continuously carry "1" signal, when the input variable X is positive.

Time-span dependency

The dependency of the time duration in which the output Y_POS (Y_NEG) carries a 1 signal; the input variable X is illustrated in the following diagram :



Operating mode

In reset mode R = "1", outputs Y_POS and Y_NEG are set to "0" signal. The internal time meters are also standardized, so that the function block begins the transfer to R=0 with the output of a new 1 signal on the associated output.

Boundary conditions

If the QPWM block is operated together with a PID controller, then the period t_period should be selected, so that it corresponds to the PID controller's scan time. It is then guaranteed that every new actuating signal from the PID controller within the period time can be fully processed.

The QPWM scan time should be in proportion with period vs. pulse time, Though this, the smallest possible actuating pulse is be determined.

The following ratio is recommended:

t_period/scan time (QPWM) $\geq$ 10

Example for the QPWM block

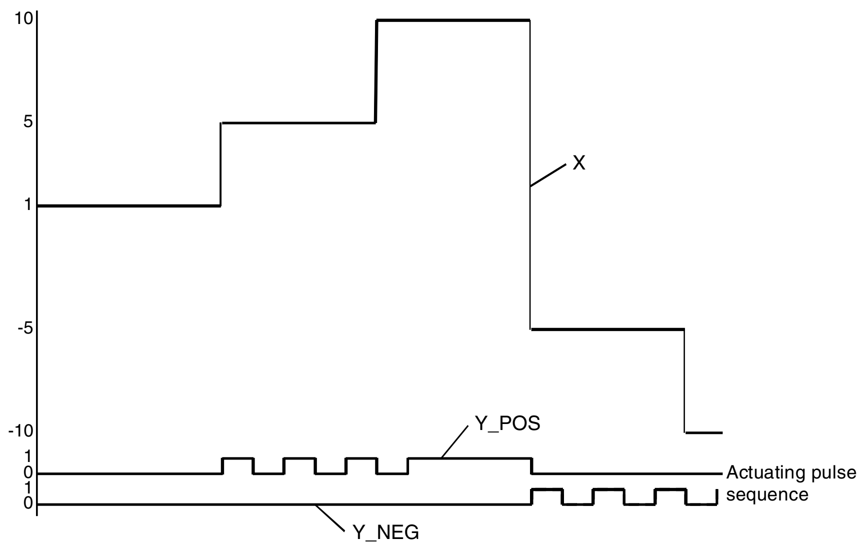
Jump response

In the example, the signal sequences on the outputs Y_POS and Y_NEG are shown for various X input signal values.

The following parameter specifications apply to the jump response display:

Parameter	Specifications
t_period	4 s
t_min	0.5 s
x_max	10

Step response timing diagram



X Analog signal

It is noticeable that pulses are no longer output for very small X input signals. This is directly attributable to the effect of time t_min. A continuous pulse is output for large X (X=x_max) signals.

RAMP: Ramp generator

47

Overview

At a glance

This chapter describes the RAMP block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block RAMP makes it possible to move in ramp-type fashion from an initial setpoint value to a particular target value. The gradients of positive and negative ramps can vary.

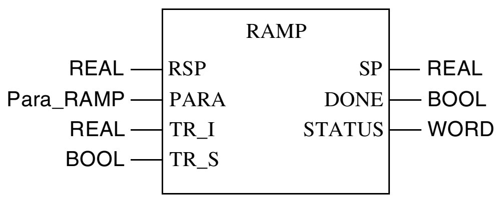
A signal (DONE output) indicates the user, whether a target value has already been reached or if the ramp had been implemented.

EN and ENO can be configured as additional parameters.

Representation

Symbol

Block representation



RAMP parameter description

Block parameter description

Parameter	Data type	Meaning
RSP	REAL	Target value of the ramp
PARA	Para_RAMP	Parameter
TR_I	REAL	Initial value of the ramp
TR_S	BOOL	Initialization command of the ramp
SP	REAL	Output
DONE	BOOL	"1": the target value has been reached "0": the ramp function has been executed
STATUS	WORD	Status word

Parameter description Para_RAMP

Data structure description

Element	Data type	Meaning
inc_rate	REAL	Positive gradient in units per second (≥0)
dec_rate	REAL	Negative gradient in units per second (≥0)

Detailed description

Parametering

If the value given on input (RSP) exceeds the current value of the SP_output, the function block increases the value of the output with the velocity inc_rate by as much as is necessary for the SP value to reach the RSP value. If the inc_rate is zero, the ramp function will not be executed and the SP is identical to the RSP.

If the given value on input falls below the current value of SP, the function block lowers the value of SP with the velocity dec_rate. If the dec_rate is zero, the ramp function will not be executed and SP is exactly the same as RSP.

If the value of RSP changes whilst the ramp is being generated, the function block immediately attempts to reach this new target value. The ramp function, which is running simultaneously, either continues or changes its direction.

Operating modes

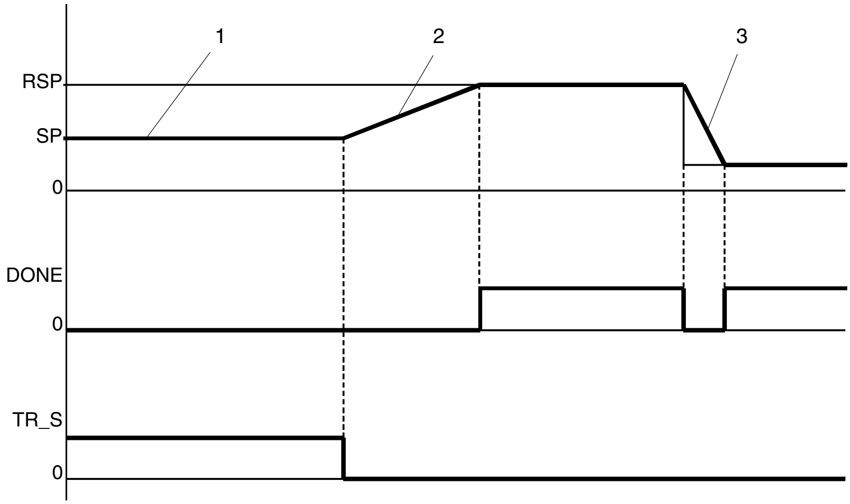
The tracking operation (TR_S = 1) allows for an initial value to be assigned to the SP output. They are as follows:

Step	Action
1	TR_I set to the desired initial value.
2	When TR_S is set to 1, the TR_I input will continue to be executed at SP. **Note:** In the tracking mode (TR_S = 1) the DONE-output remains permanently at zero.
3	If TR_S is set to zero, the function block resumes normal operation: The SP constantly approaches the RSP, where the value describes a ramp.

DONE display

The DONE output goes above 1, if a ramp function has just been completed. It will be reset to zero, when a new ramp begins or when the function block is switched to tracking mode.

Timing diagram RAMP block timing diagram



1 Initialization: SP = TR_I
2 Increasing ramp = inc_rate
3 Decreasing ramp = dec_rate

Runtime error

Status word

The following messages are displayed in the status word:

Bit	Meaning
Bit 0 = 1	Error in a calculation using floating point values
Bit 1= 1	Recording of an invalid value on one of the floating point value inputs
Bit 2= 1	Division by zero during a calculation with floating point values
Bit 3 = 1	Capacity overflow during a calculation using floating point values
Bit 4 = 1	One of the following variables is negative: inc_rate, dec_rate. For calculation, the function block uses the value 0.

Error message

An error is signaled if a non floating value is inputted or if there is a problem with a floating point calculation. In this case the outputs SP and DONE remain unmodified.

Warning

A warning appears in the following cases:
- The parameter inc_rate is negative: the function block uses the value 0 instead of the faulty value of inc_rate.
- The parameter dec_rate is negative: the function block uses the value 0 instead of the faulty value of dec_rate.

RATIO: Ratio controller

48

Overview

At a glance

This chapter describes the RATIO block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

TheFunction block RATIO executes ratio control when it is attached to a controller.

The aim of ratio control is to establish a ratio of one process variable PV (controlled variable) to another PV_TRACK (reference variable). The role of the RATIO function block is to calculate the Control setpoint corresponding to the control variable.

EN and ENO can be configured as additional parameters.

Properties

The function block has the following properties:
- The ratio can be controlled remotely (RK) or locally (K).
- Upper and lower threshold for K or RK
- Upper and lower threshold for the calculated setpoint SP
- Calculation of the real ratio: KACT = (PV - bias) / PV_TRACK

Formula

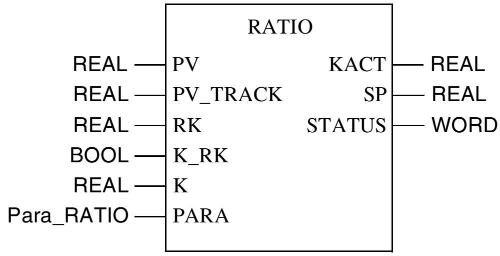
Calculation of the control setpoint

$$SP = K \times PV\_TRACK + bias$$

Representation

Symbol

Block representation



RATIO parameter description

Block parameter description

Parameter	Data type	Meaning
PV	REAL	Process value regulated by the control loop (only used to calculate KACT)
PV_TRACK	REAL	Reference variable of the control loop
RK	REAL	Remote relationship coefficient
K_RK	BOOL	Coefficient type for ratio used "1": remote ratio RK "0": local ratio K
K	REAL	Coefficient for local ratio
PARA	Para_RATIO	Parameter
KACT	REAL	Coefficient for real ratio
SP	REAL	Calculated output
STATUS	WORD	Status word

Parameter description Para_RATIO

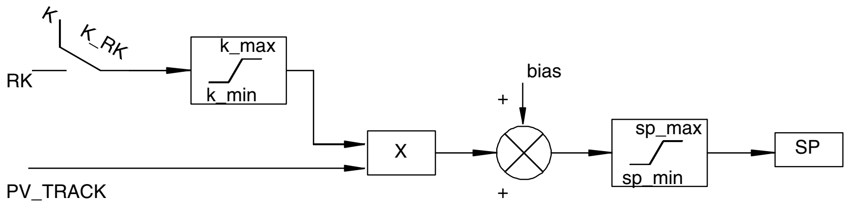
Data structure description

Element	Data type	Meaning
k_min	REAL	Lower threshold with K or RK ratio
k_max	REAL	Upper threshold with K or RK ratio
sp_min	REAL	Lower threshold of the calculated output SP
sp_max	REAL	Upper threshold of the calculated output SP
bias	REAL	Offset coefficient

Detailed description

Structure diagram

Structure diagram of the RATIO function block

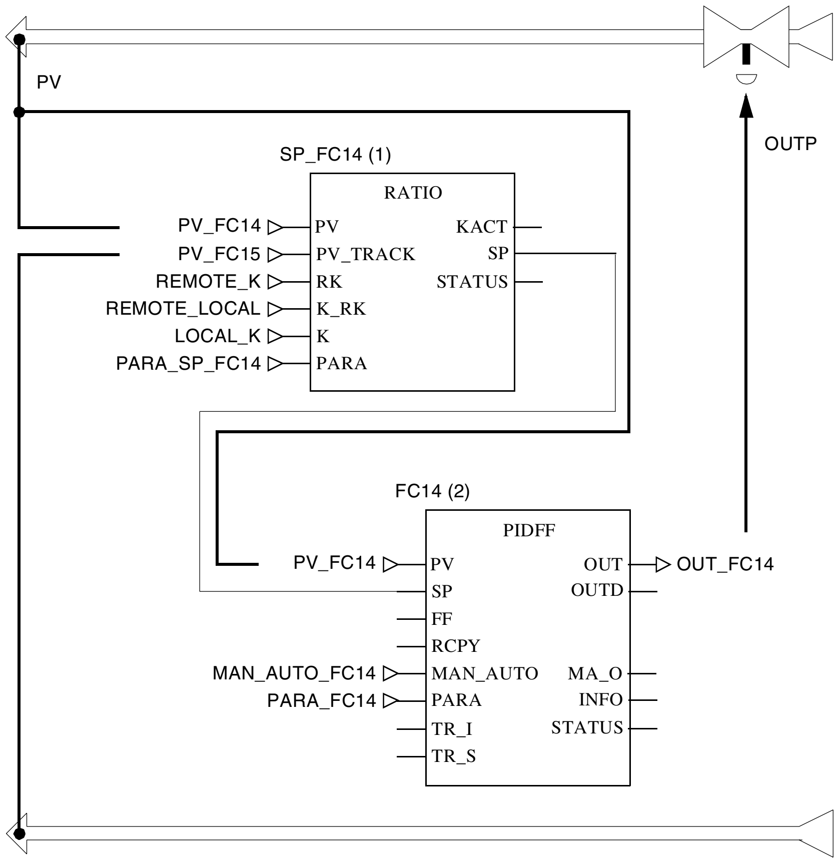


Application

The RATIO function block is upstream of a ratio controller. Its function is to calculate the remote setpoint SP of one of the controllers upgraded subsequently. The ratio controller must consist of the function blocks RATIO, SP_SEL and a controller.

Generally, this type of controller is used to regulate a flow in relation to another measured flow; it observes a specific ratio K between the two flow amounts.

Representation of the ratio controller



Runtime error

Status word

The following messages are displayed in the status word:

Bit	Meaning
Bit 0 = 1	Error in a calculation using floating point values
Bit 1= 1	Recording of an invalid value on one of the floating point value inputs
Bit 2= 1	Division by zero during a calculation with floating point values
Bit 3 = 1	Capacity overflow during a calculation using floating point values
Bit 4 = 1	The input K (or RK) is outside the range [k_min, k_max]: For calculation the function block uses the value k_min or k_max.
Bit 5 = 1	The output SP has reached the lower threshold sp_min. SP is limited to sp_min
Bit 6 = 1	The output SP has reached the upper threshold sp_max. SP is limited to sp_max

Error message

The error appears if a non floating value is inputted or if there is a problem with a floating point calculation. The outputs KACT and SP remain unmodified.

SCALING: Scaling

49

Overview

At a glance

This chapter describes the SCALING block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function block can be used to change the size of a numerical variable.

As additional parameters, EN and ENO can be projected.

Formula

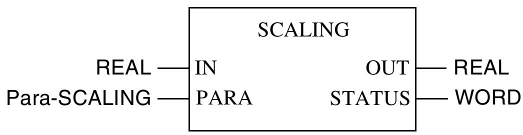
The function block carries out the following calculation:

$$OUT = (IN - in\_min) \times \frac{(out\_max - out\_min)}{(in\_max - in\_min)} + out\_min$$

Representation

Symbol

Block representation



Parameter description SCALING

Block parameter description

Parameter	Data type	Meaning
IN	REAL	Numerical variable to be scaled
PARA	Para_SCALING	Parameter
OUT	REAL	Scaled output value
STATUS	WORD	Status word

Parameter description Para_SCALING

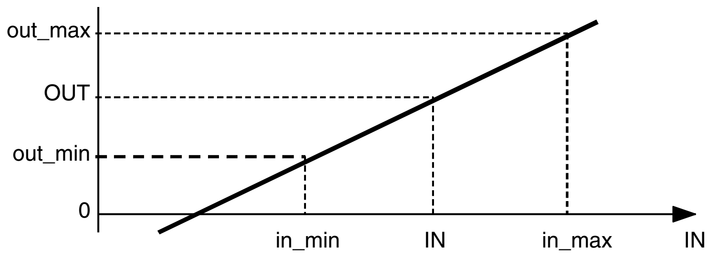
Data structure description

Element	Data type	Meaning
in_min	REAL	Lower limit of the input scale
in_max	REAL	Upper limit of the input scale
out_min	REAL	Lower limit of the output scale
out_max	REAL	Upper limit of the output scale
clip	BOOL	"1": the value of the OUT output is limited by out_min and out_max.

Parametering

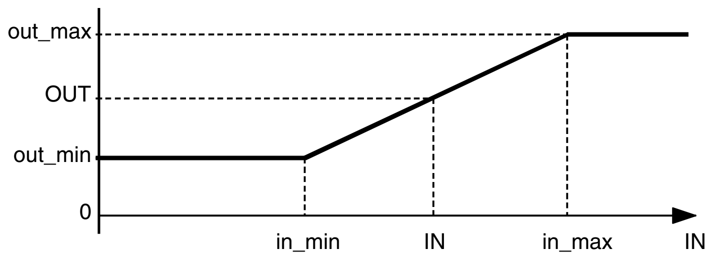
Without output limiting (clip = 0)

If the clip parameter is set to 0, then the scaling is independent of the value of the IN input.



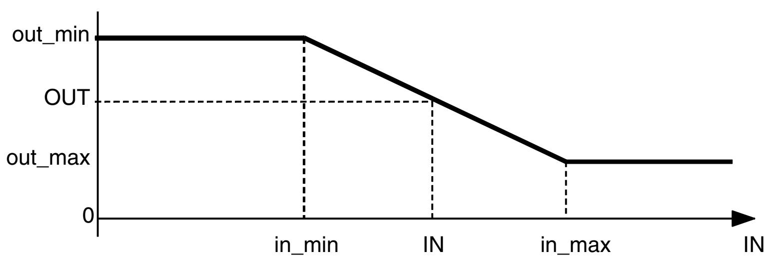
With output limiting (clip = 1)

If the clip parameter is set to 1, then the scaling takes place within the range [in_min , in_max]. Outside this range, the output will be limited by the values out_min und out_max.



Modifying the rise direction

It is possible to alter the rise direction of the numerical input variables, by setting out_max to a lower value than out_min.



Runtime error

Status word

The following messages are displayed in the status word:

Bit	Meaning
Bit 0 = 1	Error in a calculation with floating point values
Bit 1= 1	Invalid value recorded at one of the floating point value inputs
Bit 2= 1	Division by zero during a calculation with floating point values
Bit 3 = 1	Capacity overflow for a calculation with floating point values
Bit 4 = 1	The clip parameter is set to 1 and the input IN is outside this range [in_min, in_max]: for calculation the function block requires the values in_min and in_max.
Bit 7 = 1	The parameter in_min is equal to in_max

Error message

An error appears in the following cases:
- A non-floating value is on an input.
- A problem occurs during a calculation with floating point values.
- If in_min = in_max

In these cases, the OUT output remains unchanged.

SCON3: Three step controller

50

Overview

At a glance

This chapter describes the SCON3 block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block replicates a three-point step-action controller, and exhibits a PD-like behavior due to a dynamic feedback path.

As additional parameters, EN and ENO can be projected.

Properties

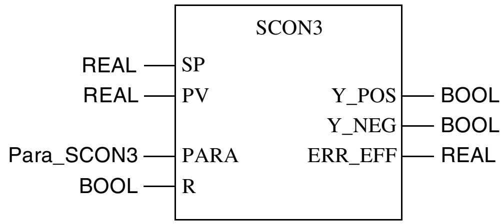
The function block SCON3 contains the following properties:

- Reset and automatic operating modes
- One internal feedback path (1st Degree Delay)

Representation

Symbol

Block representation



Parameter description SCON3

Block parameter description

Parameter	Data type	Meaning
SP	REAL	Setpoint input
PV	REAL	Process value input
PARA	Para_SCON3	Parameter
R	BOOL	Reset mode ("1" = Reset)
ERR_EFF	REAL	Effective switching value
Y_POS	BOOL	"1" = positive manipulated variable at output ERR_EFF
Y_NEG	BOOL	"1" = negative manipulated variable at output ERR_EFF

Parameter description Para_SCON3

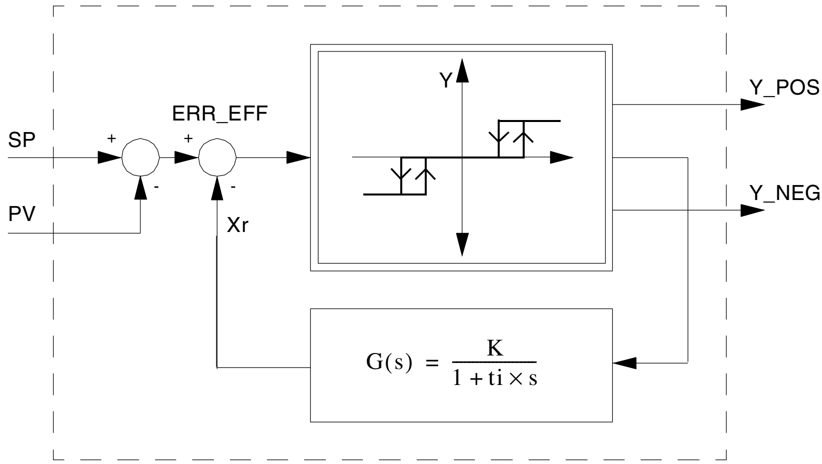
Data structure description

Element	Data type	Meaning
gain	REAL	Proportional action coefficient (gain)
ti	TIME	Reset time
t_proc	TIME	Nominal floating time of the controlled valve
hys	REAL	Three-point switch hysteresis
db	REAL	Dead zone

Detailed description

Structure of the controller

Structure of the three-point controller:



Y_POS and Y_NEG output dependency on size Y:

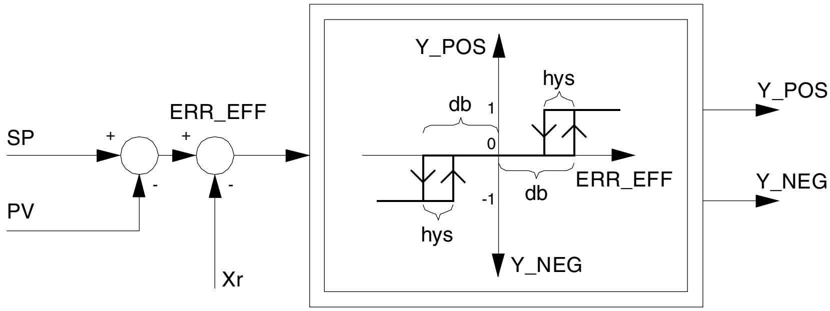
If…	Then…
Y = 1	Y_POS = 1 Y_NEG = 0
Y = 0	Y_POS = 0 Y_NEG = 0
Y = -1	Y_POS = 0 Y_NEG = 1

Size K meaning

$$K = \frac{ti}{t\_proc \times gain}$$

Principle of the three-point controller

The actual three-point controller will have a dynamic reset (PT1-element) added. By appropriately choosing the time constants (ti and t_proc) of these feedback elements, the three-point controller exhibits a dynamic behavior corresponding to that of a PID controller.



The parameter gain must be greater than zero.

Dead zone

Parameter db determines the turn-on point for the outputs Y_POS and Y_NEG. Output Y_POS/Y_NEG goes from "0" to "1" when the absolute value of positive/ negative effective error ERR_EFF becomes greater than db. If the effective switch value ERR_EFF is negative and is smaller than -DB, then the output Y_NEG will switch from "0" to "1". The parameter db is typically set to 1% of the maximum control range [max. (SP - PV)].

Note: The amount is evaluated from the dead zone DB

Hysteresis

The parameter hys specifies the hysteresis "bandwidth" extending below db, beneath which the absolute value of positive/negative effective error ERR_EFF must pass, to trigger output Y_POS/Y_NEG being reset back to "0". The connection between Y_POS and Y_NEG depending on the effective switch value ERR_EFF and the parameters DB and HYS Is illustrated in the image *Principle of the three-point controller, p. 451*. The parameter hys is typically set to 0.5% of the maximum control range [max. (SP - PV)].

Note: The amount is evaluated from the hysteresis HYS

Behavior with faulty time constants

Should the time constant ti = 0 or the proportional action coefficient gain ≤ 0 (configuration error), the block will still continue to operate. The functions feedback path is disabled however, so that the block operates as a conventional three-point switch.

If the time constant t_proc = 0 (configuration error), the block will still continue to operate. In this case T_PROC is set to a predetermined value of T_PROC = 60s (60 000 msec).

Operating modes

There are two operating modes selectable through the R parameter input:

Operating mode	R	Meaning
Automatic	0	The Function block will be handled as described previously.
Reset	1	The internal value of the feedback element is set to SP – PV. The outputs Y_POS and Y_NEG are both set to "0".

Runtime error

Error message

With hys > 2 * db, an Error Messageappears.

Warning

In the following cases there will be a Warning:
- GAIN ≤ 0 : the controller operates without feedback response.
- TI = 0 the controller operates without feedback response.
- T_PROC = 0 the controller operates with a predetermined value of T_PROC = 60s.

SERVO: Control for electric servo motors

51

Overview

At a glance

This chapter describes the SERVO block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function block enables PID control of electric servo motors with or without positional feedback. The function block can be switched to be the controller (PIDFF, PI_B) so that the digital outputs become the two logical outputs RAISE and LOWER.

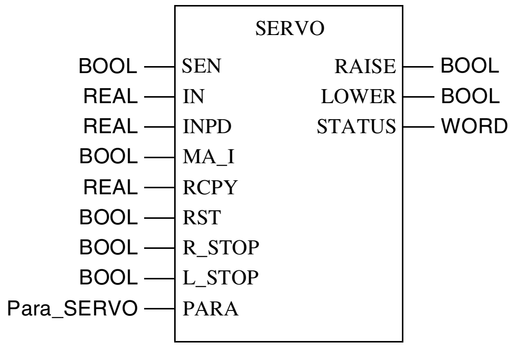
If the function block uses positional feedback, then positioning controlling of the actuator will be performed. If positional feedback is not being used, the controller and the servo function block operate a continuous static control together.

As additional parameters, EN and ENO can be projected.

Representation

Symbol

Block representation



Parameter description SERVO

Block parameter description

Parameter	**Data type**	**Meaning**
SEN	BOOL	"1" : Including a new value at the INPD or IN inputs "0" : no inclusion of the new values of INPD or IN
IN	REAL	Control output OUT (0 to 100%)
INPD	REAL	Output alteration OUTD of the controller (-100% to 100%)
MA_I	BOOL	Control operating mode (Output MA_O) "1" : Automatic mode "0" : Manual or tracking mode
RCPY	REAL	Positional feedback (0 to 100%)
RST	BOOL	"1" : Reinitialization of the function block (resetting outputs and the internal function block status)
R_STOP	BOOL	End position RAISE reached
L_STOP	BOOL	End position LOWER reached
PARA	Para_SERVO	Parameter
RAISE	BOOL	Logical output in the direction RAISE
LOWER	BOOL	Logical output in the direction LOWER
STATUS	WORD	Status word

Parameter description Para_SERVO

Data structure description

Element	Data type	Meaning
en_rcpy	BOOL	"1" : Function with positional feedback (including RCPY)
rcpy_rev	BOOL	"1" : Inversion of RCPY "0" : no inversion of RCPY
t_motor	TIME	Actuator opening time
t_mini	TIME	Minimum impulse length

Parametering

Parametering overview

The following function block modes are explained in sequence:

- *With positional feedback (en_rcpy = 1), p. 456*
- *Without positional feedback (en_rcpy = 0), p. 456*
- *Actuator opening time (t_motor), p. 457*
- *Minimum impulse length (t_mini), p. 457*
- *Sweep / parameter SEN, p. 457*
- *Recording the end position, p. 457*

With positional feedback (en_rcpy = 1)

If the positional feedback RCPY (en_rcpy = 1) is used, the input IN must be attached to the absolute value output OUT of a controller (control range 0 to 100%). For each new value for output OUT generated by the controller the SERVO function block generates a discrete output RAISE or LOWER whose length is proportional to the variance IN - RCPY. To guarantee that the function block operates correctly, the input MA_I must be attached to the controller's MA_O output.

The RCPY input value can correspond to an opening percentage (with rcpy_rev = 0) or a closing percentage (rcpy_rev set to 1).

Without positional feedback (en_rcpy = 0)

If no positional feedback is assigned (en_rcpy = 0) the INPD input should be attached to a controller's output alteration OUTD (control range -100 to 100%). For each new OUTD value generated by the controller, the function block SERVO generates a discrete output RAISE or LOWER whose length is proportional to the output length of the controller INPD. In this case it is essential that the input MA_I is attached to the same controller's MA_O output because the algorithm varies slightly for each operating mode (see section "*SERVO function block algorithms, p. 458*").

Actuator opening time (t_motor)

The parameter t_motor enables the function block to be set to the various servomotors.

The RAISE or LOWER pulse duration to be switched must be proportional to the actuator opening time with full control range.

Minimum impulse length (t_mini)

Use the t_mini parameter to avoid generation of pulses which are too short and can damage the actuator. If the RAISE or LOWER pulse length is calculated to be below t_mini the function block does not generate a pulse. Every pulse which has already commenced lasts at least t_mini.

Sweep / parameter SEN

In automatic mode the resolution of the control performed using the SERVO function block is expressed by the ratio (servoloop sampling period / SERVO function block execution period).

This means the controller must be sampled before the SERVO function block (using a SAMPLETM function block). The SERVO function block must, however, be executed every cycle. In the opposite case (if the control block is executed at the same time as the SERVO block) an inexact two point control, which the actuator makes great use of, is performed.

The SEN input of the SERVO function block indicates whether or not the PID control block was executed while the cycle was running.

The SEN input allows determination of whether or not the controller generated a new output so that the same output is not considered several times.

SEN =	Meaning
1	Including a new value
0	no inclusion of a new value

If the controller samples using the function block SAMPLETM, as is the usual case, it suffices to attach the SERVO block's SEN input to the SAMPLETM output (see section "*Examples of function block SERVO, p. 459*").

Recording the end position

If an end position is gathered (R_STOP = 1 or L_STOP = 1), the corresponding output (RAISE or LOWER) is forced to 0.

SERVO function block algorithms

Algorithm without positional feedback

In this case the SERVO function block assigned to the controller allows astatic control. The algorithm uses the output alteration OUTD rather than the controller's absolute value output OUT The output RAISE (or LOWER, depending on the modification sign) is set to 1 for a certain time. This time is proportional to the valve opening time (t_motor) and the modification value OUTD.

The formula enters an initial theoretical value for the length of the pulse (T_IMP) to be sent to the output:

$$T\_IMP = OUTD(\%).t\_motor$$

The following still applies for T_IMP (the length of the pulse sent to the output):

If…	Then…
T_IMP < t_mini	the block does not generate a pulse, but stores the value for the next calculation. This allows correct processing of control applications in which the controller's output modifications are weak but continuous. To ensure that pulses which are too short are not generated, the pulses to be sent to the output are limited to a minimum length t_mini.
the PID controller is in manual mode,	T_IMP is calculated continuously at every cycle. The calculation takes into consideration the time periods with a limit of t_motor which have previously been calculated, but not yet assigned. In this way any PID controller output modification can be considered even if the pulse lasts several cycles.
the PID controller is in automatic mode,	the function block SERVO always recalculates the parameter T_IMP if the controller updates its output, i.e. whenever SEN is set to 1. In this operating mode the previously calculated time periods are no longer considered.

Algorithm with positional feedback

The algorithm is very similar to the previous case.

In place of the PID controller output modification the SERVO function block uses the variance between the PID controller absolute value output and the positional feedback (IN - RCPY).

Positioning controlling, in which the PID controller output corresponds to the nominal value and the positional feedback RCPY to the process value, is performed by the function block.

In contrast to the algorithm without positional feedback, in manual mode the function block stores the time periods, which were calculated previously, but are not yet locked onto the RAISE and LOWER outputs.

Operating mode

Operating mode adjustment

The input MA_I allows the SERVO function block to adjust to the controller's operating mode. To do this it must be attached to the output MA_O of the controller or the corresponding MS function block.

Automatic mode

The function block SERVO only rereads the control output if this has been updated (i.e. whenever SEN is set to 1).

Manual mode

The user can modify the control output here at any time. In order that a new value can be included as soon as possible, the function block reads the control output at every cycle.

In this operating mode the user can manually modify variables connected to the OUT output of a controller or a MS block. If no positional feedback is used this variable can adopt the end position (100% or 0%) even if the actuator has not reached either of its end positions. It is still possible to modify the output modification OUTD manually by setting the output OUT of the function block MS to more than 100% (or to less than 0%). The value inputted for OUT is used for the calculation of OUTD before it is limited again.

Examples of function block SERVO

Example overview

In this section the use of the function block SERVO is shown in the following examples:
- *Automatic mode with positional feedback, p. 459*
- *Example of operating mode automatic without positional feedback in manual mode, p. 463*

Automatic mode with positional feedback

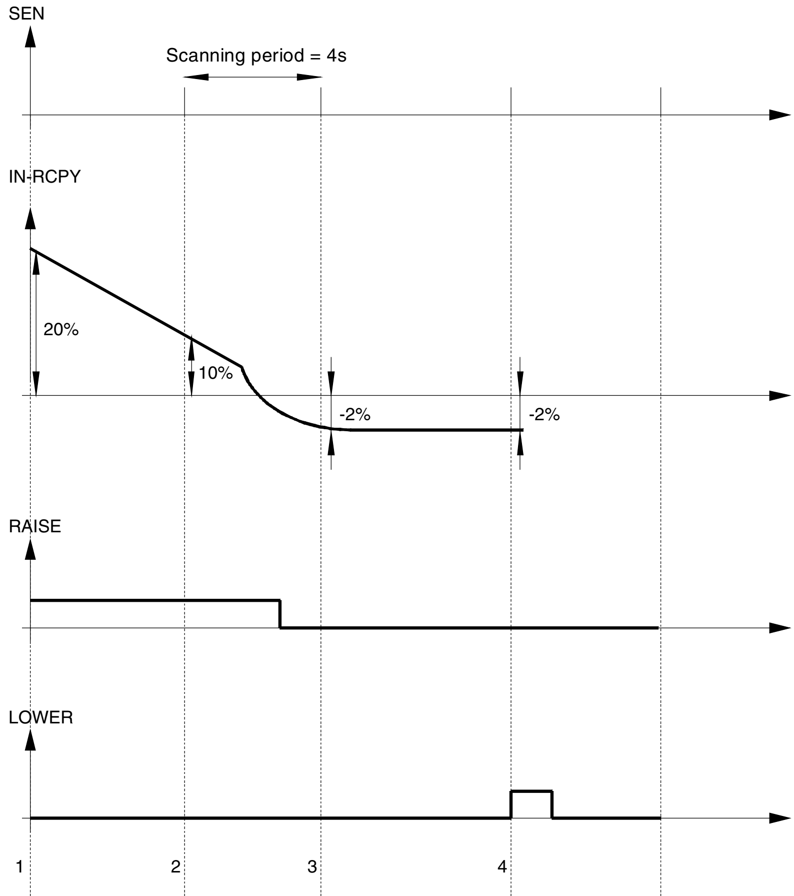
The example shows the behavior of the function block in automatic mode with positional feedback. If the SEN input is set to 1 (every 4 s in the example), the function block SERVO always takes a new variance value IN-RCPY into account.

The following parameter specifications hold:

Parameter	Specification
t_motor	25 s
t_mini	1s
sampling period	4 s

Timing diagram (automatic with positional feedback)

Timing diagram for automatic mode with positional feedback



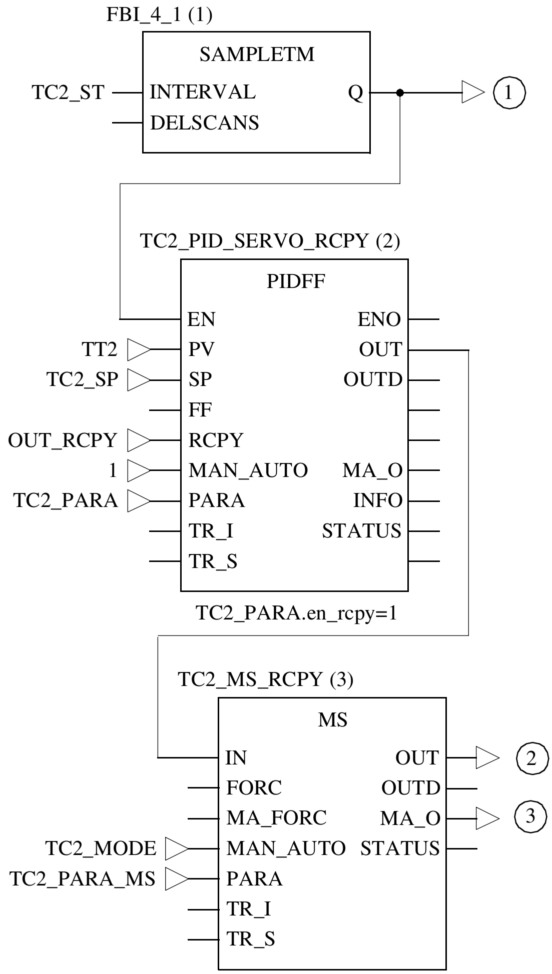
Explanation of the timings

Explanation of the marked positions:

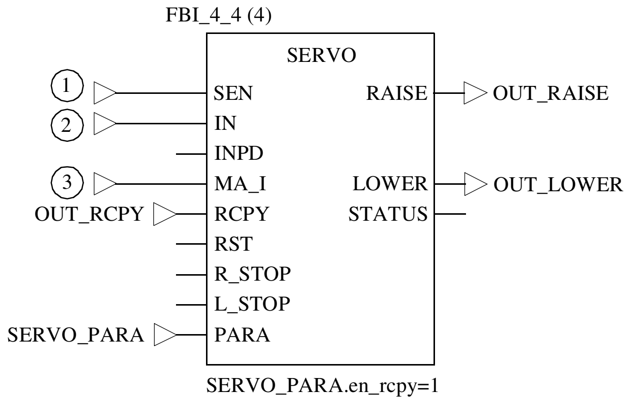
Position No.	Explanation
1	The variance IN-RCPY is 20%: a pulse of length 5 s (=20% of 25 s) was generated at the RAISE output.
2	The variance is still only 10%, a pulse of 2.5s (= 10% of 25 s) was generated at the RAISE output; the second left over from the previous pulse is not taken into account.
3	The variance is now –2% which corresponds to a pulse of 0.5 s at LOWER. Since t_mini corresponds to 1s, no pulse is generated (the duration time of 0.5 s is, however, stored).
4	The variance is still -2%, but the corresponding pulse (0.5 s) is added to the previously stored pulse to make 1 s. The length corresponds to t_mini, so the pulse is locked onto the LOWER output.

Programming example (automatic with positional feedback)

Representation of the function plan, part 1



Representation the function plan, part 2



OUT_RCPY Process value of the valve positional feedback-{}-

Example of operating mode automatic without positional feedback in manual mode

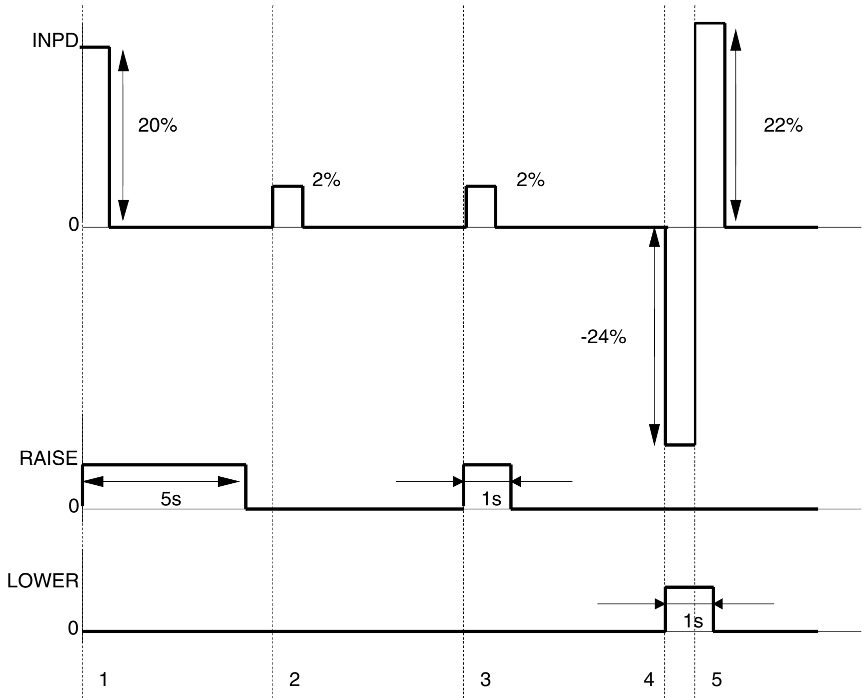
The example shows the behavior of the function block in automatic operating mode without positional feedback in manual mode. In this case the INPD value for each execution of the function block SERVO is taken into account, irrespective of the value of the SEN input.

The following parameter specifications hold:

Parameter	Specification
t_motor	25 s
t_mini	1s

Timing diagram (automatic without positional feedback)

Automatic mode without positional feedback in manual mode



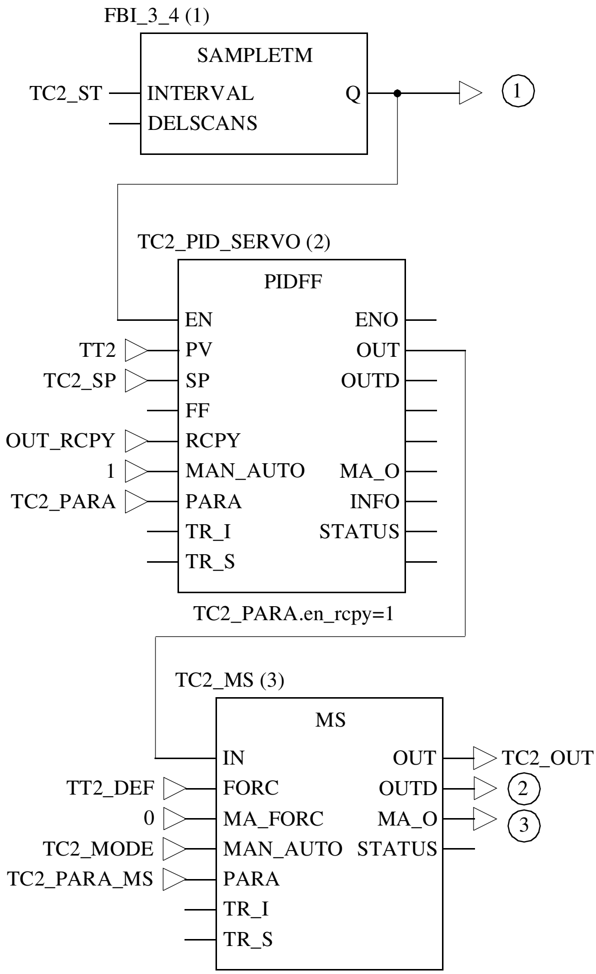
Explanation of the timings

Explanation of the marked positions:

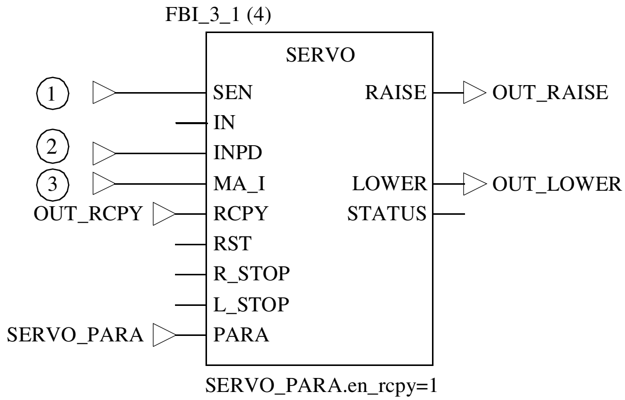
Position No.	Explanation
1	The modification of the PID control output is +20%, in this case the pulse affects the RAISE output and lasts 5 s (= 20% of 25 s).
2	The modification of the PID controller is +2% which corresponds to a pulse duration of 0.5 s. The pulse is less than t_mini (=1 s.) so it does not influence the outputs.
3	At the second modification of +2 % the function adds this modification to the previous one (which corresponds to a variance which was below the minimum value), which corresponds to a positive total modification of +4 %, i.e. a pulse of 1 s at the RAISE output.
4	For a modification of -24 % the pulse at the LOWER output is 6 s
5	Before the end of the following second a further modification of + 22 % leads to a total system modification of 2 %< modification of t_mini (4 %). The function ends with the minimum pulse of 1 s.

Programming example (automatic without positional feedback)

Representation of the function plan, part 1



Representation of the function plan, part 2



TT2_DEFF Error output of the process value TT2: If TT2 is faulty, the servoloop is forced into manual mode.

Runtime error

Status word

The following messages are displayed in the status word:

Bit	Meaning
Bit 0 = 1	Error in a floating point value calculation
Bit 1= 1	Recording of an invalid value on one of the floating point value inputs
Bit 2= 1	Division by zero during a floating point value calculation
Bit 3 = 1	Capacity overflow during floating point value calculation
Bit 4 = 1	IN or RCPY do not lie in the range [0, 100] or INPD lies outside the range [-100, 100]. To calculate the function block uses a value that is limited by the next closest correct value, i.e. 0, 100 or –100, depending on the value.

Error message

An error appears if a non floating point value is inputted or if there is a problem with a floating point calculation. In this case the outputs RAISE and LOWER are set to zero.-

SMOOTH_RATE: Differentiator with smoothing

52

Overview

At a glance

This chapter describes the SMOOTH_RATE block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This Function block implements a differential element with an output Y respecting the delay time constant LAG.

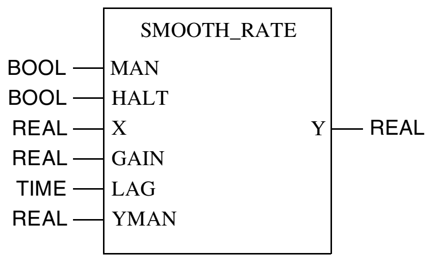
The function block has the following operating mode:
- Manual
- Halt
- Automatic

EN and ENO can be configured as additional parameters.

Representation

Symbol

Block representation



Parameter description

Block parameter description

Parameter	Data type	Meaning
MAN	BOOL	"1" = Manual mode
HALT	BOOL	"1" =Halt mode
X	REAL	Input variable
GAIN	REAL	Gain of the differentiation
LAG	TIME	Delay time constants
YMAN	REAL	Manually manipulated value
Y	REAL	Output derivative unit with smoothing

Function block SMOOTH_RATE formulas

Transfer function

The transfer function for Y is:

$$G(s) = GAIN \times \frac{1}{1 + s \times LAG}$$

Output Y

The output Y is determined as follows:

$$Y = \frac{dt}{dt + LAG} \times (Y_{(old)} + GAIN \times (X_{(new)} - X_{(old)}))$$

Explanation of formula variables

Meaning of the variables in the above formulas:

Variable	Meaning
dt	Time difference between current and previous cycle
$X_{(new)}$	Value of input X for the current cycle
$X_{(old)}$	Value of input X for the previous cycle
$Y_{(old)}$	Value of output Y for the previous cycle

Detailed description

Parametering

Parameter assignment for this function block is accomplished by selecting the GAIN of the derivative unit and the lag time constant LAG by which the output Y will be delayed.

For very short scan times and the unit jump at the input X (jump at input X from 0 to 1.0), the output Y will jump to the value GAIN (theoretical value - in reality somewhat smaller due to the fact that the scan time is not infinitely short), in order to then return with the time constant LAG to the value 0.

Operating mode

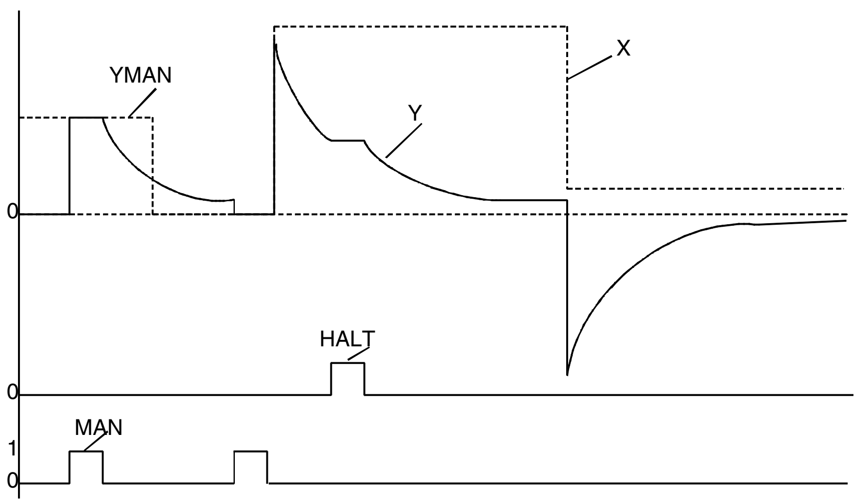
The function block SMOOTH_RATE has 3 operating mode: Automatic, manual and halt.

The operating mode are selected via the inputs MAN and HALT:

Operating mode	MAN	HALT	Meaning
Automatic	0	0	The function block operates as described in "Parametering".
Manual	1	0 or 1	The input YMAN will be transferred directly to the output Y.
Halt	0	1	The output Y will be held at the last calculated value.

Example

In the following illustration the jump response of the function block SMOTH_RATE with GAIN = 1 and LAG = 10 is shown:



SP_SEL: Setpoint switch

53

Overview

At a glance

This chapter describes the SP_SEL block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This Function blockallows the selection of setpoint value types used in the servoloop.

Setpoint value type	Explanation
Remote setpoint (SP_RSP = 1)	The setpoint comes from a block external calculation using the input RSP (Remote setpoint). The input value RSP leads to the SP output.
Local setpoint (SP_RSP = 0)	The setpoint must be modified directly by the user (Local setpoint). In this operating mode the output SP is not entered using the function block, the variable attached to the SP is modified by the user.

EN and ENO can be configured as additional parameters.

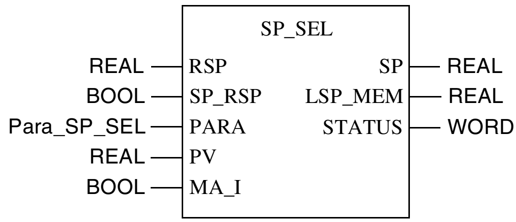
Properties

The function block SP_SEL has the following properties:
- The switchover between the setpoint values can be bumpless
- Operation with adjusting setpoint values if the controller is in manual mode
- Upper and lower limit of the setpoint value used

Representation

Symbol

Block representation



SP_SEL parameter description

Block parameter description

Parameter	Data type	Meaning
RSP	REAL	Remote setpoint
SP_RSP	BOOL	Setpoint type used by the controller: "1" : Remote setpoint "0" : Local setpoint
PARA	Para_SP_SEL	Parameter
PV	REAL	Variables to be controlled
MA_I	BOOL	Operating mode of the linked controller "1" : Automatic mode "0" : Manual mode
SP	REAL	Setpoint used by the controller
LSP_MEM	REAL	Local setpoint MEMory
STATUS	WORD	Status word

Parameter description Para_SP_SEL

Data structure description

Element	Data type	Meaning
sp_min	REAL	Lower threshold for setpoint used
sp_max	REAL	Upper threshold for setpoint used
bump	BOOL	During remote/local changeover: "1" : the SP output is forced with the value of LSP_MEM "0" : bumpless changeover
track	BOOL	"1" : the values of SP and PV are brought into line in manual mode (local setpoint only)
rate	REAL	SP increase during local/remote changeover in units per second (≥0)

Detailed description

Switching the setpoint

The setpoint can be switched in two directions

If...	Then ...
SP_RSP of 0 → 1	the local setpoint is switched to a remote setpoint
SP_RSP of 1 → 0	the remote setpoint is switched to a local setpoint

SP_RSP of 0 → 1

The changeover from local setpoint to remote setpoint is bumpless: the value of the SP output is increasingly adjusted to correspond to the remote setpoint RSP, and it describes the ramp rate.

If rate = 0, there is no ramp and the SP is identical to the RSP.

SP_RSP of 1 → 0

The changeover from remote setpoint to local setpoint depends on the bump element in two ways:

If...	Then ...
bump = 0	the changeover is bumpless: The function block stops copying the RSP input to the SP output. The local setpoint value SP then corresponds to the last remote setpoint value RSP that was present before the changeover. The user can then modify this. In this case it is not necessary to attach the LSP_MEM output.
bump = 1	the value of the LSP_MEM output is moved to the SP output during changeover (bumps can occur here). The value given for LSP_MEM corresponds to the last setpoint value SP before the function block transfers to remote mode. To restart the local mode with a different setpoint, it is sufficient to modify LSP_MEM as long as the block remains in remote mode (for further details see "*Function of the output LSP_MEM, p. 476*").

Tracked setpoint (track = 1)

At local setpoint value (SP_RSP=0), and with the linked controller in manual mode, the PV input can be continuously copied to the setpoint SP value being used. This enables a bumpless changeover from manual to automatic mode (it is also possible for the controller to control the bumpless behavior itself).

In this operating mode, the inputs PV and MA_I of the function block SP_SEL must be attached. The same values as the PV input of the controller and its MA_O output must be accepted. If track = 0, these inputs do not need to be attached.

Limits

In each operating mode (remote or local) the setpoint value SP used is limited to the range between sp_min and sp_max.

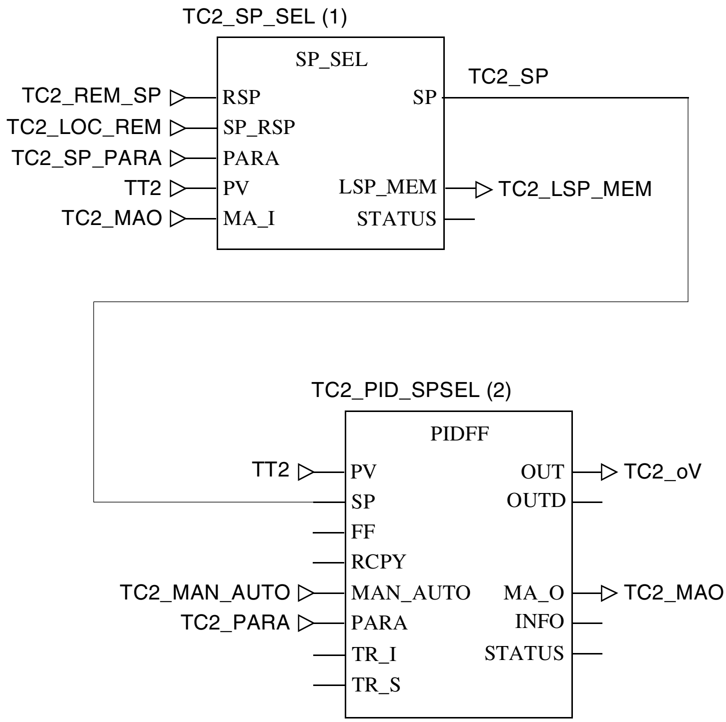
Function of the output LSP_MEM

This output enables the user to control the setpoint value SP during a remote – local changeover:

Type of setpoint	Output behavior
Local setpoint	The value of SP is continuously moved to LSP_MEM.
Changeover to remote setpoints	The value of LSP_MEM is no longer modified by the function block and therefore retains the value of the last local setpoint used.
Reverting to the local setpoint	There are three possibilities for this: **1.** bump = 0: The last remote setpoint value is used as a basis; in this case LSP_MEM does not need to be attached). **2.** bump = 1: The last local value saved is used as a basis; during changeover the block copies the value of LSP_MEM onto SP. **3.** The function block can start local mode using any value selected by the user. If the value of the variable attached to LSP_MEM before transfer to the local setpoint (with bump = 1) is modified, it is moved to SP during the changeover.

Example of programming

An example of how to program the SP_SEL function block follows.



TC2_SP is entered by the operator in "local setpoint" operating mode.

Runtime error

Status word

The following messages are displayed in the status word:

Bit	Meaning
Bit 0 = 1	Error in a floating point value calculation
Bit 1= 1	Invalid value recorded at one of the floating point inputs
Bit 2= 1	Division by zero during a floating point value calculation
Bit 3 = 1	Capacity overflow during a floating point value calculation
Bit 4 = 1	rate is negative : For calculation, the function block uses the value 0
Bit 5 = 1	The output SP has reached the lower threshold sp_min. SP is forced to sp_min
Bit 6 = 1	The output SP has reached the upper threshold sp_max. SP is forced to sp_max

Error message

An runtime error appears if a non floating point value is inputted or if there is a problem with a floating point calculation. The outputs SP and LSP_MEM remain unmodified.

Warning

A warning is giving if rate is negative; the block then uses the value 0 for calculation.

SPLRG: Controlling 2 actuators

54

Overview

At a glance

This chapter describes the SPLRG block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This Function block should be used when two actuators are in use to enable coverage of the whole area (when two operating points are far apart: one below and one above).

The controller is also suitable for three-point step-action controls, i.e. for cases where the two actuators work in opposition (one heats, the other cools).

EN and ENO can be configured as additional parameters.

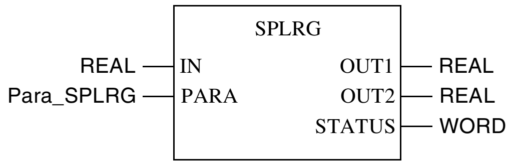
Properties

The function block SPLRG has the following properties:
- The possibility of controlling a dead zone or a transition zone where the properties of both actuators are in line
- The IN input is expressed as a percentage (0-100%) and the outputs OUT1 and OUT2 are expressed in physical units.

Representation

Symbol

Representation of the block



SPLRG parameter description

Block parameter description

Parameter	Data type	Meaning
IN	REAL	Value to be resolved (0 to 100%)
PARA	Para_SPLRG	Parameter
OUT1	REAL	Manipulated variable for actuator 1
OUT2	REAL	Manipulated variable for actuator 2
STATUS	WORD	Status word

Parameter description Para_SPLRG

Data structure description

Element	Data type	Meaning
out1_th1	REAL	Input value IN, for which the following applies: OUT1 = out1_inf
out1_th2	REAL	Input value IN, for which the following applies: OUT1 = out1_sup
out1_inf	REAL	Lower threshold of the output OUT1
out1_sup	REAL	Upper threshold of the output OUT1
out2_th1	REAL	Input value IN, for which the following applies: OUT2 = out2_inf
out2_th2	REAL	Input value IN, for which the following applies: OUT2 = out2_sup
out2_inf	REAL	Lower threshold for output OUT2
out2_sup	REAL	Upper threshold for output OUT2

Detailed description

Parametering

Parametering the function block consists of defining the properties of each actuator, i.e. in the kind of gradient modification of both control outputs in relation to the input IN.

The following points should be defined for the output OUT1:

Element	Meaning
out1_inf	Lower zone threshold
out1_sup	Upper zone threshold
out1_th1	Threshold value, i.e the input value IN, for which the following applies: Output OUT1 = out1_inf
out1_th2	Threshold value, i.e the input value IN, for which the following applies: Output OUT1 = out1_sup

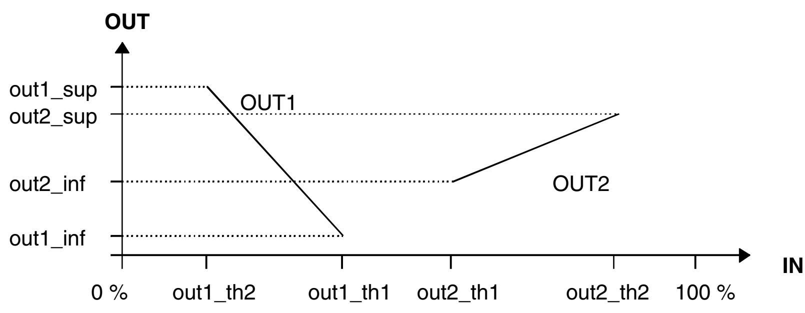
The modification of the value of OUT1 is linear for both threshold values. Apart from the two threshold values, no further output modification can occur; it is limited to out1_inf or out1_sup.

Depending on the adjustment of the two threshold values, the control properties are designated by a positive increase (for out1_th1 < out1_th2) or a negative one (with out1_th2 < out1_th1).

The following diagrams show the properties of the two actuators with Split range and Three-point step-action control.

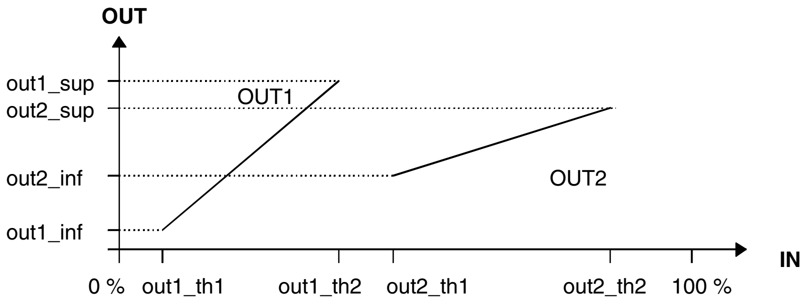
Three step step-control

The following shows the properties of the two actuators in three-point step-control



Split range control

The following shows the properties of the two actuators in split range control



Note: The outputs of this controller cannot be used to control a SERVO function block without positional feedback.

Operating modes

The SPLRG function block is not assigned to any specific operating mode. However both function block outputs may be controlled manually because an MS function block is locked on to each output. During programming the user should ensure a bumpless return to automatic mode.

Runtime error

Status word

The following messages are displayed in the status word:

Bit	Meaning
Bit 0 = 1	Error in a floating point value calculation
Bit 1= 1	Invalid value recorded at one of the floating point value inputs
Bit 2= 1	Division by zero during a floating point value calculation
Bit 3 = 1	Capacity overflow during floating point value calculation
Bit 4 = 1	IN or one of the parameters out1_th1, out1_th2, out2_th1, out2_th2 is not in the [0 - 100] range: for calculation, the function block uses the value 0 or 100.
Bit 5 = 1	The output OUT1 has reached the lower threshold out1. OUT1 is forced to out1_inf.
Bit 6 = 1	The output OUT1 has reached the upper threshold out1_sup. OUT1 is forced to out1_sup.
Bit 7 = 1	Both the threshold values of an output are identical: out1_th1 = out1_th2, out2_th1 = out2_th2.
Bit 8 = 1	The output OUT2 has reached the lower threshold out2_inf. OUT2 is forced to out2_inf.
Bit 9 = 1	The output OUT2 has reached the upper threshold out2_sup. OUT2 is forced to out2_sup.

Error message

A runtime error appears in the following cases:
- A non-floating value is on an input
- A problem occurs during a floating point value calculation.
- Both the thresholds of the same output are identical: out1_th1 = out1_th2 or out2_th1 = out2_th2.

The outputs OUT1 and OUT2 are never modified.

Warning

A warning is given if one of the parameters out1_th1, out1_th2, out2_th1, out2_th2 is not in the [0 - 100] range. In this case the function block uses the value 0 or 100 for calculating.

STEP2: Two point controller

55

Overview

At a glance

This chapter describes the STEP2 block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This Function block is suitable for simple two point controls.

Control of the actuator proceeds according to the direction of the process/setpoint value deviation in relation to the upper and lower threshold.

EN and ENO can be configured as additional parameters.

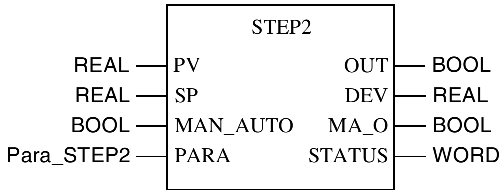
Properties

The control block has the following properties:
- Upper and lower limiting of the setpoint value between pv_inf and pv_sup.
- The control input values (process value, setpoint and associated parameters) are expressed in physical units.

Representation

Symbol

Block representation



STEP2 parameter description

Block parameter description

Parameter	**Data type**	**Meaning**
PV	REAL	Process value
SP	REAL	Setpoint
MAN_AUTO	BOOL	Controller operating mode: "1" : Automatic mode "0" : Halt mode
PARA	Para_STEP2	Parameter
OUT	BOOL	Logical output
DEV	REAL	Deviation (PV-SP)
MA_O	BOOL	Current operating mode of the function block (0: Halt, 1: Automatic)
STATUS	WORD	Status word

Parameter description Para_STEP2

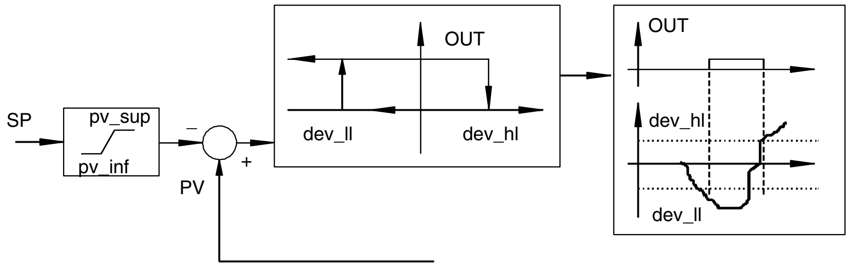
Data structure description

Element	**Data type**	**Meaning**
dev_ll	REAL	Lower deviation threshold (≤ 0)
dev_hl	REAL	Upper deviation threshold (≤ 0)
pv_inf	REAL	Lower limit of the process value range
pv_sup	REAL	Upper limit of the process value range

Detailed description

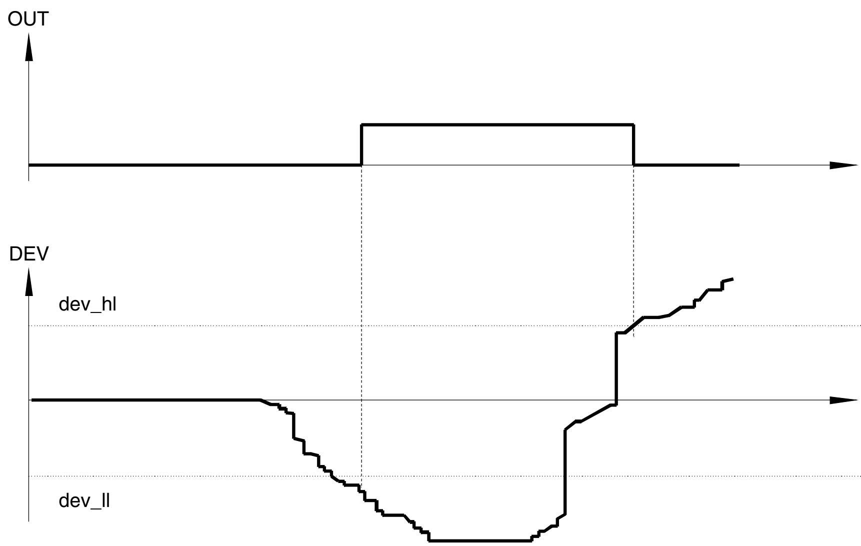
Structure diagram

The following is a structure diagram of the STEP2 block:



Behavior of the output

Behavior of the output OUT



If the deviation (DEV = PV – SP) is less than the lower threshold dev_ll, the configured output OUT is set to 1. If however the deviation increases again, the output OUT is only set to zero if it exceeds dev_hl.

Note: To ensure that the block functions without errors, the output OUT should not be inverted.

Operating modes

The STEP2 function block has 2 operating modes available according to the value of the MAN_AUTO parameter :

Operating mode	MAN_AUTO	Meaning
Automatic	1	The output OUT is self-calculated by the controller block.
Halt	0	The output OUT will be held at the last calculated value.

Runtime error

Status word

The following messages are displayed in the status word:

Bit	Meaning
Bit 0 = 1	Error in a floating point value calculation
Bit 1= 1	Invalid value recorded at one of the floating point value inputs
Bit 2= 1	Division by zero during a floating point value calculation
Bit 3 = 1	Capacity overflow during floating point value calculation
Bit 4 = 1	The following behavior is displayed: • The SP lies outside the zone [pv_inf, pv_sup]: SP is limited to pv_inf or pv_sup • dev_ll > 0 bzw. dev_hl < 0: the block uses the value 0

Error message

An runtime error appears if a non floating point value is inputted or if there is a problem with a floating point calculation. The output OUT is then set to 0; the outputs DEV and MA remain unmodified.

Warning

A warning is given if dev_ll > 0 or. dev_hl < is 0. In this case the function block uses the value 0.

STEP3: Three point controller

56

Overview

At a glance

This chapter describes the STEP3 block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This Function block is suitable for simple three-point step-action controls.

Control of the actuator proceeds according to the direction of the process/setpoint value deviation in relation to the upper and lower threshold value. The control of the threshold value describes a parameterable hysteresis.

This controller can also be used for temperature regulation. A traditional controller (such as a PI_B controller), which a function block such as the PWM1 should be switched to is preferable for complex regulation.

EN and ENO can be configured as additional parameters.

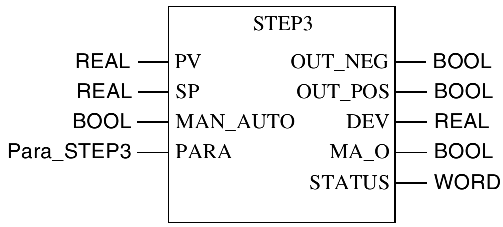
Properties

The control block has the following properties:
- Limiting the setpoint value between pv_inf and pv_sup
- The control input values (process value, setpoint, and corresponding parameters) are expressed in physical units.

Representation

Symbol

Block representation



STEP3 parameter description

Block parameter description

Parameter	Data type	Meaning
PV	REAL	Process value
SP	REAL	Setpoint
MAN_AUTO	BOOL	Controller operating mode: "1" : Automatic mode "0" : HALT mode
PARA	Para_STEP3	Parameter
OUT_NEG	BOOL	Logical output: is set to 1 for negative deviations
OUT_POS	BOOL	Logical output: is set to 1 for positive deviations
DEV	REAL	Deviation (PV-SP)
MA_O	BOOL	Current operating mode of the function block (0: HALT, 1: Automatic)
STATUS	WORD	Status word

Parameter description Para_STEP3

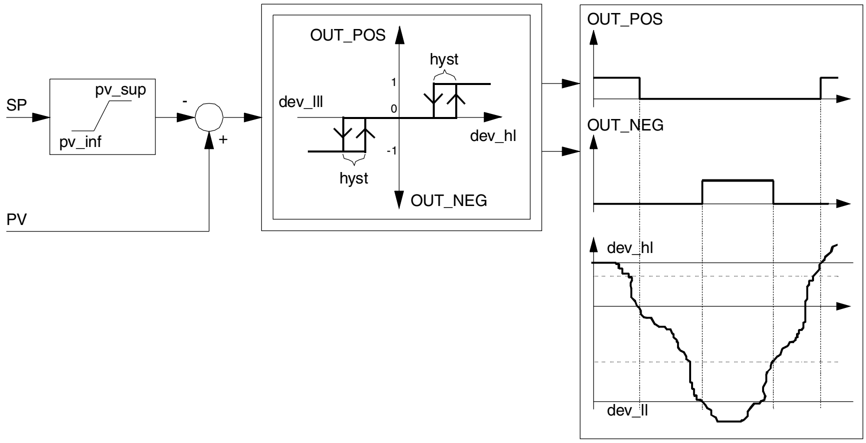
Data structure description

Element	Data type	Meaning
dev_ll	REAL	Lower deviation threshold (≤ 0)
dev_hl	REAL	Upper deviation threshold (≤ 0)
hys	REAL	Hysteresis
pv_inf	REAL	Lower limit of the process value range
pv_sup	REAL	Upper limit of the process value range

Detailed description

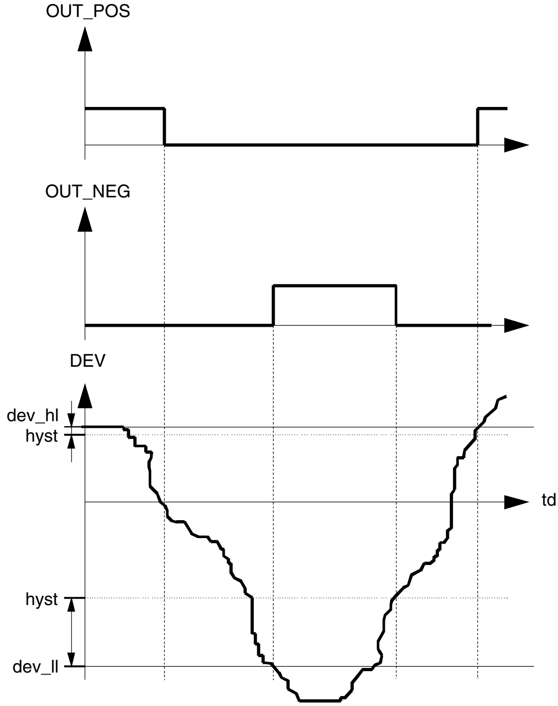
Structure diagram

The following is a structure diagram of the STEP3 block:



Behavior of the outputs

Behavior of the OUT_POS and OUT_NEG outputs:



td Duration

If the deviation (DEV = PV – SP) exceeds dev_hl, the configured output OUT_POS is set to 1. If the deviation is less, OUT_POS is then only set to zero if the deviation is less than dev_hl – hyst.

If the deviation is less than dev_ll, the configured output OUT_NEG is set to 1. If the deviation increases again, OUT_NEG is only set to zero if the deviation exceeds dev_ll + hyst.

Note: To ensure that the block functions without errors, the outputs OUT_NEG and OUT_POS should not be invented.

Operating modes

The STEP3 function block has 2 operating modes available according to the value of the MAN_AUTO parameter :

Operating mode	MAN_AUTO	Meaning
Automatic	1	The block calculates the outputs OUT_NEG and OUT_POS itself.
HALT	0	The outputs OUT_NEG and OUT_POS will be held at the last calculated value.

Runtime error

Status word

The following messages are displayed in the status word:

Bit	Meaning
Bit 0 = 1	Error in a calculation with floating point values
Bit 1= 1	Recording of an invalid value on one of the floating point value inputs
Bit 2= 1	Division by zero for a calculation with floating point values
Bit 3 = 1	Capacity overflow during calculation in floating point values
Bit 4 = 1	The following behavior is displayed: ● The SP lies outside the zone [pv_inf, pv_sup]: In this case SP is limited to pv_inf or pv_sup. ● dev_ll > 0 or. dev_hl < 0: the block uses the value 0 ● hyst is outside the [0, minimum (dev_hl, -dev_ll)] zone: the block uses a value limited to zero or to minimum (dev_hl, -dev_ll)

Error message

An run time error appears if a non floating point value is inputted or if there is a problem with a floating point calculation. In this case the outputs OUT_NEG and OUT_POS are set to 0; the DEV and MA_O outputs remain unmodified.

Warning

In the following cases a warning is given:
- dev_ll > 0 bzw. dev_hl < 0: the block uses the value 0.
- hyst is outside the [0, minimum (dev_hl, -dev_ll)] zone: the block uses a limited value.

SUM_W: Summer

57

Overview

At a glance

This chapter describes the SUM_W block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block performs the weighted summation of 3 numerical input variables according to the underlying formula.

EN and ENO can be configured as additional parameters.

Formula

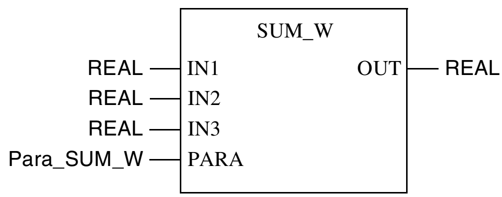
The block SUM_W operates as follows:

$$OUT = k1 \times IN1 + k2 \times IN2 + k3 \times IN3 + c1$$

Representation

Symbol

Block representation



SUM_W parameter description

Block parameter description

Parameter	Data type	Meaning
IN1 to IN3	REAL	Numerical variables to be processed
PARA	Para_SUM_W	Parameter
OUT	REAL	Result of the calculation

Parameter description Para_SUM_W

Data structure description

Element	Data type	Meaning
k1 to k3, c1	REAL	Calculation coefficients

Runtime error

Error message

An runtime error appears if a non floating point value is inputted or if there is a problem with a floating point calculation. The output OUT will not be altered.

THREEPOINT_CON1: Three point controller

58

Overview

At a glance

This chapter describes the THREEPOINT_CON1 block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block forms a three-point controller, which maintains PID-similar behavior through two dynamic feedback paths.

EN and ENO can be configured as additional parameters.

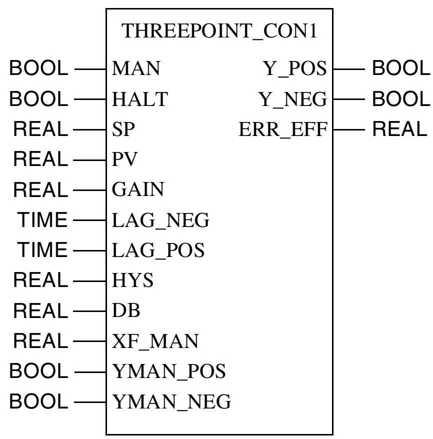
Properties

The function block THREEPOINT_CON1 contains the following properties:
- Manual, halt and automatic modes
- two internal feedback paths (1st Degree Delay)

Representation

Symbol

Block representation



Parameter description

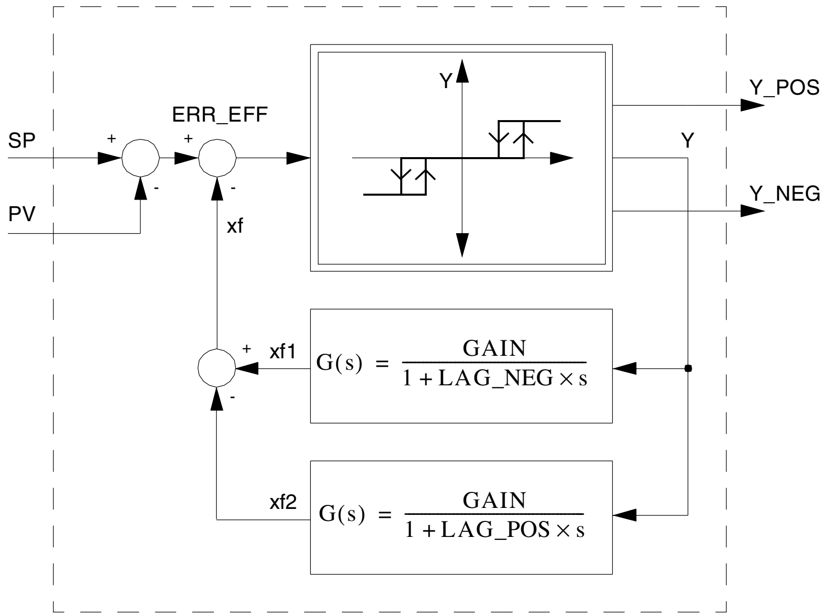
Block parameter description

Parameter	Data type	Meaning
MAN	BOOL	"1" = Manual mode
HALT	BOOL	"1" =Halt mode
SP	REAL	Setpoint input
PV	REAL	Process value input
GAIN	REAL	Feedback gain (Feedback Parameter Set)
LAG_NEG	TIME	Rapid feedback path time constant (Feedback Parameter Set)
LAG_POS	TIME	Slow feedback path time constant (Feedback Parameter Set)
HYS	REAL	Three-point switch hysteresis
DB	REAL	Dead zone
XF_MAN	REAL	Feedback path reset value in % (-100 to 100)
YMAN_POS	BOOL	Manually manipulated value for Y_POS
YMAN_NEG	BOOL	Manually manipulated value for Y_NEG
Y_POS	BOOL	"1" = positive manipulated variable at output ERR_EFF
Y_NEG	BOOL	"1" = negative manipulated variable at output ERR_EFF
ERR_EFF	REAL	Effective switching value

Detailed description

Structure of the controller

Structure of the three-point controller:



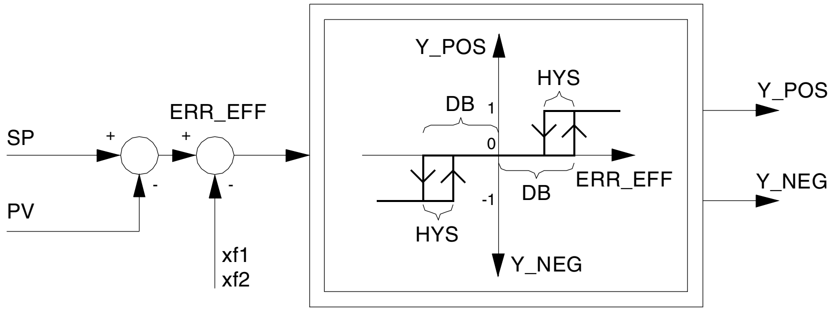
Dependency of outputs Y_POS and Y_NEG on variable Y:

If...	Then...
Y = 1	Y_POS = 1 Y_NEG = 0
Y = 0	Y_POS = 0 Y_NEG = 0
Y = -1	Y_POS = 0 Y_NEG = 1

Principle of the three-point controller

The actual three-point controller will have two dynamic feedback paths (PT1 elements) added. By appropriately selecting the time constant of these feedback elements, the three-point controller exhibits a dynamic behavior corresponding to that of a PID controller.

Principle of the three-point controller



The parameter GAIN must > be 0

Note: Entries for XF_MAN (percentages from -100% to 100%) must be in the range -100 to 100 inclusive!

Internal feedback paths

The function block has a parameter set for the internal feedback paths consisting of the feedback gain GAIN and the feedback time constants LAG_NEG and LAG_POS.

The following table provides detailed information:

Feedback	LAG_NEG	LAG_POS
3-point behavior (without feedback path)	= 0	= 0
negative feedback	> 0	= 0
negative + positive feedback	> 0	> LAG_NEG
Warning, regeneration! (neg. feedback with LAG_POS)	= 0	> 0
Warning, regeneration! (only neg. feedback with lag_neg)	> LAG_POS	> 0

Dead zone

Parameter DB determines the turn-on point for the outputs Y_POS and Y_NEG. Output Y_POS goes from "0" to "1" when the absolute value of positive effective error ERR_EFF becomes greater than DB. Output Y_NEG goes from "0" to "1" when the absolute value of negative effective error ERR_EFF becomes smaller than DB. The parameter DB is typically set to 1% of the maximum control range [max. (SP - PV)].

Note: The amount is evaluated from the dead zone DB

Hysteresis

The parameter HYS specifies the hysteresis "bandwidth" extending below DB, beneath which the absolute value of positive/negative effective error ERR_EFF must pass, to trigger output Y_POS/Y_NEG being reset back to "0". The connection between Y_POS and Y_NEG depending on effective switch-value ERR_EFF and the parameters DB and HYS will be made clear in the illustration "*Principle of the three-point controller, p. 503*". The value of the parameter HYS is typically set to 0.5% of the maximum control range [max. (SP - PV)].

Note: The amount is evaluated from the hysteresis HYS

Operating modes

There are three operating modes selectable through the inputs MAN and HALT:

Operating mode	MAN	HALT	Meaning
Automatic	0	0	The Function block will be handled as described previously.
Manual	1	0 or 1	The outputs Y_POS and Y_NEG are set to the values YMAN_POS and YMAN_NEG. A priority logic (Y_NEG is dominant over Y_POS) prevents both outputs being simultaneously set. xf1 and xf2 are calculated according to the following formula: $xf1 = XF\_MAN \times \frac{GAIN}{100}$ $xf2 = XF\_MAN \times \frac{GAIN}{100}$
Halt	0	1	The outputs Y_POS and Y_NEG are held at their last respective values. xf1 and xf2 are set to GAIN * Y.

Runtime error

Warning

In the following cases there will be a Warning:

If…	Then…
LAG_NEG = 0 and LAG_POS > 0	the controller works as if it only had a negative feedback path with the time constant LAG_POS.
LAG_POS < LAG_NEG > 0	the controller works as if it only had a negative feedback path with the time constant LAG_NEG.
XF_MAN < -100 or XF_MAN > 100	the controller operates without internal feedback paths.

THREE_STEP_CON1: Three step controller

59

Overview

At a glance

This chapter describes the THREE_STEP_CON1 block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block replicates a three-point step-action controller, and exhibits a PD-like behavior due to a dynamic feedback path.

EN and ENO can be configured as additional parameters.

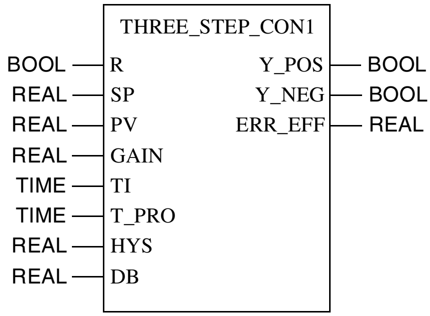
Properties

The function block THREE_STEP_CON1 has the following properties:
- Reset and automatic operating modes
- One internal feedback path (1st degree delay)

Representation

Symbol

Block representation



Parameter description

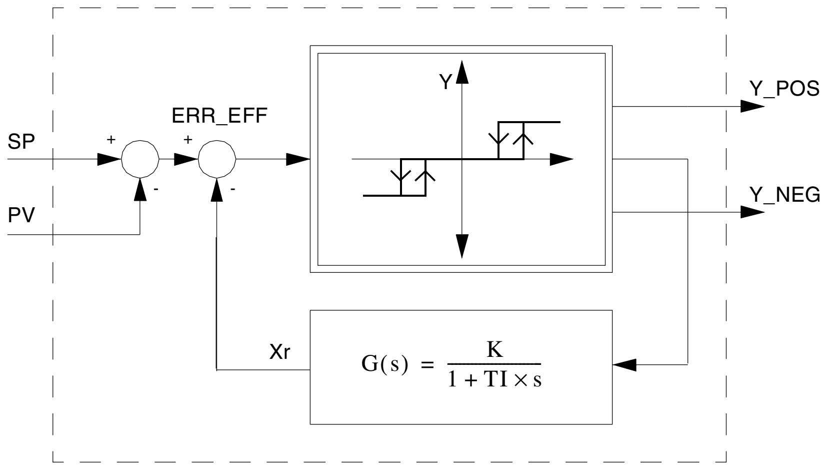
Block parameter description

Parameter	Data type	Meaning
R	BOOL	"1" = Reset mode
SP	REAL	Setpoint input
PV	REAL	Process value input
GAIN	REAL	Proportional action coefficient (gain)
TI	TIME	Reset time
T_PROC	TIME	Nominal floating time of the controlled valve
HYS	REAL	Three-point switch hysteresis
DB	REAL	Dead zone
ERR_EFF	REAL	Effective error
Y_POS	BOOL	"1" = positive manipulated variable at output ERR_EFF
Y_NEG	BOOL	"1" = negative manipulated variable at output ERR_EFF

Detailed description

Structure of the controller

Structure of the three-point controller:



Dependency of outputs Y_POS and Y_NEG on variable Y:

If…	Then…
Y = 1	Y_POS = 1 Y_NEG = 0
Y = 0	Y_POS = 0 Y_NEG = 0
Y = -1	Y_POS = 0 Y_NEG = 1

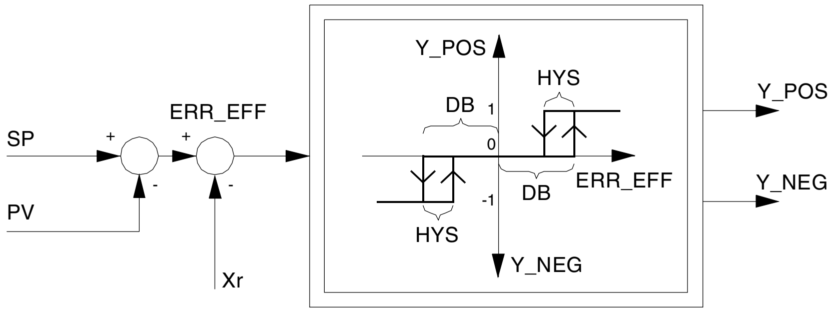
Meaning of variable K:

$$K = \frac{TI}{T\_PROC \times GAIN}$$

Principle of the three-point controller

The actual three-point controller will have a dynamic feedback path (PT1-element) added. By appropriately choosing the time constants TI and T_PROC of these feedback elements, the three-point controller exhibits a dynamic behavior corresponding to that of a PID controller.

Principle of the three-point controller



The parameter GAIN must > be 0

Dead zone

Parameter DB determines the turn-on point for the outputs Y_POS and Y_NEG. Output Y_POS goes from "0" to "1" when the absolute value of positive effective error ERR_EFF = SP - PV - XR becomes greater than DB. Output Y_NEG goes from "0" to "1" when the absolute value of negative effective error ERR_EFF becomes smaller than DB. The parameter DB is typically set to 1% of the maximum control range [max. (SP - PV)].

Note: The amount is evaluated from the dead zone DB

Hysteresis

The parameter HYS specifies the hysteresis "bandwidth" extending below DB, beneath which the absolute value of positive/negative effective error ERR_EFF must pass, to trigger output Y_POS/Y_NEG being reset back to "0". The connection between Y_POS and Y_NEG depending on the effective switch value ERR_EFF and the parameters DB and HYS will be made clear in the illustration "*Principle of the three-point controller, p. 511*". The value of the parameter HYS is typically set to 0.5 % of the maximum control range [max. (SP - PV)].

Note: The amount is evaluated from the hysteresis HYS

Behavior with faulty time constants

Should the time constant TI = 0 or the gain GAIN ≤(configuration error), the block will still continue to operate. The function of the feedback path is disabled however, so that the block operates as a conventional three-point switch.

If the time constant T_PROC = 0 (configuration error), the block will still continue to operate. In this case T_PROC is set to a predetermined value of T_PROC = 60s (60 000 msec).

Operating modes

There are two operating modes selectable through the R parameter input:

Operating mode	R	Meaning
Automatic	0	The Function block will be handled as described previously.
Reset	1	The internal value of the feedback element is set to SP – PV. The outputs Y_POS and Y_NEG are both set to "0".

Runtime error

Error message

If HYS > 2 * DB, an Error Messageappears.

Warning

In the following cases there will be a Warning:

If…	Then…
GAIN ≤ 0	the controller operates without feedback response.
TI = 0	the controller operates without feedback response.
T_PROC = 0	the controller operates with a predetermined value of T_PROC = 60s.

TOTALIZER: Integrator

60

Overview

At a Glance

This chapter describes the TOTALIZER block.

What's in this Chapter?

This chapter contains the following topics:

Brief Description

Function Description

This function block integrates the value of the IN-input (typically a flow volume) over time, until a controllable limit has been reached (typically a volume).

Additional parameters EN and ENO can be defined.

> **Note:** When using the EN enable input the following must be taken into account: If the block has not been called for a long time because the EN enable input is set to FALSE, the totalizer block runtime is extended until the next call. If the watchdog timeout is exceeded this can lead to a PLC stop.
> To remedy this, the enable input should not be used or set permanently to TRUE, so that the block is processed during every cycle.

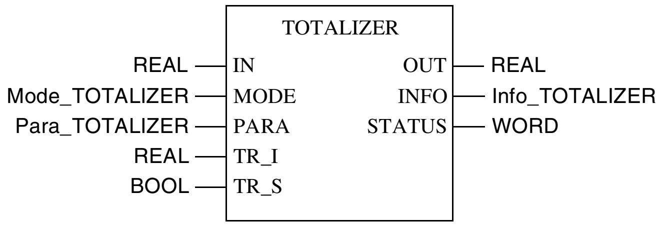
Properties

The function block has the following properties.
- The integration can be stopped for a time and reinitialized.
- Device which can also take very small values into account
- Cut-off point where the IN values are no longer taken into account.
- Use in the "reversing of the integral summation" operating mode: the output OUT is reduced from the limit value to null (inc_dec = 1)

Representation

Symbol

Block representation



Parameter description TOTALIZER

Block parameter description

Parameter	Data type	Meaning
IN	REAL	To integrated numerical sizes (only when > 0)
MODE	Mode_TOTALIZER	Operating modes
PARA	Para_TOTALIZER	Parameter
TR_I	REAL	Initiating input from outc
TR_S	BOOL	Initiating command
OUT	REAL	Result of the integration of IN (limited to thld)
INFO	Info_TOTALIZER	additional information generated by function block
STATUS	WORD	Status word

Parameter description mode_ TOTALIZER

Data structure description

Element	Data type	Meaning
hold	BOOL	"1": Stopping the integration
rst	BOOL	"1": Resetting the function block

Parameter description Para_ TOTALIZER

Data structure description

Element	Data type	Meaning
thld	REAL	Integral threshold of IN
cutoff	REAL	Division (≥0)
inc_dec	BOOL	"1" : Reverse of integration "0" : Normal mode

Parameter description Info_TOTALIZER

Data structure description

Element	Data type	Meaning
outc	REAL	Total result of the integration of IN
cter	UINT	Counter for integral calculation
done	BOOL	"1" : output OUT achieves integral threshold thld

Formulas

Calculation of the output OUT

With each execution the output OUT is calculated with the following formula:

$$OUT(new) = OUT(old) + IN \times \Delta T$$

If OUT exceeds the threshold value thld:

- the counter cter will be incremented: $cter = cter + 1$
- the threshold value thld will be deducted from the output: $OUT = OUT - thld$

Explanation of formula variables

Meaning of the variables in the formulas above:

Variable	Meaning
ΔT	time elapsed since last block execution
$OUT(old)$	Value of the output OUT at the end of the previous execution of the controller

Output of the integral results

In consideration of this principle, the function block can issue three integral results:

Result	Explanation
Partial collective index OUT	indicates the integral result of input IN from the last threshold value overflow.
cter	Frequency of achieving the threshold value
Collective register (outc)	corresponds to the integral result of the input IN since the beginning of the integral invoice This counter will be updated at every execution via the following formula: $outc = thld \times cter + OUT$

Detailed description

Setting the integral threshold thld

The integral threshold value corresponds in general to a process property, which is simple to determine (e.g. the content of a tank).

The function block can also be used for the integral calculation of smaller input values, as well as when the result of the integral invoice is very large. In this case there is the risk that the integral values will become so strongly reduced in relation to the total values that they will no longer be considered. The solution offered by TOTALIZER is in the limit of the collective index OUT on the threshold value thld, so that the integral value is never insignificant in relation to the partial collective index. The result of the integral total (outc) is also calculated: the controller saves the frequency of achieving the threshold value thld on the collective index OUT.

When the threshold value thld corresponds to the value 0, the integral value will not be calculated, the outputs remain blocked.

Further properties

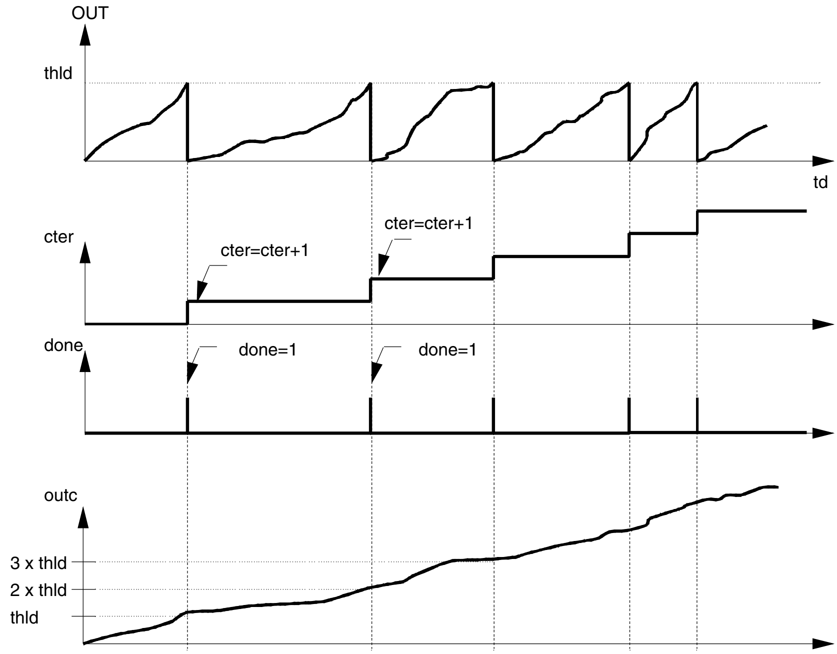
As soon as the output OUT exceeds the threshold value thld, the output done is set to 1. With the following execution of the function block they are set to zero again.

When the counter cter achieves its maximum value (65535), this value will no longer change. The outputs OUT and done continue to function when the threshold value thld is included, the output outc and the counter cter may however no longer be used.

The negative values of the input IN will never be considered, because they always lie below the division cut-off.

Timing diagram

Timing diagram of the TOTALIZER block



td Time span

Operating modes

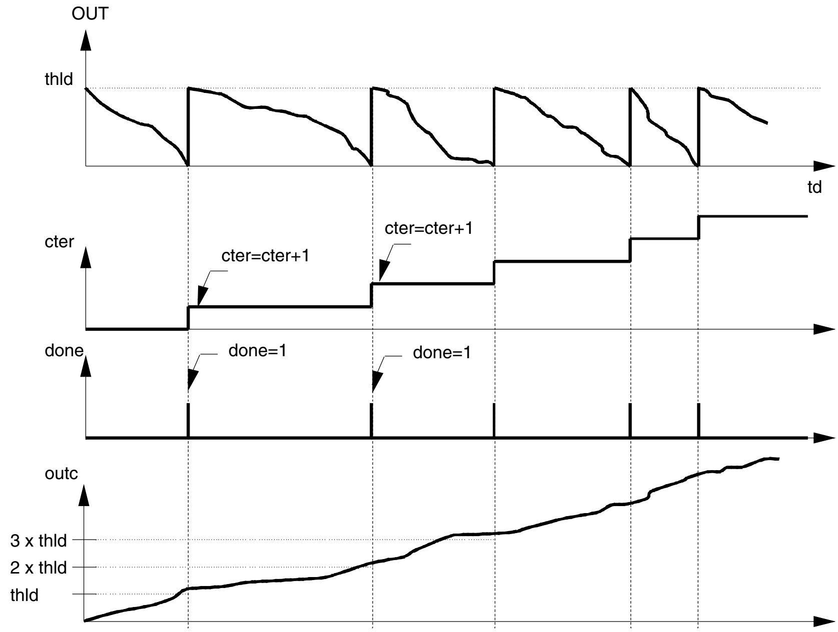
There are 3 individual operating modes for the TOTALIZER function block: Tracking, Reset and Halt:

Operating mode	Parameter	Meaning
Tracking	TR_S = 1	The parameter TR_I will be run on outc and the parameter OUT and cter will be set so that the following equation applies: outc= thld x cter + OUT. The tracking mode enables renewed synchronization of the controller outputs with the control process (e.g. as a consequence of a sensor failure).
Reset	rst = 1	The outputs OUT, outc, cter and done are set to zero. The reset via rst allows a new start from the zero reference point (for example after phase change in production).
Halt	hold = 1	Integration is paused. The outputs keep their previous values.

Note: By simultaneous activation of the inputs TR_S, rst and hold, the tracking mode has priority over the other operating modes and the reset operating mode has priority over halt.

Reverse integral summation (inc_dec = 1)

Display of the function principle



td Time span

In tracking mode (TR_S = 1) the parameter TR_I will be run on outc and the parameter OUT and cter will be set so that the following equation applies:

outc = thld x cter + (thld –OUT).

outc is calculated using the following formula: outc = thld x cter + (thld –OUT).

Function principle of the reverse of the integral summation

The following function principle applies:

Step	Action
1	At the first execution or positive on edge on rst the output OUT will be initiated by thld.
2	Thereafter with each execution the output OUT is calculated with the following formula: $OUT(new) = OUT(old) - IN \times \Delta T$
3	As soon as the output OUT becomes negative, the following happens: ● The counter cter will be incremented: cter = cter + 1 ● The threshold value thld will be added on to the output OUT: OUT = OUT + thld ● done is set to 1

Runtime error

Status word

The following messages are displayed in the status word:

Bit	Meaning
Bit 0 = 1	Error in a floating point value calculation
Bit 1= 1	Invalid value recorded at one of the floating point value inputs
Bit 2= 1	Division by zero during a floating point value calculation
Bit 3 = 1	Capacity overflow during floating point value calculation
Bit 4 = 1	The input TR_I or one of the Paramaters thld or cutoff are negative: For calculation, the function block uses the value 0
Bit 6 = 1	The count register cter has reached its maximum value (65535) : cter is locked at this value and the output outc no longer has any meaning. The OUT outputs and done can however continue to be used.

Error message

A runtime error is signaled if a non-floating point value is inputted or if there is a problem with a floating point calculation. In this case the OUT, outc, cter and done outputs remain unmodified.

Warning

In the following cases a warning is given:

If…	Then…
thld < 0	For calculation, the controller uses the value 0
cutoff < 0	For calculation, the controller uses the value 0
cter = 65535	cter is blocked at this value and the output outc no longer has any meaning. The OUT and done outputs can however continue to be used.

TWOPOINT_CON1: Two point controller

61

Overview

At a Glance

This chapter describes the TWOPOINT_CON1 block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block forms a two-point controller, which maintains PID-similar behavior through two dynamic feedback paths.

EN and ENO can be configured as additional parameters.

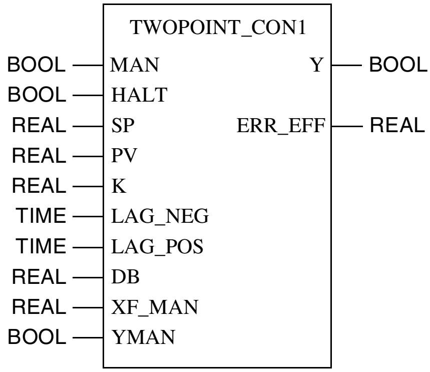
Properties

The function block TWOPOINT_CON1 has the following properties:
- Manual, halt and automatic modes
- Two internal feedback paths (1st Degree Delay)

Representation

Symbol

Block representation



Parameter description

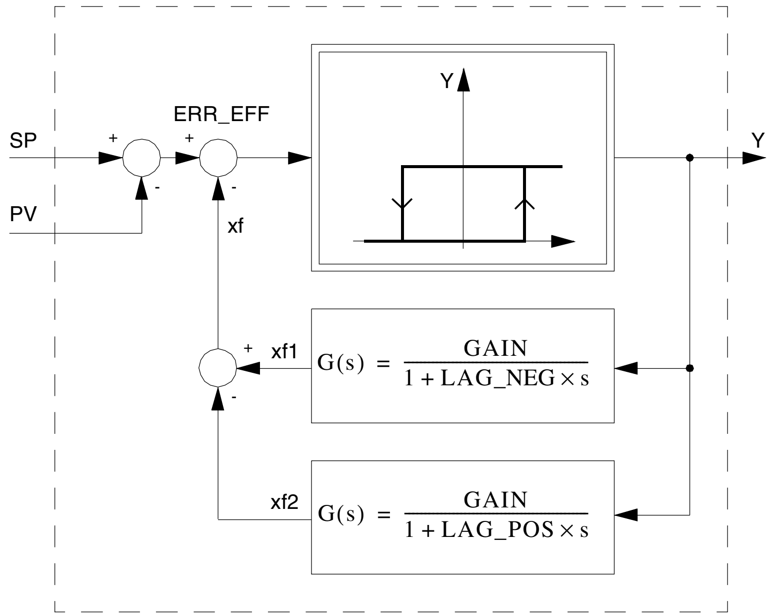
Block parameter description

Parameter	Data type	Meaning
MAN	BOOL	"1" = Manual mode
HALT	BOOL	"1" =Halt mode
SP	REAL	Setpoint input
PV	REAL	Process value input
K	REAL	Feedback gain
LAG_NEG	TIME	Rapid feedback path time constant
LAG_POS	TIME	Slow feedback path time constant
DB	REAL	Two-point switch hysteresis
XF_MAN	REAL	Feedback path reset value in % (0 to 100)
YMAN	BOOL	"1" = Manual value for ERR_EFF
Y	BOOL	"1" = Output manipulated variable
ERR_EFF	REAL	Effective error

Detailed description

Structure of the controller

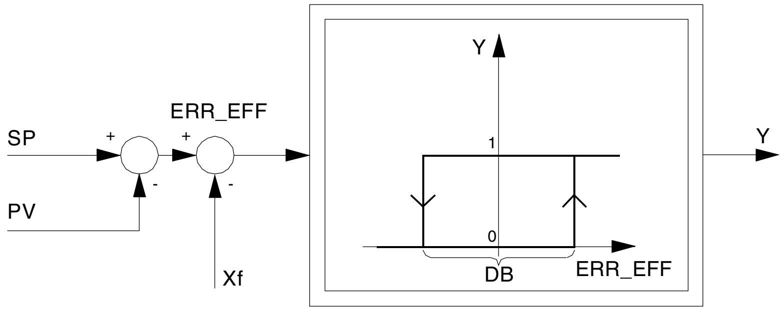
Structure of the two-point controller:



Principle of the two-point controller

The actual two-point controller will have two dynamic feedback paths (PT1 elements) added. By appropriately choosing the time constant of these feedback elements, the two-point controller exhibits a dynamic behavior corresponding to that of a PID controller.

Principle of the two-point controller:



The selected feedback gain K must be greater than zero!

Entries for XF_MAN (percentages from 0 to 100%) must be in the range 0 to 100 inclusive!

Internal feedback paths

The feedback parameter set, consisting of the feedback gain K and the feedback time constants LAG_NEG and LAG_POS, allows a universal employment of the two-point controller.

The following table provides detailed information:

Feedback	**LAG_NEG**	**LAG_POS**
2-point behavior (without feedback path)	= 0	= 0
negative feedback	> 0	= 0
negative + positive feedback	> 0	> LAG_NEG
Warning, regeneration! (neg. feedback with LAG_POS)	= 0	> 0
Warning, regeneration! (only neg. feedback with lag_neg)	> LAG_POS	> 0

Hysteresis

The parameter DB indirectly specifies the threshold values, below which the effective error ERR_EFF must pass, to trigger output Y being reset back to "0" (i.e. hys is the hysteresis "bandwidth" centered on "0", the absolute values of the relative switching points = DB/2). The dependence of the output Y depending of the effective switch value ERR_EFF and the Parameter DB, becomes clear in the illustration "*Principle of the two-point controller, p. 527*". The value of the parameter DB is typically set to 1% of the maximum control range [max. (SP - PV)].

Operating modes

There are three operating modes selectable through the inputs MAN and HALT:

Operating mode	MAN	HALT	Meaning
Automatic	0	0	The Function block will be handled as described previously.
Manual	1	0 or 1	The output Y is set to the YMAN value. xf1 and xf2 are calculated according to the following formula: $xf1 = XF\_MAN \times \frac{GAIN}{100}$ $xf2 = XF\_MAN \times \frac{GAIN}{100}$
Halt	0	1	The output Y is held at its last value. xf1 and xf2 are set to GAIN * Y.

Runtime error

Error message

If HYS > 2 * DB, an Error Messageappears.

Warning

In the following cases there will be a Warning:

If…	Then…
LAG_NEG = 0 and LAG_POS > 0	the controller works as if it only had a negative feedback path with the time constant LAG_POS.
LAG_POS < LAG_NEG > 0	the controller works as if it only had a negative feedback path with the time constant LAG_NEG.
XF_MAN < 0 or XF_MAN > 100	the controller operates without internal feedback paths response.

VEL_LIM: Velocity limiter

62

Overview

At a Glance

This chapter describes the VEL_LIM block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block realizes a velocity limiter with manipulated variable limiting.

The gradient of the input variable IN is limited to a predefinable RATE value. It also limits the output OUT to within OUT_MAX and OUT_MIN. This allows the function block to adjust signals to the technologically limited pace and limits from actuators.

EN and ENO can be configured as additional parameters.

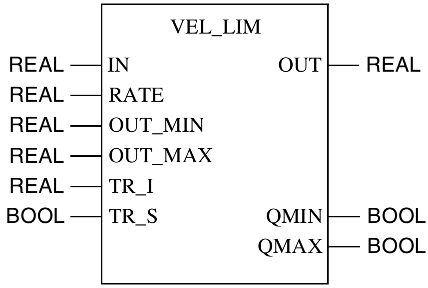
Properties

The function block has the following properties:
- Tracking and automatic operating modes
- Manipulated variable limiting in automatic mode

Representation

Symbol

Block representation



Parameter description

Block parameter description

Parameter	Data type	Meaning
IN	REAL	Input
RATE	REAL	Maximum velocity limiting
OUT_MIN	REAL	Lower limit
OUT_MAX	REAL	Upper limit
TR_I	REAL	Initiating input
TR_S	BOOL	Initiation type "1" = Operating mode Tracking "0" = Automatic operating mode
OUT	REAL	Output
QMIN	BOOL	"1" = Output OUT, has reached lower limit
QMAX	BOOL	"1" = Output OUT has reached upper limit

Detailed description

Parametering

Parameter assignment for the function block is accomplished by establishing the maximum velocity RATE as well as the OUT_MAX and OUT_MIN limits for the output OUT. The maximum velocity rate indicates by how much the output may change within one second.

Actual RATE = 0, becomes OUT = IN.

The limits OUT_MAX and OUT_MIN limit the upper output as well as the lower output. Hence OUT_MIN ≤ OUT ≤ OUT_MAX.

The outputs QMAX and QMIN signal that the output has reached a limit, and thus been capped.

- QMAX = 1 if OUT ≥ OUT_MAX
- QMIN = 1 if OUT ≤ OUT_MIN

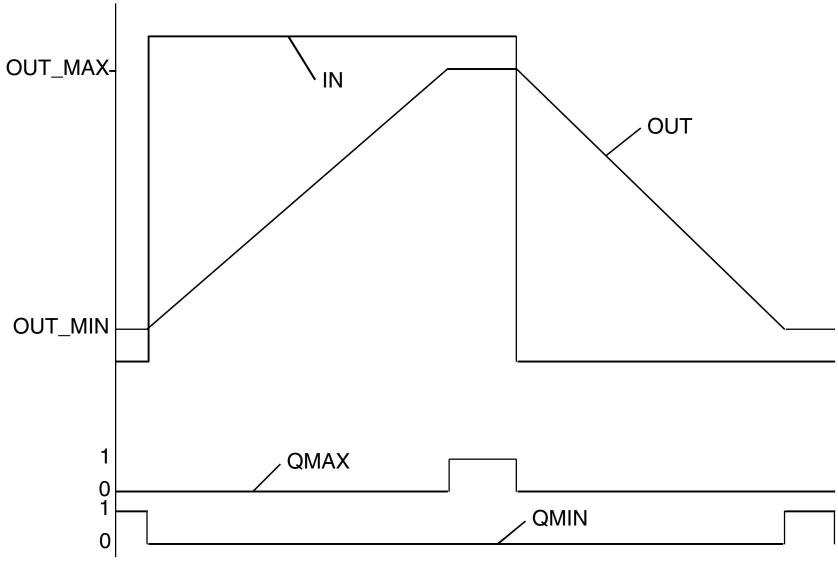
Operating modes

There are two operating modes, which can be selected via the input TR_S:

Operating mode	TR_S	Meaning
Automatic	0	The current value for OUT will be constantly calculated and displayed.
Tracking	1	The tracking value TR_I is transferred directly to the output OUT. The control output is, however, limited by OUT_MAX and OUT_MIN.

Example

Explanation of the dynamic behavior of the VEL_LIM function block.



The function block follows the transition at the input IN at its maximum velocity change rate. It can also be clearly seen that the output OUT is limited by OUT_MAX and OUT_MIN with the associated QMAX and QMIN signals.

Runtime error

Error message

With OUT_MAX < OUT_MIN an Error message appears

VLIM: Velocity limiter: 1st order

63

Overview

At a Glance

This chapter describes the VLIM block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block realizes a velocity limiter of the 1st order with limiting of the manipulated variable.

The output Y follows the input X, but at the maximum gradient rate. Furthermore, the output Y will be limited by YMAX and YMIN. This allows the function block to adjust signals to the technologically limited pace and limits from controlling elements.

EN and ENO can be projected as additional parameters.

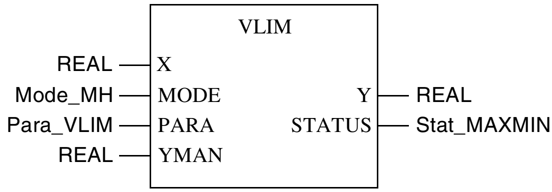
Properties

The function block contains the following properties:
- Operating mode, Hand, Halt, Automatic
- Manipulated variable limiting

Representation

Symbol

Block representation



VLIM parameter description

Block parameter description

Parameter	Data type	Meaning
X	REAL	Input
MODE	Mode_MH	Operating modes
PARA	Para_VLIM	Parameter
YMAN	REAL	Manually manipulated value
Y	REAL	Output
STATUS	Stat_MAXMIN	Y output status

Parameter description Mode_VLIM

Data structure description

Element	Data type	Meaning
Man	BOOL	"1": Manual mode
Halt	BOOL	"1": Halt mode

Parameter description Para_VLIM

Data structure description

Element	Data type	Meaning
rate	REAL	Maximum velocity (maximum x' / sec)
ymax	REAL	Upper limit
ymin	REAL	Lower limit

Parameter description Stat_MAXMIN

Data structure description

Element	Data type	Meaning
qmax	BOOL	"1" = Y has reached upper limit
qmin	BOOL	"1" = Y has reached lower limit

Detailed description

Parametering

The parametering of the function block appears through specification of the maximum upper speed RATE as well as the limits YMAX and YMIN for output Y. The maximum upper speed specifies to which value the output can change within one second.

The amount will be resolved from the parameter RATE.

Exception: RATE = 0

If RATE = 0 is projected, then output Y follows input X automatically (Y=X).

Limits

The limits YMAX and YMIN limit the upper output as well as the lower output. So that means YMIN ≤ Y ≤ YMAX.

The outputs QMAX and QMIN signal that the output has reached a limit, and thus been capped.
- QMAX = 1 if Y ≥ YMAX
- QMIN = 1 if Y ≤ YMIN

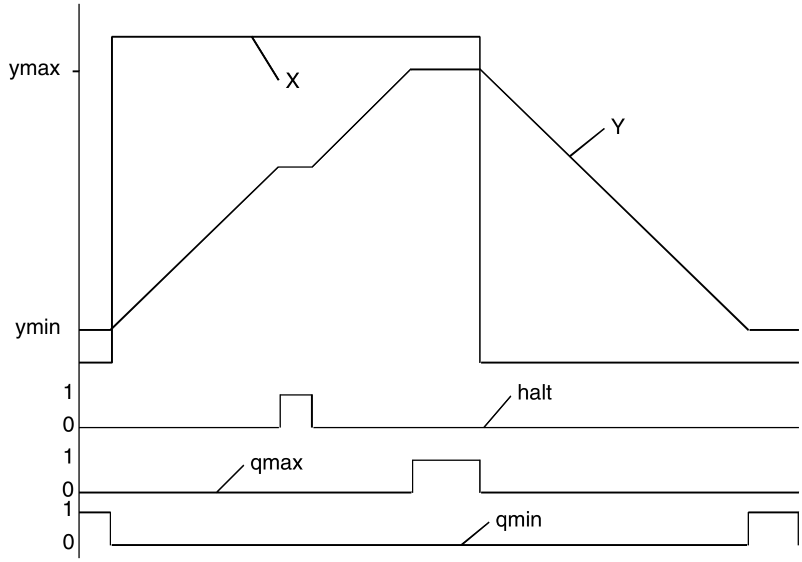
Operating mode

There are three operating mode, which are selected via the elements MAN and HALT:

Operating mode	MAN	HALT	Meaning
Automatic	0	0	The current value for Y will be constantly calculated and displayed.
Hand	1	0 or 1	The manual value YMAN will be transmitted fixed to the output Y. The control output is, however, limited through ymax and ymin.
Halt	0	1	The output Y will be held at the last value. The output will no longer be changed, but can be overwritten by the user.

Example

Explanation of the dynamic behavior of the VLIM function block.



The function block follows the jump to input X with maximum change in speed (RATE). Output Y remains at a standstill in Halt mode, in order to subsequently move on from the rank at which it has stopped. It is also clear to see the limits of output Y through YMAX and YMIN with the relevant messages QMAX and QMIN.

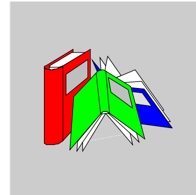
Rum-time error

Error message

There is a Error message, if

- an invalid floating point number lies at input YMAN or X,
- is ymax < ymin.

Glossary



A

active Window
The window, which is currently selected. Only one window can be active at any given time. When a window is active, the color of the title bar changes, so that it is distinguishable from the other windows. Unselected windows are inactive.

Actual Parameters
Current connected Input / Output Parameters.

Addresses
(Direct) addresses are memory ranges on the PLC. They are located in the State RAM and can be assigned Input/Output modules.
The display/entry of direct addresses is possible in the following formats:
- Standard Format (400001)
- Separator Format (4:00001)
- Compact format (4:1)
- IEC Format (QW1)

ANL_IN
ANL_IN stands for the "Analog Input" data type and is used when processing analog values. The 3x-References for the configured analog input module, which were specified in the I/O component list, are automatically assigned to the data type and should therefore only be occupied with Unlocated Variables.

ANL_OUT
ANL_OUT stands for the "Analog Output" data type and is used when processing analog values. The 4x-References for the configured analog output module, which were specified in the I/O component list, are automatically assigned to the data type and should therefore only be occupied with Unlocated Variables.

ANY
In the present version, "ANY" covers the BOOL, BYTE, DINT, INT, REAL, UDINT, UINT, TIME and WORD elementary data types and related Derived Data Types.

ANY_BIT In the present version, "ANY_BIT" covers the BOOL, BYTE and WORD data types.

ANY_ELEM In the present version, "ANY_ELEM" covers the BOOL, BYTE, DINT, INT, REAL, UDINT, UINT, TIME and WORD data types.

ANY_INT In the present version, "ANY_INT" covers the DINT, INT, UDINT and UINT data types.

ANY_NUM In the present version, "ANY_NUM" covers the DINT, INT, REAL, UDINT and UINT data types.

ANY_REAL In the present version, "ANY_REAL" covers the REAL data type.

Application Window The window contains the workspace, menu bar and the tool bar for the application program. The name of the application program appears in the title bar. An application window can contain several Document windows. In Concept the application window corresponds to a Project.

Argument Synonymous with Actual parameters.

ASCII-Mode The ASCII (American Standard Code for Information Interchange) mode is used to communicate with various host devices. ASCII works with 7 data bits.

Atrium The PC based Controller is located on a standard AT board, and can be operated within a host computer in an ISA bus slot. The module has a motherboard (requires SA85 driver) with two slots for PC104 daughter-boards. In this way, one PC104 daughter-board is used as a CPU and the other as the INTERBUS controller.

B

Backup file (Concept-EFB) The backup file is a copy of the last Source coding file. The name of this backup file is "backup??.c" (this is assuming that you never have more than 100 copies of the source coding file). The first backup file has the name "backup00.c". If you have made alterations to the Definitions file which do not cause any changes to the EFB interface, the generation of a backup file can be stopped by editing the source coding file (**Objects** → **Source**). If a backup file is created, the source file can be entered as the name.

Base 16 literals Base 16 literals are used to input whole number values into the hexadecimal system. The base must be denoted using the prefix 16#. The values can not have any signs (+/-). Single underscores (_) between numbers are not significant.

Example
16#F_F or 16#FF (decimal 255)
16#E_0 or 16#E0 (decimal 224)

Base 2 literals Base 2 literals are used to input whole number values into the dual system. The base must be denoted using the prefix 2#. The values can not have any signs (+/-). Single underscores (_) between numbers are not significant.

Example
2#1111_1111 or 2#11111111 (decimal 255)
2#1110_0000 or 2#11100000 (decimal 224)

Base 8 literals Base 8 literals are used to input whole number values in the octosystem. The base must be denoted using the prefix 8#. The values can not have any signs (+/-). Single underscores (_) between numbers are not significant.

Example
8#3_77 or 8#377 (decimal 255)
8#34_0 or 8#340 (decimal 224)

Binary Connections Connections between FFB outputs and inputs with the data type BOOL.

Bit sequence A data element, which consists of one or more bits.

BOOL BOOL stands for the data type "boolean". The length of the data element is 1 bit (occupies 1 byte in the memory). The value range for the variables of this data type is 0 (FALSE) and 1 (TRUE).

Bridge A bridge is a device which connects networks. It enables communication between nodes on two networks. Each network has its own token rotation sequence - the token is not transmitted via the bridge.

BYTE BYTE stands for the data type "bit sequence 8". Entries are made as base 2 literal, base 8 literal or base 16 literal. The length of the data element is 8 bits. A numerical value range can not be assigned to this data type.

C

Clipboard
The clipboard is a temporary memory for cut or copied objects. These objects can be entered in sections. The contents of the clipboard are overwritten with each new cut or copy.

Coil
A coil is a LD element which transfers the status of the horizontal connection on its left side, unchanged, to the horizontal connection on its right side. In doing this, the status is saved in the relevant variable/direct address.

Compact format (4:1)
The first digit (the Reference) is separated from the address that follows by a colon (:) where the leading zeros are not specified.

Constants
Constants are Unlocated variables, which are allocated a value that cannot be modified by the logic program (write protected).

Contact
A contact is a LD element, which transfers a status on the horizontal link to its right side. This status comes from the boolean AND link of the status of the horizontal link on the left side, with the status of the relevant variable/direct address. A contact does not change the value of the relevant variable/direct address.

D

Data transfer settings
Settings which determine how information is transferred from your programming device to the PLC.

Data Types
The overview shows the data type hierarchy, as used for inputs and outputs of functions and function blocks. Generic data types are denoted using the prefix "ANY".
- ANY_ELEM
  - ANY_NUM
    ANY_REAL (REAL)
    ANY_INT (DINT, INT, UDINT, UINT)
  - ANY_BIT (BOOL, BYTE, WORD)
  - TIME
- System Data types (IEC Extensions)
- Derived (from "ANY" data types)

DCP I/O drop

A remote network with a super-ordinate PLC can be controlled using a Distributed Control Processor (D908). When using a D908 with remote PLC, the super-ordinate PLC considers the remote PLC as a remote I/O drop. The D908 and the remote PLC communicate via the system bus, whereby a high performance is achieved with minimum effect on the cycle time. The data exchange between the D908 and the super-ordinate PLC takes place via the remote I/O bus at 1.5Mb per second. A super-ordinate PLC can support up to 31 D908 processors (addresses 2-32).

DDE (Dynamic Data Exchange)

The DDE interface enables a dynamic data exchange between two programs in Windows. The user can also use the DDE interface in the extended monitor to call up their own display applications. With this interface, the user (i.e. the DDE client) can not only read data from the extended monitor (DDE server), but also write data to the PLC via the server. The user can therefore alter data directly in the PLC, while monitoring and analyzing results. When using this interface, the user can create their own "Graphic Tool", "Face Plate" or "Tuning Tool" and integrate it into the system. The tools can be written in any language, i.e. Visual Basic, Visual C++, which supports DDE. The tools are invoked when the user presses one of the buttons in the Extended Monitor dialog field. Concept Graphic Tool: Configuration signals can be displayed as a timing diagram using the DDE connection between Concept and Concept Graphic Tool.

Declaration

Mechanism for specifying the definition of a language element. A declaration usually covers the connection of an identifier to a language element and the assignment of attributes such as data types and algorithms.

Definitions file (Concept-EFB)

The definitions file contains general descriptive information on the selected EFB and its formal parameters.

Defragmenting

With defragmenting, unanticipated gaps (e.g. resulting from deleting unused variables) are removed from memory.

Derived Data Type

Derived data types are data types, which are derived from Elementary Data Types and/or other derived data types. The definition of the derived data types is found in the Concept data type editor.
A distinction is made between global data types and local data types.

Derived Function Block (DFB)
A derived function block represents the invocation of a derived function block type. Details of the graphic form of the invocation can be found in the "Functional block (instance)". In contrast to the invocation of EFB types, invocations of DFB types are denoted by double vertical lines on the left and right hand side of the rectangular block symbol.
The output side of a derived function block is created in FBD language, LD language, ST language, IL language, but only in the current version of the programming system. Derived functions can also not be defined in the current version.
A distinction is made between local and global DFBs.

DFB Code
The DFB code is the section's DFB code which can be executed. The size of the DFB code is mainly dependent upon the number of blocks in the section.

DFB instance data
The DFB instance data is internal data from the derived function blocks used in the program.

DINT
DINT stands for the data type "double length whole number (double integer)". Entries are made as integer literal, base 2 literal, base 8 literal or base 16 literal. The length of the data element is 32 bits. The value range for variables of this data type reaches from -2 exp (31) to 2 exp (31) -1.

Direct Representation
A method of displaying variables in the PLC program, from which the assignment to the logical memory can be directly - and indirectly to the physical memory - derived.

Document Window
A window within an application window. Several document windows can be open at the same time in an application window. However, only one document window can ever be active. Document windows in Concept are, for example, sections, the message window, the reference data editor and the PLC configuration.

DP (PROFIBUS)
DP = Remote Peripheral

Dummy
An empty file, which consists of a text heading with general file information, such as author, date of creation, EFB designation etc. The user must complete this dummy file with further entries.

DX Zoom
This property enables the user to connect to a programming object, to monitor and, if necessary change, its data value.

E

EFB code

The EFB code is the executable code of all EFBs used. In addition the used EFBs count in DFBs.

Elementary functions/ function blocks (EFB)

Identifier for Functions or Function blocks, whose type definitions are not formulated in one of the IEC languages, i.e. whose body for example can not be modified with the DFB editor (Concept-DFB). EFB types are programmed in "C" and are prepared in a pre-compiled form using libraries.

EN / ENO (Enable / Error signal)

If the value of EN is equal to "0" when the FFB is invoked, the algorithms that are defined by the FFB will not be executed and all outputs keep their previous values. The value of ENO is in this case automatically set to "0". If the value of EN is equal to "1", when the FFB is invoked, the algorithms which are defined by the FFD will be executed. After the error-free execution of these algorithms, the value of ENO is automatically set to "1". If an error occurs during the execution of these algorithms, ENO is automatically set to "0". The output behavior of the FFB is independent of whether the FFBs are invoked without EN/ENO or with EN=1. If the EN/ENO display is switched on, it is imperative that the EN input is switched on. Otherwise, the FFB is not executed. The configuration of EN and ENO is switched on or off in the Block Properties dialog box. The dialog box can be invoked with the **Objects** → **Properties...**menu command or by double-clicking on the FFB.

Error

If an error is recognized during the processing of a FFB or a step (e.g. unauthorized input values or a time error), an error message appears, which can be seen using the **Online** → **Event Viewer...**menu command. For FFBs, the ENO output is now set to "0".

Evaluation

The process, through which a value is transmitted for a Function or for the output of a Function block during Program execution.

Expression

Expressions consist of operators and operands.

F

FFB (Functions/ Function blocks)

Collective term for EFB (elementary functions/function blocks) and DFB (Derived function blocks)

Field variables A variable, which is allocated a defined derived data type with the key word ARRAY (field). A field is a collection of data elements with the same data type.

FIR Filter (Finite Impulse Response Filter) a filter with finite impulse answer

Formal parameters Input / Output parameters, which are used within the logic of a FFB and led out of the FFB as inputs/outputs.

Function (FUNC) A program organization unit, which supplies an exact data element when processing. a function has no internal status information. Multiple invocations of the same function using the same input parameters always supply the same output values.
Details of the graphic form of the function invocations can be found in the definition "Functional block (instance)". In contrast to the invocations of the function blocks, function invocations only have a single unnamed output, whose name is the same as the function. In FBD each invocation is denoted by a unique number via the graphic block, this number is automatically generated and can not be altered.

Function block (Instance) (FB) A function block is a program organization unit, which correspondingly calculates the functionality values that were defined in the function block type description, for the outputs and internal variable(s), if it is invoked as a certain instance. All internal variable and output values for a certain function block instance remain from one function block invocation to the next. Multiple invocations of the same function block instance with the same arguments (input parameter values) do not therefore necessarily supply the same output value(s).
Each function block instance is displayed graphically using a rectangular block symbol. The name of the function block type is stated in the top center of the rectangle. The name of the function block instance is also stated at the top, but outside of the rectangle. It is automatically generated when creating an instance, but, depending on the user's requirements, it can be altered by the user. Inputs are displayed on the left side of the block and outputs are displayed on the right side. The names of the formal input/output parameters are shown inside the rectangle in the corresponding places.
The above description of the graphic display is especially applicable to the function invocations and to DFB invocations. Differences are outlined in the corresponding definitions.

Function Block Dialog (FBD) One or more sections, which contain graphically displayed networks from Functions, Function blocks and Connections.

Function block type A language element, consisting of: 1. the definition of a data structure, divided into input, output and internal variables; 2. a set of operations, which are performed with elements of the data structure, when a function block type instance is invoked. This set of operations can either be formulated in one of the IEC languages (DFB type) or in "C" (EFB type). A function block type can be instanced (invoked) several times.

Function Number
The function number is used to uniquely denote a function in a program or DFB. The function number can not be edited and is automatically assigned. The function number is always formed as follows: .n.m

n = Number of the section (consecutive numbers)
m = Number of the FFB object in the section (current number)

G

Generic Data Type
A data type, which stands in place of several other data types.

Generic literals
If the literal's data type is not relevant, simply specify the value for the literal. If this is the case, Concept automatically assigns the literal a suitable data type.

Global Data
Global data are Unlocated variables.

Global derived data types
Global derived data types are available in each Concept project and are occupied in the DFB directory directly under the Concept directory.

Global DFBs
Global DFBs are available in each Concept project. The storage of the global DFBs is dependant upon the settings in the CONCEPT.INI file.

Global macros
Global macros are available in each Concept project and are stored in the DFB directory directly under the Concept directory.

Groups (EFBs)
Some EFB libraries (e.g. the IEC library) are divided into groups. This facilitates locating the EFBs especially in expansive libraries.

H

Host Computer
Hardware and software, which support programming, configuring, testing, operating and error searching in the PLC application as well as in a remote system application, in order to enable source documentation and archiving. The programming device can also be possibly used for the display of the process.

I

I/O Map The I/O and expert modules from the various CPUs are configured in the I/O map.

Icon Graphical representation of different objects in Windows, e.g. drives, application programs and document windows.

IEC 61131-3 International standard: Programmable Logic Controls - Part 3: Programming languages.

IEC Format (QW1) There is an IEC type designation in initial position of the address, followed by the five-figure address.
- %0x12345 = %Q12345
- %1x12345 = %I12345
- %3x12345 = %IW12345
- %4x12345 = %QW12345

IEC name conventions (identifier) An identifier is a sequence of letters, numbers and underscores, which must begin with either a letter or underscore (i.e. the name of a function block type, an instance, a variable or a section). Letters of a national typeface (i.e.: ö,ü, é, õ) can be used, except in project and DFB names.
Underscores are significant in identifiers; e.g. "A_BCD" and "AB_CD" are interpreted as two separate identifiers. Several leading and multiple successive underscores are not allowed.
Identifiers should not contain any spaces. No differentiation is made between upper and lower case, e.g. "ABCD" and "abcd" are interpreted as the same identifier.
Identifiers should not be Keywords.

IEC Program Memory The IEC program memory consists of the program code, EFB code, the section data and the DFB instance data.

IIR Filter (Infinite Impulse Response Filter) a filter with infinite impulse answer

Initial step The first step in a sequence. A step must be defined as an initial step for each sequence. The sequence is started with the initial step when first invoked.

Initial value The value, which is allocated to a variable when the program is started. The values are assigned in the form of literals.

Input bits (1x references)

The 1/0 status of the input bits is controlled via the process data, which reaches from an input device to the CPU.

> **Note:** The x, which follows the initial reference type number, represents a five-figure storage location in the user data memory, i.e. the reference 100201 signifies an output or marker bit at the address 201 in the State RAM.

Input parameter (Input)

Upon invocation of a FFB, this transfers the corresponding argument.

Input words (3x references)

An input word contains information, which originates from an external source and is represented by a 16 bit number. A 3x register can also contain 16 sequential input bits, which were read into the register in binary or BCD (binary coded decimal) format. Note: The x, which follows the initial reference type number, represents a five-figure storage location in the user data memory, i.e. the reference 300201 signifies a 16-bit input word at the address 201 in the State RAM.

Instance Name

An identifier, which belongs to a certain function block instance. The instance name is used to clearly denote a function block within a program organization unit. The instance name is automatically generated, but it can be edited. The instance name must be unique throughout the whole program organization unit, and is not case sensitive. If the name entered already exists, you will be warned and you will have to choose another name. The instance name must comply with the IEC name conventions otherwise an error message appears. The automatically generated instance name is always formed as follows: FBI_n_m

FBI = Function Block Instance
n = Number of the section (consecutive numbers)
m = Number of the FFB object in the section (current number)

Instancing

Generating an Instance.

Instruction (IL)

Instructions are the "commands" of the IL programming language. Each instruction begins on a new line and is performed by an operator with a modifier if necessary, and if required for the current operation, by one or more operands. If several operands are used, they are separated by commas. A character can come before the instruction, which is then followed by a colon. The comment must, if present, be the last element of the line.

Instruction (LL984) When programming electrical controls, the user must implement operation-coded instructions in the form of picture objects, which are divided into a recognizable contact form. The designed program objects are, on a user level, converted to computer usable OP codes during the download process. The OP codes are decoded in the CPU and processed by the firmware functions of the controller in a way that the required control is implemented.

Instruction (ST) Instructions are "commands" of the ST programming language. Instructions must be completed by semicolons. Several instructions can be entered in one line (separated by semicolons).

Instruction list (IL) IL is a text language according to IEC 1131, which is shown in operations, i.e. conditional or unconditional invocations of Functions blocks and Functions, conditional or unconditional jumps etc. through instructions.

INT INT stands for the data type "whole number (integer)". Entries are made as integer literal, base 2 literal, base 8 literal or base 16 literal. The length of the data element is 16 bits. The value range for variables of this datatype reaches from -2 exp (15) to 2 exp (15) -1.

Integer literals Integer literals are used to input whole number values into the decimal system. The values can have a preceding sign (+/-). Single underscores (_) between numbers are not significant.

Example
-12, 0, 123_456, +986

INTERBUS (PCP) The new INTERBUS (PCP) I/O drop type is entered into the Concept configurator, to allow use of the INTERBUS PCP channel and the INTERBUS process data pre-processing (PDV). This I/O drop type is assigned the INTERBUS switching module 180-CRP-660-01.
The 180-CRP-660-01 differs from the 180-CRP-660-00 only in the fact that it has a clearly larger I/O range in the control state RAM.

Invocation The process by which the execution of an operation is initiated.

J

Jump Element of the SFC language. Jumps are used to skip zones in the sequence.

K

Keywords Keywords are unique combinations of characters, which are used as special syntactical components, as defined in Appendix B of the IEC 1131-3. All keywords which are used in the IEC 1131-3 and therefore in Concept, are listed in Appendix C of the IEC 1131-3. These keywords may not be used for any other purpose, i.e. not as variable names, section names, instance names etc.

L

Ladder Diagram (LD) Ladder Diagram is a graphic programming dialog according to IEC1131, which is optically oriented to the "rung" of a relay contact plan.

Ladder Logic 984 (LL) The terms Ladder Logic and Ladder Diagram refer to the word Ladder being executed. In contrast to a circuit diagram, a ladder diagram is used by electrotechnicians to display an electrical circuit (using electrical symbols), which should show the course of events and not the existing wires, which connect the parts with each other. A usual user interface for controlling the actions of automation devices permits a Ladder Diagram interface, so that electrotechnicians do not have to learn new programming languages to be able to implement a control program.
The structure of the actual Ladder Diagram enables the connection of electric elements in such a way that generates a control output, which is dependent upon a logical power flow through used electrical objects, which displays the previously requested condition of a physical electrical device.
In simple form, the user interface is a video display processed by the PLC programming application, which sets up a vertical and horizontal grid in which programming objects are classified. The diagram contains the power grid on the left side, and when connected to activated objects, the power shifts from left to right.

Landscape Landscape means that when looking at the printed text, the page is wider than it is high.

Language Element Every basic element in one of the IEC programming languages, e.g. a step in SFC, a function block instance in FBD or the initial value of a variable.

Library Collection of software objects, which are intended for re-use when programming new projects, or even building new libraries. Examples are the libraries of the Elementary function block types.
EFB libraries can be divided up into Groups.

Link
A control or data flow connection between graphical objects (e.g. steps in the SFC Editor, function blocks in the FBD Editor) within a section, represented graphically as a line.

Literals
Literals are used to provide FFB inputs, and transition conditions etc with direct values. These values can not be overwritten by the program logic (write-protected). A distinction is made between generic and standardized literals.
Literals are also used to allocate, to a constant, a value or a variable, an initial value. Entries are made as base 2 literal, base 8 literal, base 16 literal, integer literal, real literal or real literal with exponent.

Local derived data types
Local derived data types are only available in a single Concept project and the local DFBs and are placed in the DFB directory under the project directory.

Local DFBs
Local DFBs are only available in a single Concept project and are placed in the DFB directory under the project directory.

Local Link
The local network is the network, which connects the local nodes with other nodes either directly or through bus repeaters.

Local macros
Local macros are only available in a single Concept project and are placed in the DFB directory under the project directory.

Local network nodes
The local node is the one which is currently being configured.

Located variable
A state RAM address (reference addresses 0x, 1x, 3x,4x) is allocated to located variables. The value of these variables is saved in the state RAM and can be modified online using the reference data editor. These variables can be addressed using their symbolic names or their reference addresses.

All inputs and outputs of the PLC are connected to the state RAM. The program can only access peripheral signals attached to the PLC via located variables. External access via Modbus or Modbus Plus interfaces of the PLC, e.g. from visualization systems, is also possible via located variables.

M

Macro

Macros are created with the help of the Concept DFB software.
Macros are used to duplicate frequently used sections and networks (including their logic, variables and variable declaration).
A distinction is made between local and global macros.

Macros have the following properties:
- Macros can only be created in the FBD and LD programming languages.
- Macros only contain one section.
- Macros can contain a section of any complexity.
- In programming terms, there is no difference between an instanced macro, i.e. a macro inserted into a section and a conventionally created section.
- DFB invocation in a macro
- Declaring variables
- Using macro-specific data structures
- Automatic transfer of the variables declared in the macro.
- Initial values for variables
- Multiple instancing of a macro in the entire program with differing variables
- The name of the section, variable names and data structure names can contain up to 10 different exchange marks (@0 to @9).

MMI

Man-Machine-Interface

Multi element variables

Variables to which a Derived data type defined with STRUCT or ARRAY is allocated.
A distinction is made here between field variables and structured variables.

N

Network

A network is the collective switching of devices to a common data path, which then communicate with each other using a common protocol.

Network node

A node is a device with an address (1...64) on the Modbus Plus network.

Node

Node is a programming cell in a LL984 network. A cell/node consists of a 7x11 matrix, i.e. 7 rows of 11 elements.

Node Address The node address is used to uniquely denote a network node in the routing path. The address is set on the node directly, e.g. using the rotary switch on the back of the modules.

O

Operand An operand is a literal, a variable, a function invocation or an expression.

Operator An operator is a symbol for an arithmetic or boolean operation which is to be executed.

Output parameter (output): A parameter, through which the result(s) of the evaluation of a FFB is/are returned.

Output/Marker bits (0x references) An output/marker bit can be used to control real output data using an output unit of the control system, or to define one or more discrete outputs in the state RAM. Note: The x, which follows the initial reference type number, represents a five-figure storage location in the user data memory, i.e. the reference 000201 signifies an output or marker bit at the address 201 in the State RAM.

Output/marker words (4x references) An output / marker word can be used to save numerical data (binary or decimal) in the state RAM, or to send data from the CPU to an output unit in the control system. Note: The x, which follows the initial reference type number, represents a five-figure storage location in the user data memory, i.e. the reference 400201 signifies a 16 bit output or marker word at the address 201 in the State RAM.

P

Peer CPU The Peer CPU processes the token execution and the data flow between the Modbus Plus network and the PLC user logic.

PLC Memory programmable controller

Portrait Portrait means that the sides are larger than the width when printed.

Program The uppermost program organization unit. A program is closed on a single PLC download.

Program organization unit A function, a function block, or a Program. This term can refer to either a type or an instance.

Program redundancy system (Hot Standby) A redundancy system consists of two identically configured PLC machines, which communicate with one another via redundancy processors. In the case of a breakdown of the primary PLC, the secondary PLC takes over the control check. Under normal conditions, the secondary PLC does not take over the control function, but checks the status information, in order to detect errors.

Project General description for the highest level of a software tree structure, which specifies the super-ordinate project name of a PLC application. After specifying the project name you can save your system configuration and your control program under this name. All data that is created whilst setting up the configuration and program, belongs to this super-ordinate project for this specific automation task.
General description for the complete set of programming and configuration information in the project database, which represents the source code that describes the automation of a system.

Project database The database in the host computer, which contains the configuration information for a project.

Prototype file (Concept-EFB) The prototype file contains all the prototypes of the assigned functions. In addition, if one exists, a type definition of the internal status structure is specified.

R

REAL REAL stands for the data type "floating point number". The entry can be real-literal or real-literal with an exponent. The length of the data element is 32 bits. The value range for variables of this data type extends from +/-3.402823E+38.

> **Note:** Dependent on the mathematical processor type of the CPU, different ranges within this permissible value range cannot be represented. This applies to values that are approaching ZERO and for values that approach INFINITY. In these cases NAN (**N**ot **A** **N**umber) or INF (**INF**inite) will be displayed in the animation mode instead of a number value.

Real literals Real literals are used to input floating point values into the decimal system. Real literals are denoted by a decimal point. The values can have a preceding sign (+/-). Single underscores (_) between numbers are not significant.

Example
-12.0, 0.0, +0.456, 3.14159_26

Real literals with exponents

Real literals with exponents are used to input floating point values into the decimal system. Real literals with exponents are identifiable by a decimal point. The exponent indicates the power of ten, with which the existing number needs to be multiplied in order to obtain the value to be represented. The base can have a preceding negative sign (-). The exponent can have a preceding positive or negative sign (+/-). Single underscores (_) between numbers are not significant. (Only between characters, not before or after the decimal point and not before or after "E", "E+" or "E-")

Example
-1.34E-12 or -1.34e-12
1.0E+6 or 1.0e+6
1.234E6 or 1.234e6

Reference

Every direct address is a reference that begins with an indicator, which specifies whether it is an input or an output and whether it is a bit or a word. References that begin with the code 6, represent registers in the extended memory of the state RAM.
0x range = Output/Marker bits
1x range = Input bits
3x range = Input words
4x range = Output registers
6x range = Register in the extended memory

> **Note:** The x, which follows each initial reference type number, represents a five-digit storage location in the user data memory, i.e. the reference 400201 signifies a 16 bit output or marker word at the address 201 in the State RAM.

Register in the extended memory (6x-reference)

6x references are holding registers in the extended memory of the PLC. They can only be used with LL984 user programs and only with a CPU 213 04 or CPU 424 02.

Remote Network (DIO)

Remote programming in the Modbus Plus network enables maximum performance when transferring data and dispenses with the need for connections. Programming a remote network is simple. Setting up a network does not require any additional ladder logic to be created. All requirements for data transfer are fulfilled via corresponding entries in the Peer Cop Processor.

RIO (Remote I/O)

Remote I/O indicates a physical location of the I/O point controlling devices with regard to the CPU controlling them. Remote inp./outputs are connected to the controlling device via a twisted communication cable.

RTU-Mode

Remote Terminal Unit
The RTU mode is used for communication between the PLC and an IBM compatible personal computer. RTU works with 8 data bits.

Runtime error Errors, which appear during program processing on the PLC, in SFC objects (e.g. Steps) or FFBs. These are, for example, value range overflows for numbers or timing errors for steps.

S

SA85 module The SA85 module is a Modbus Plus adapter for IBM-AT or compatible computers.

Scan A scan consists of reading the inputs, processing the program logic and outputting the outputs.

Section A section can for example be used to describe the functioning mode of a technological unit such as a motor.
A program or DFB consists of one or more sections. Sections can be programmed with the IEC programming languages FBD and SFC. Only one of the named programming languages may be used within a section at any one time.
Each section has its own document window in Concept. For reasons of clarity, however, it is useful to divide a very large section into several small ones. The scroll bar is used for scrolling within a section.

Section Code Section Code is the executable code of a section. The size of the Section Code is mainly dependent upon the number of blocks in the section.

Section Data Section data is the local data in a section such as e.g. literals, connections between blocks, non-connected block inputs and outputs, internal status memory of EFBs.

> **Note:** Data which appears in the DFBs of this section is not section data.

Separator Format (4:00001) The first digit (the reference) is separated from the five-digit address that follows by a colon (:).

Sequence language (SFC) The SFC Language Elements enable a PLC program organization unit to be divided up into a number of Steps and Transitions, which are connected using directional Links. A number of actions belong to each step, and transition conditions are attached to each transition.

Serial Connections With serial connections (COM) the information is transferred bit by bit.

Source code file (Concept-EFB)

The source code file is a normal C++ source file. After executing the **Library** → **Create files** menu command, this file contains an EFB-code frame, in which you have to enter a specific code for the EFB selected. To do this invoke the **Objects** → **Source** menu command.

Standard Format (400001)

The five-digit address comes directly after the first digit (the reference).

Standardized literals

If you would like to manually determine a literal's data type, this may be done using the following construction: ’Data type name’#’value of the literal’.

Example
INT#15 (Data type: integer, value: 15),
BYTE#00001111 (Data type: byte, value: 00001111)
REAL#23.0 (Data type: real, value: 23.0)

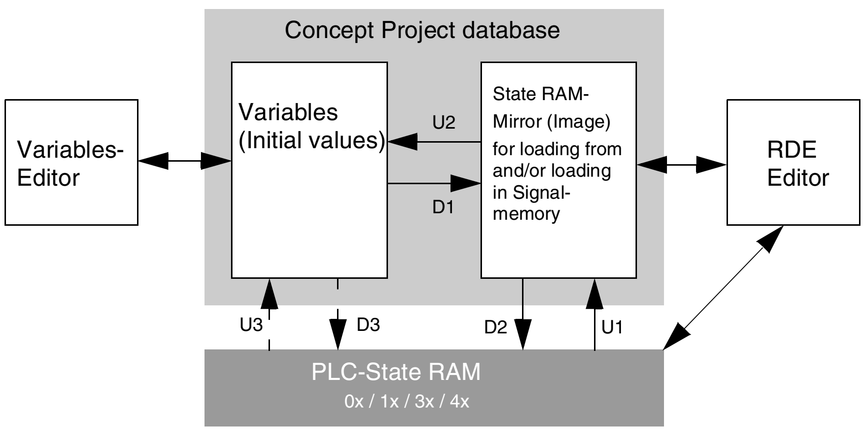
To assign the data type REAL, the value may also be specified in the following manner: 23.0.
Entering a comma will automatically assign the data type REAL.

State RAM

The state RAM is the memory space for all variables, which are accessed via References (Direct representation) in the user program. For example, discrete inputs, coils, input registers, and output registers are located in the state RAM.

State RAM overview for uploading and downloading

Overview:



Status Bits

For every device with global inputs or specific inputs/outputs of Peer Cop data, there is a status bit. If a defined group of data has been successfully transferred within the timeout that has been set, the corresponding status bit is set to 1. If this is not the case, this bit is set to 0 and all the data belonging to this group is deleted (to 0).

Step

SFC-language element: Situation, in which the behavior of a program, in reference to its inputs and outputs, follows those operations which are defined by the actions belonging to the step.

Step name

The step name is used to uniquely denote a step in a program organization unit. The step name is generated automatically, but it can be edited. The step name must be unique within the entire program organization unit, otherwise an error message will appear.
The automatically generated step name is always formed as follows: S_n_m

S = step
n = Number of the section (consecutive numbers)
m = Number of the step in the section (current number)

Structured text (ST)

ST is a text language according to IEC 1131, in which operations, e.g. invocations of Function blocks and Functions, conditional execution of instructions, repetitions of instructions etc. are represented by instructions.

Structured variables

Variables to which a Derived data type defined with STRUCT (structure) is allocated. A structure is a collection of data elements with generally different data types (elementary data types and/or derived data types).

SY/MAX

In Quantum control devices, Concept includes the preparation of I/O-map SY/MAX-I/O modules for remote controlling by the Quantum PLC. The SY/MAX remote backplane has a remote I/O adapter in slot 1, which communicates via a Modicon S908 R I/O System. The SY/MAX-I/O modules are executed for you for labeling and inclusion in the I/O map of the Concept configuration.

T

Template file (Concept-EFB)

The template file is an ASCII file with layout information for the Concept FBD Editor, and the parameters for code creation.

TIME

TIME stands for the data type "time". The entry is time literal. The length of the data element is 32 bits. The value range for variables of this data type extends from 0 to 2exp(32)-1. The unit for the data type TIME is 1 ms.

Time literals

Permissible units for times (TIME) are days (D), hours (H), minutes (M), seconds (S) and milliseconds (MS) or combinations of these. The time must be marked with the prefix t#, T#, time# or TIME#. The "overflow" of the unit with the highest value is permissible, e.g. the entry T#25H15M is allowed.

Example
t#14MS, T#14.7S, time#18M, TIME#19.9H, t#20.4D, T#25H15M, time#5D14H12M18S3.5MS

Token

The network "token" controls the temporary possession of the transfer right via a single node. The token passes round the nodes in a rotating (increasing) address sequence. All nodes follow the token rotation and can receive all the possible data that is sent with it.

Total IEC memory

The total IEC memory consists of the IEC program memory and the global data.

Traffic Cop

The traffic cop is an IO map, which is generated from the user-IO map. The traffic cop is managed in the PLC and in addition to the user IO map, contains e.g. status information on the I/O stations and modules.

Transition

The condition, in which the control of one or more predecessor steps passes to one or more successor steps along a directed link.

U

UDEFB

User-defined elementary functions/function blocks
Functions or function blocks, which were created in the C programming language, and which Concept provides in libraries.

UDINT

UDINT stands for the data type "unsigned double integer". Entries are made as integer literal, base 2 literal, base 8 literal or base 16 literal. The length of the data element is 32 bits. The value range for variables of this data type extends from 0 to 2exp(32)-1.

UINT

UINT stands for the data type "unsigned integer". Entries are made as integer literal, base 2 literal, base 8 literal or base 16 literal. The length of the data element is 16 bits. The value range for variables of this data type extends from 0 to (2exp 16)-1.

Unlocated variable

Unlocated variables are not allocated a state RAM address. They therefore do not occupy any state RAM addresses. The value of these variables is saved in the internal system and can be changed using the reference data editor. These variables are only addressed using their symbolic names.

Signals requiring no peripheral access, e.g. intermediate results, system tags etc., should be primarily declared as unlocated variables.

V

Variables

Variables are used to exchange data within a section, between several sections and between the program and the PLC.
Variables consist of at least one variable name and one data type.
If a variable is assigned a direct address (reference), it is called a located variable.
If the variable has no direct address assigned to it, it is called an unlocated variable.
If the variable is assigned with a derived data type, it is called a multi element variable.
There are also constants and literals.

W

Warning

If a critical status is detected during the processing of a FFB or a step (e.g. critical input values or an exceeded time limit), a warning appears, which can be seen using the **Online** → **Event Viewer...**menu command. For FFBs, the ENO remains set to "1".

WORD

WORD stands for the data type "bit sequence 16". Entries are made as base 2 literal, base 8 literal or base 16 literal. The length of the data element is 16 bits. A numerical value range can not be assigned to this data type.

Index

A

C

D

Q

R

S

T

V