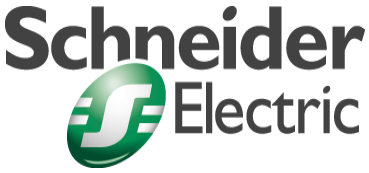
Concept
Block library IEC
Part: CONT_CTL

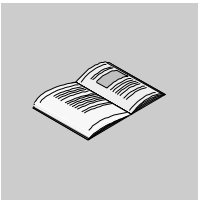
Volume 2

840 USE 504 00 eng Version 2.6

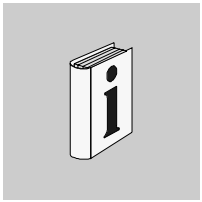


33002211.00

Table of Contents



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 2.5 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

User Comments We welcome your comments about this document. You can reach us by e-mail at TECHCOMM@modicon.com

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

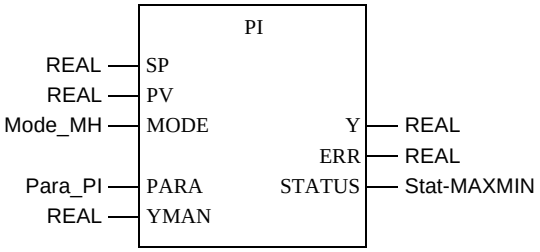
The function block has the following properties:

- 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

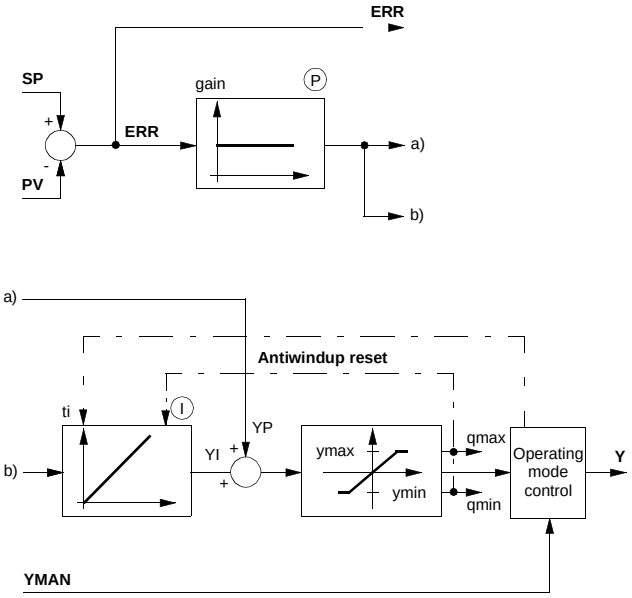
The meaning of the formula variables is given in the following table:

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. 217* 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. 248*).

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

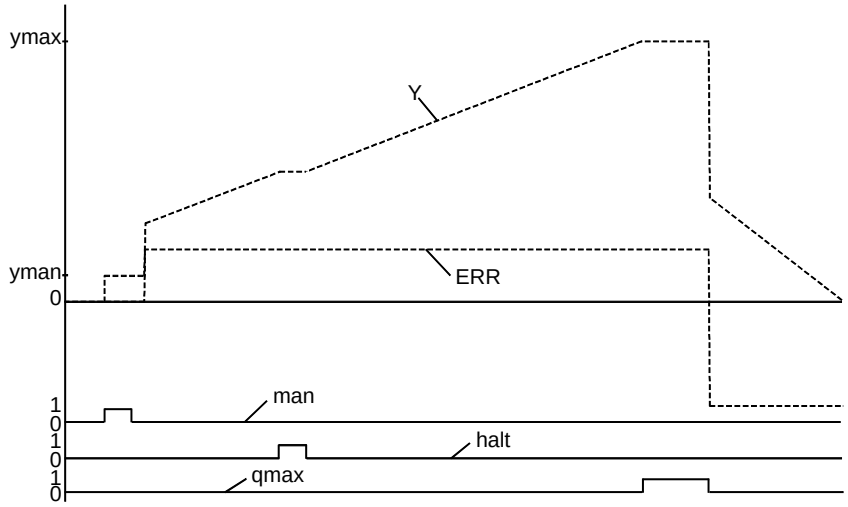
Example

The jump response of the PI controller is shown in the following Diagram (See *PI controller jump response, p. 220*) 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

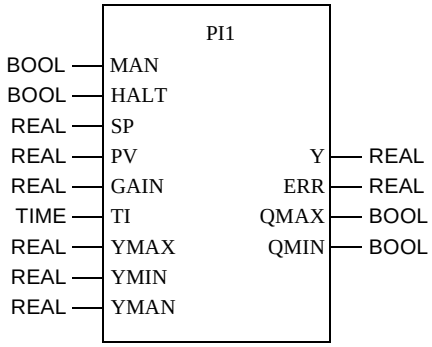
The function block has the following properties:

- 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

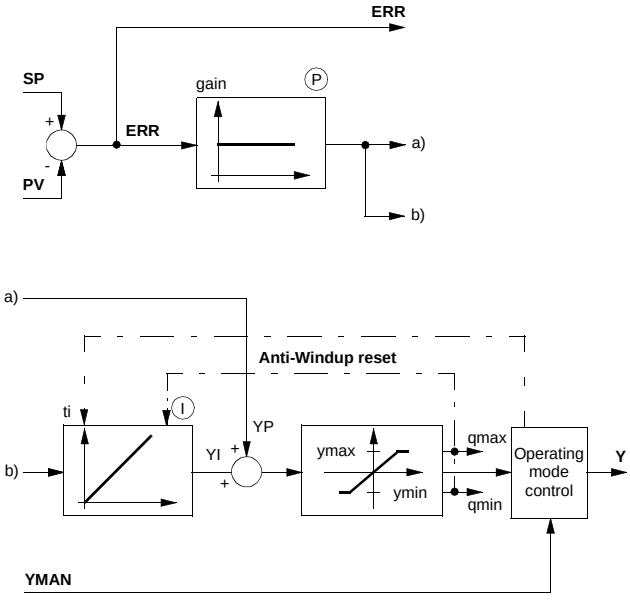
The meaning of the formula sizes is given in the following table:

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. 224* 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 ≤ YI ≤ 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. 260*).

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. 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.

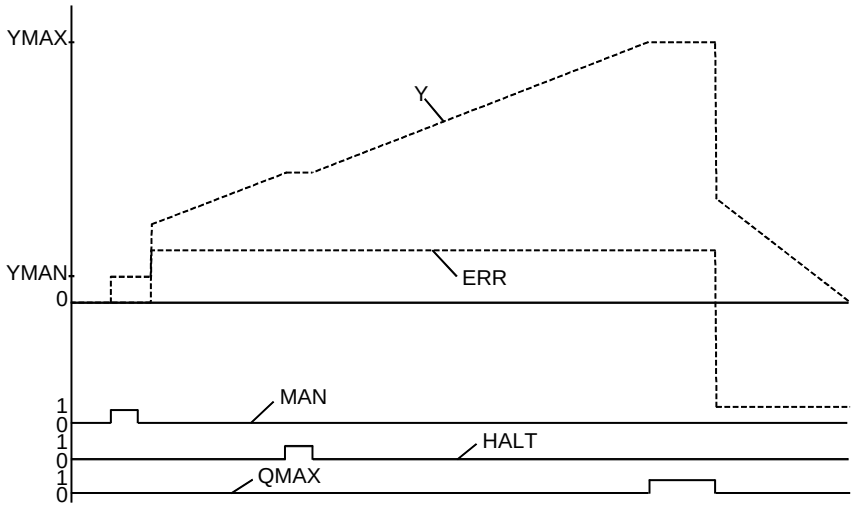
PI1 controller example

Example

The jump response of the PI1 controller is shown in the following Diagram (See *The jump response of the PI1 controller, p. 227*) 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 $\mathrm{GAIN} \times (\mathrm{ERR}_{(\mathrm{new})} - \mathrm{ERR}_{(\mathrm{old})})$; 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. 275*). 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

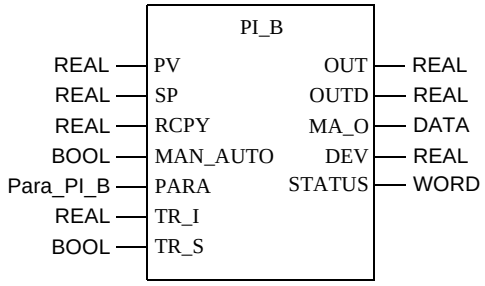
Extended functions

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. 15*) 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

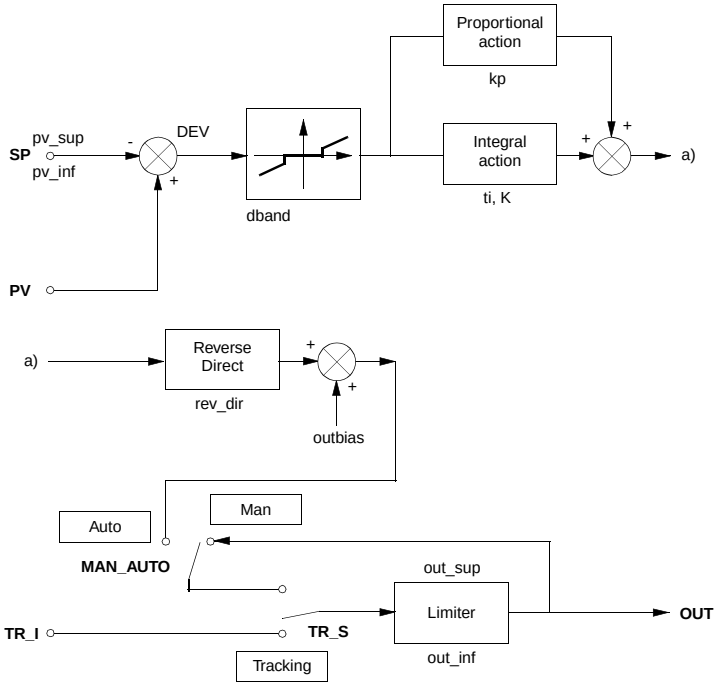
The meaning of the formula sizes is given in the following table:

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:

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

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

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

to

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

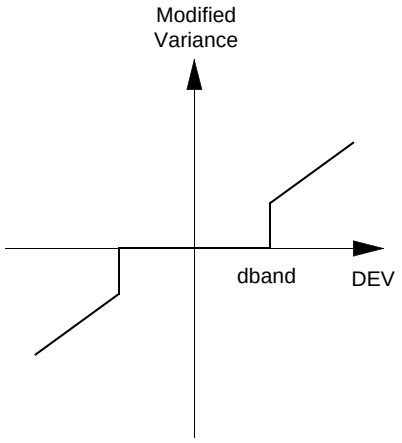
This is particularly useful for cascades or cascade-like controls.

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:

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

YD
YI
YP

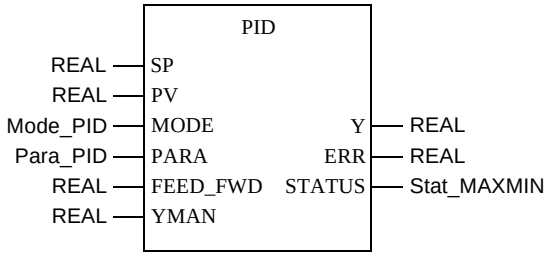
Explanation of the sizes:

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

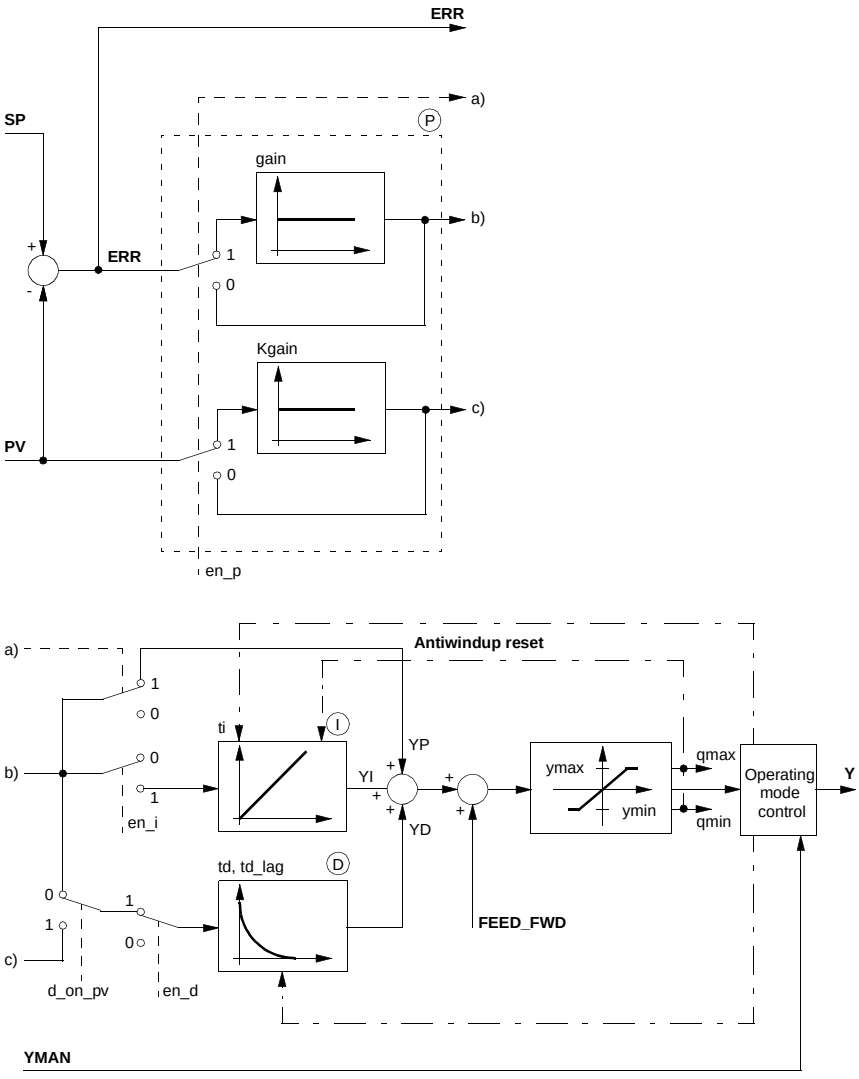
Data structure description

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. 245*.
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 $ymin \leq Y \leq ymax$.
The outputs qmax and qmin signal that the limit value has been reached, i.e. that the output signal is limited.

- qmax = 1 when $Y \geq ymax$
- qmin = 1 when $Y \leq ymin$

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:

$$ymin - YP - FEED\_FWD \leq YI \leq ymax - YP - FEED\_FWD$$

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.

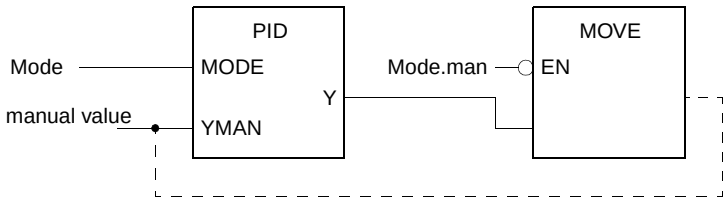
Switching from automatic to manual

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:

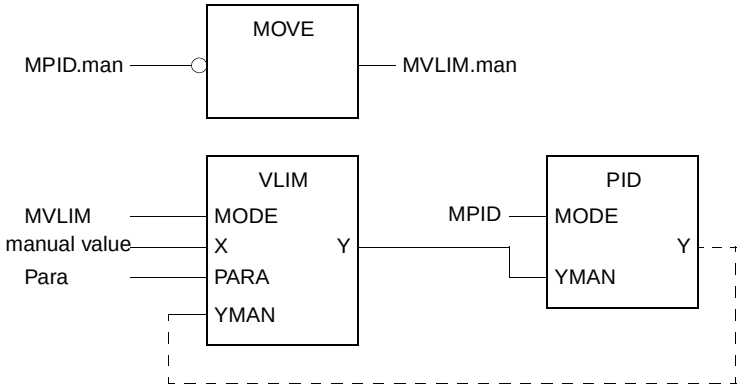


Note: This type of display was selected purely to facilitate comprehension. The links represented by a dotted line can not be programmed as Links (link objects), as they forme unauthorized (in Concept) loops. During programming the links must be implemented through changes.

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:

$$ymin \leq Y \leq ymax$$

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:

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

YD
YI
YP

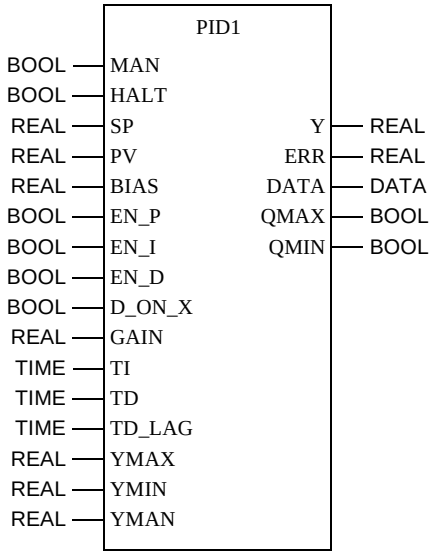
Explaining the sizes:

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:



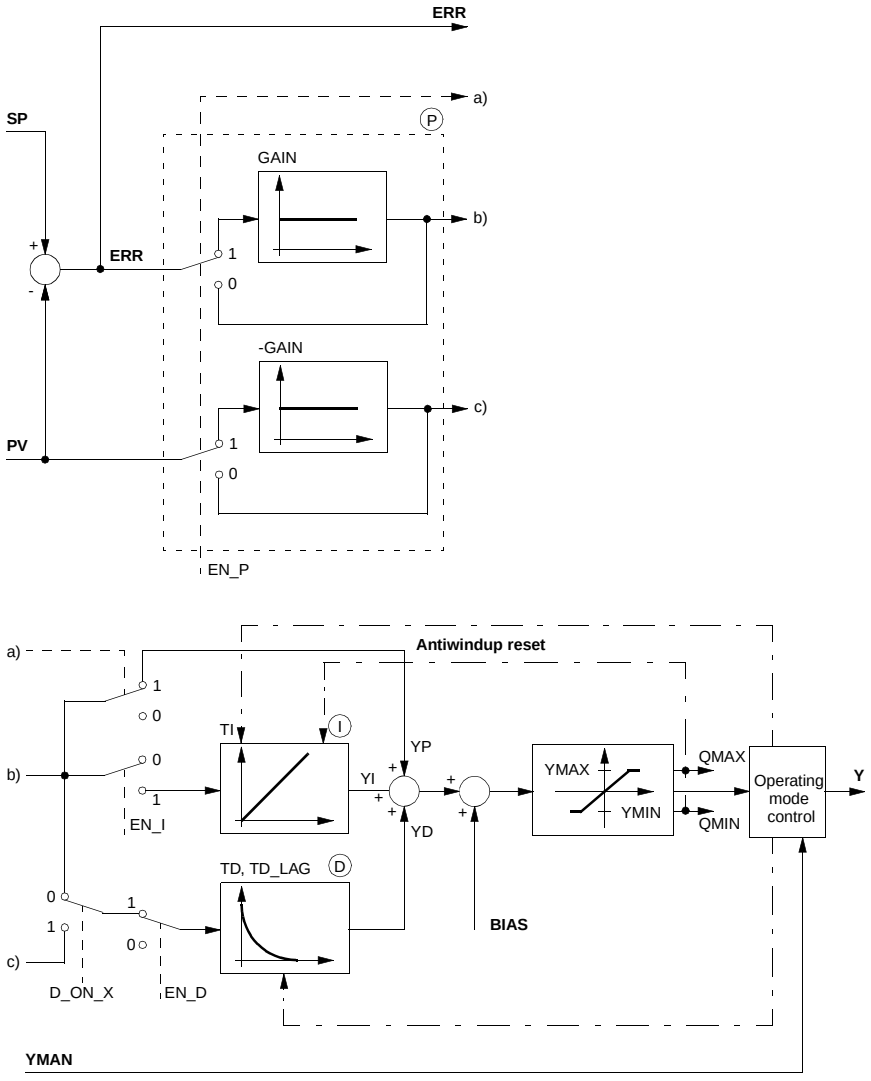
Parameter description

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. 257*.
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, $YMIN \leq Y \leq YMAX$.
The outputs QMAX and QMIN signal that the output has reached a limit, and thus been capped.

- QMAX = 1 if $Y \geq YMAX$
- QMIN = 1 if $Y \leq YMIN$

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:

$$YMIN - YP - BIAS \leq YI \leq YMAX - YP - BIAS$$

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. Halt mode is also useful in allowing an external operator device to adjust control output Y, whereby the controller's internal components are correctly tracked.
In this operating mode the D component is automatically set to 0.

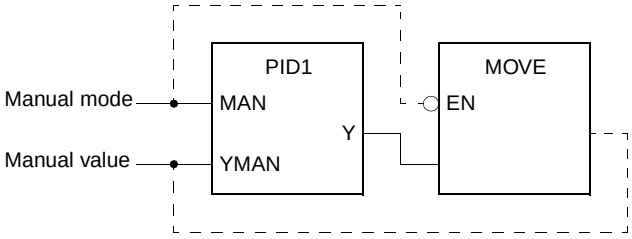
Switching from automatic to manual

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:

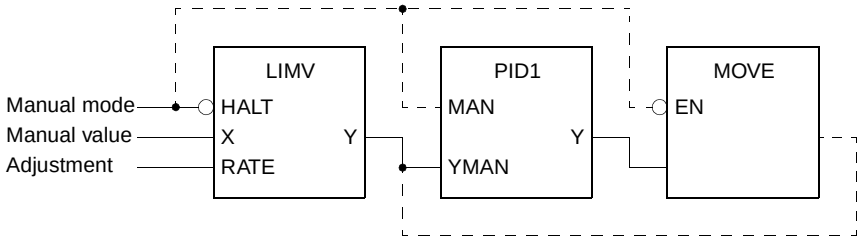


Note: This type of display as selected purely to facilitate comprehension. The links represented by a dotted line can not be programmed as Links (link objects), as they form unauthorized (in Concept) loops. In programming, the links must be established using variables.

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. 161*)):



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:

$$G(s) = kp + \frac{ki}{s} + \frac{kd \times s}{s + \frac{1}{td\_lag}}$$

YD
YI
YP

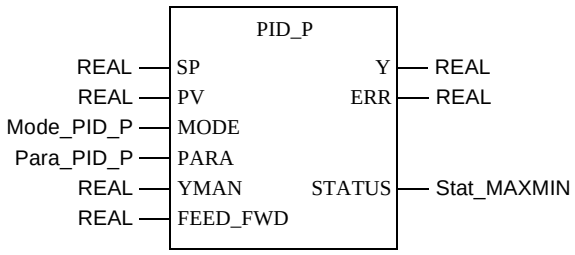
Explanation of the variables:

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
ymax	REAL	Upper limit
ymin	REAL	Lower limit

Parameter description Stat_MAXMIN

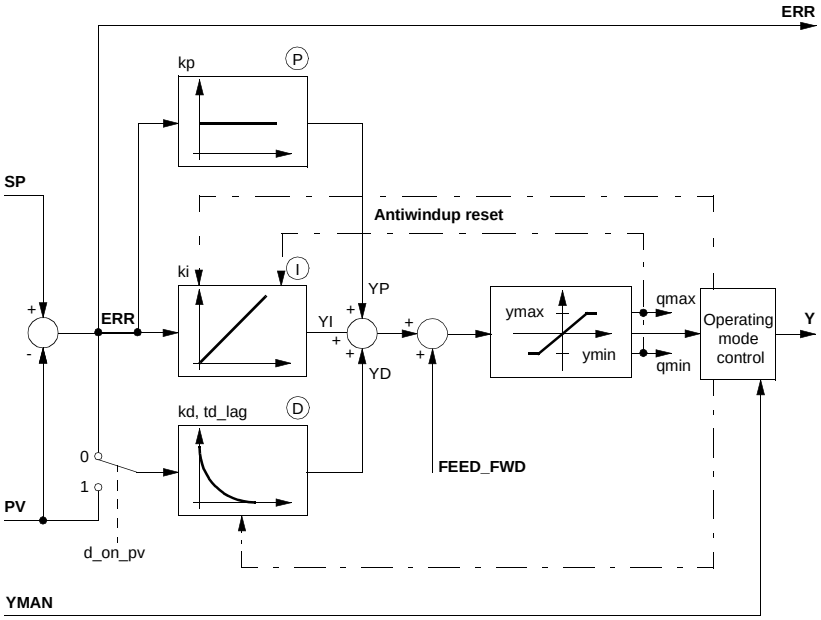
Data structure description

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. 269*.
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. 248*).

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:

$ymin \leq Y \leq ymax$

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

38

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

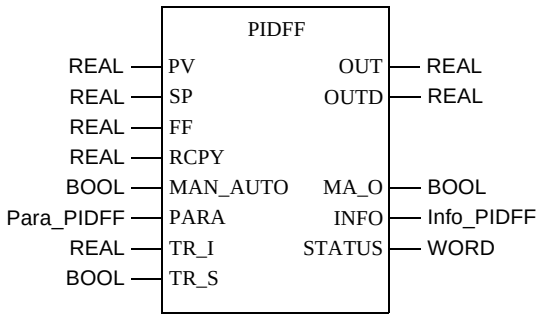
Other function blocks complement these functions when used in conjunction with the PIDFF block:

- 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. 15*) 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

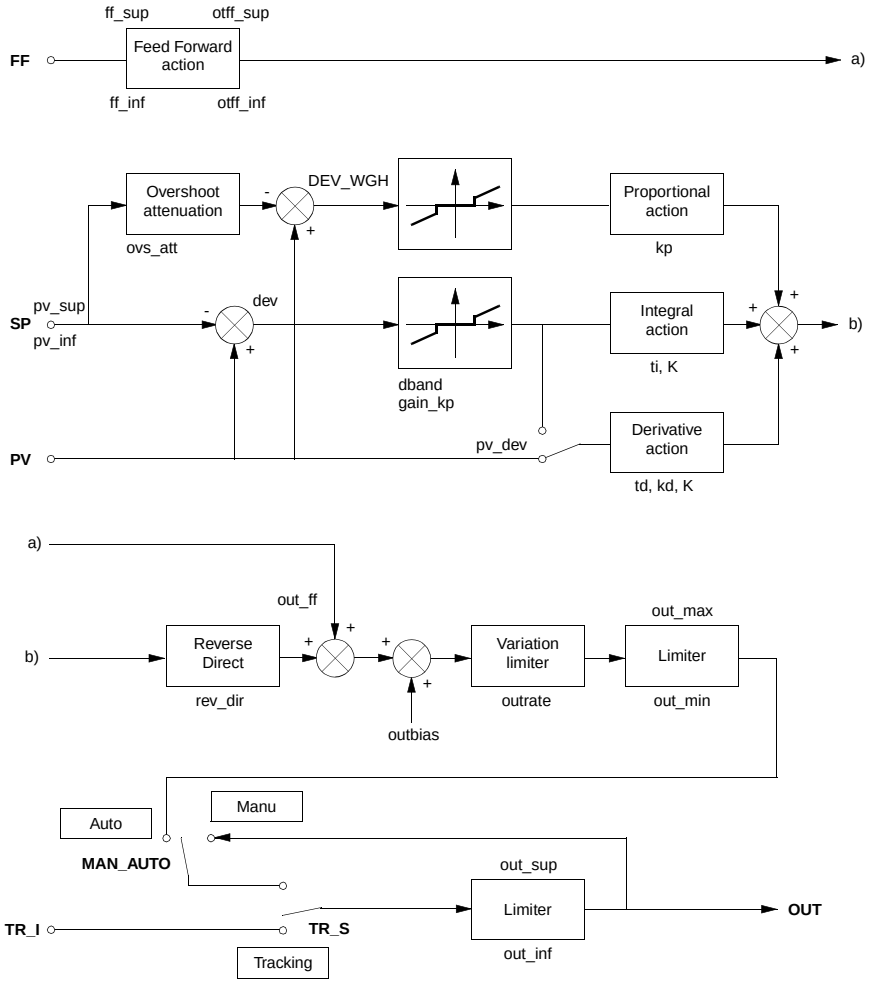
The meaning of the formula sizes is given in the following table:

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. 281*) 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:

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

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.

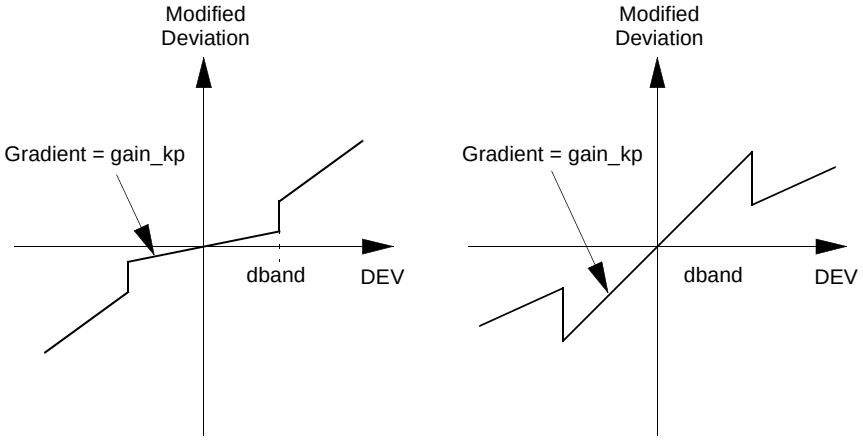
Weight of the setpoint in the proportional component (reducing the overrun)

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



0 <= gain_kp < 1 gain_kp > 1

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:

$$\text{out\_ff} = \frac{(\text{FF} - \text{ff\_inf}) \times (\text{otff\_sup} - \text{otff\_inf})}{(\text{ff\_sup} - \text{ff\_inf})} + \text{otff\_inf}$$

A specific user example of this function is given in the section "*Application example of the feed forward function , p. 294*".

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. 289*
- *Incremental algorithm PID controller, p. 290*
  - Normal incremental algorithms (aw_type = 0)
  - With bumpless antiwindup measure (aw_type = 1)
- *Incremental algorithms in integral mode, p. 292*
  - 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;

$$OUTD = TermP + TermI + TermD + TermFF + TermAW$$

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

$$OUT = limiter(OUTc)$$

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

$$TermFF = \Delta\left[\frac{(FF - ff\_inf) \times (otff\_sup - otff\_inf)}{(ff\_sup - ff\_inf)} + otff\_inf\right]$$

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

$$TermAW = \frac{dt}{ti}[RCPY - OUTc(old)]$$

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;

$$OUTD = TermI + TermFF$$

$$OUT = limiter(OUT)$$

If en_rcpy = 0, then

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

If en_rcpy = 1, then

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

Value of the integral component TermI:

$$TermI = sense \times \alpha \times \frac{dt}{ti} \times dev$$

Value of the feed forward component TermFF

$$TermFF = \Delta\left[\frac{(FF - ff\_inf) \times (otff\_sup - otff\_inf)}{(ff\_sup - ff\_inf)} + otff\_inf\right]$$

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

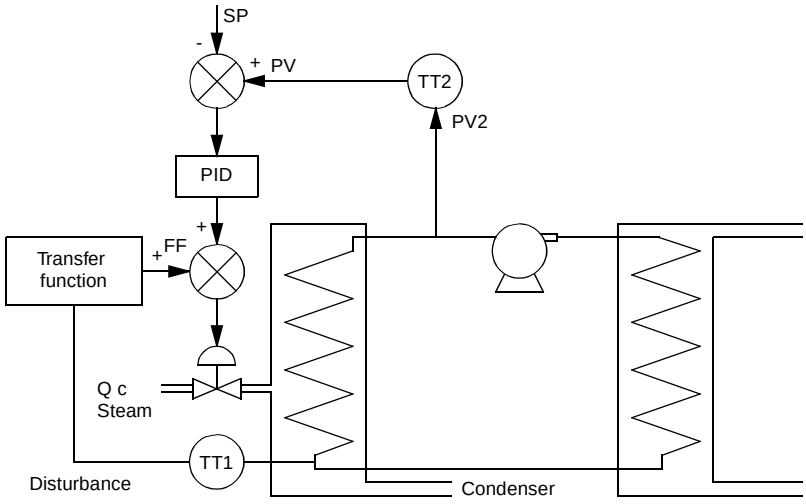
Example-overview

This chapter contains the following examples:

- *Application example of the feed forward function , p. 294*
- Classic control examples programmed via the PIDFF function block:
  - *Example of the cascaded arrangement of two controllers, p. 296*
  - *Example of cascade-like control, p. 298*

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:

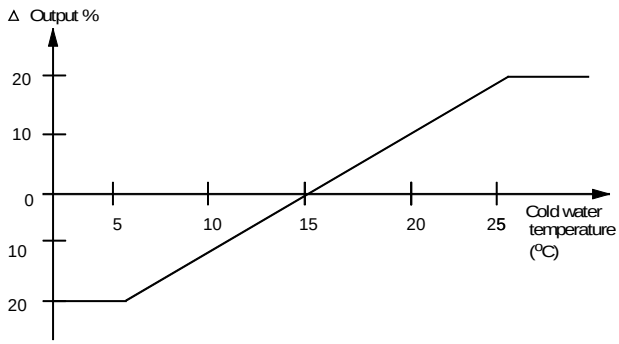


The following hypotheses are accepted:

- 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

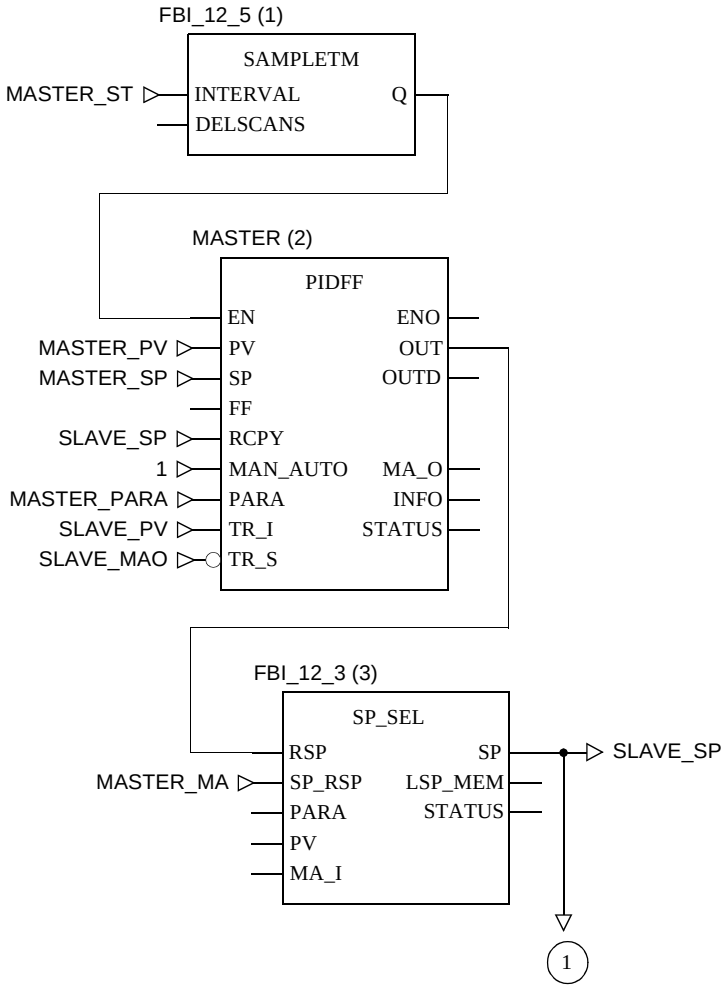
Temperature range	Effects
10% per 5 °C between 5 and 25 °C	

Adjustments to be pre-set

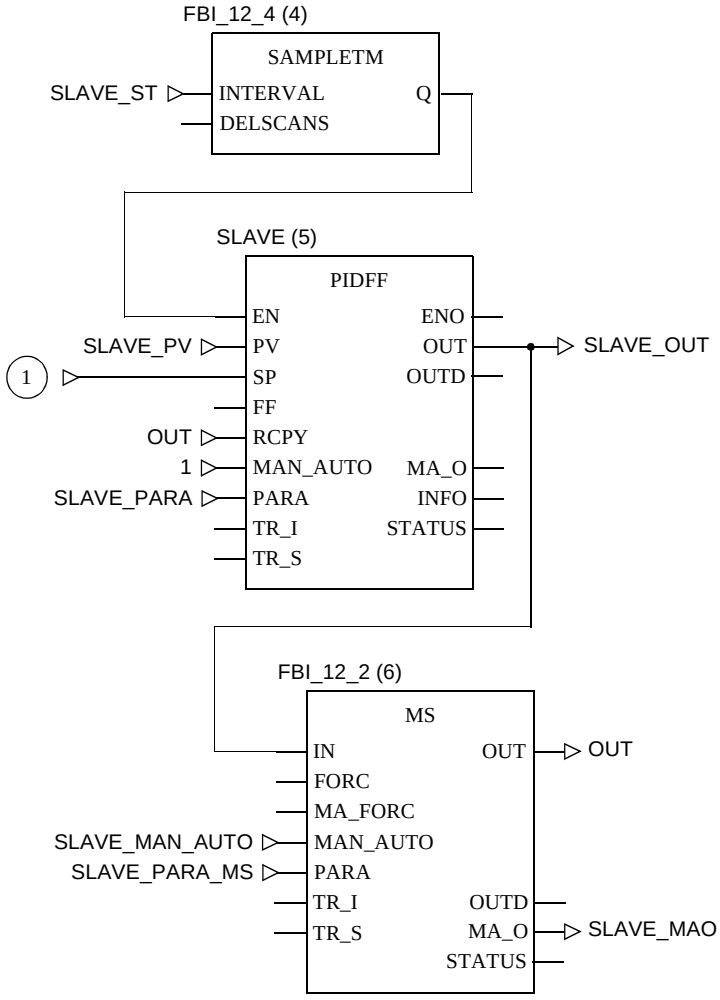
Element	Value
ff_sup	25 °C
ff_inf	5 °C
otff_sup	10 %
otff_inf	- 10 %

Example of the cascaded arrangement of two controllers

A representation of the function map, part 1, follows:

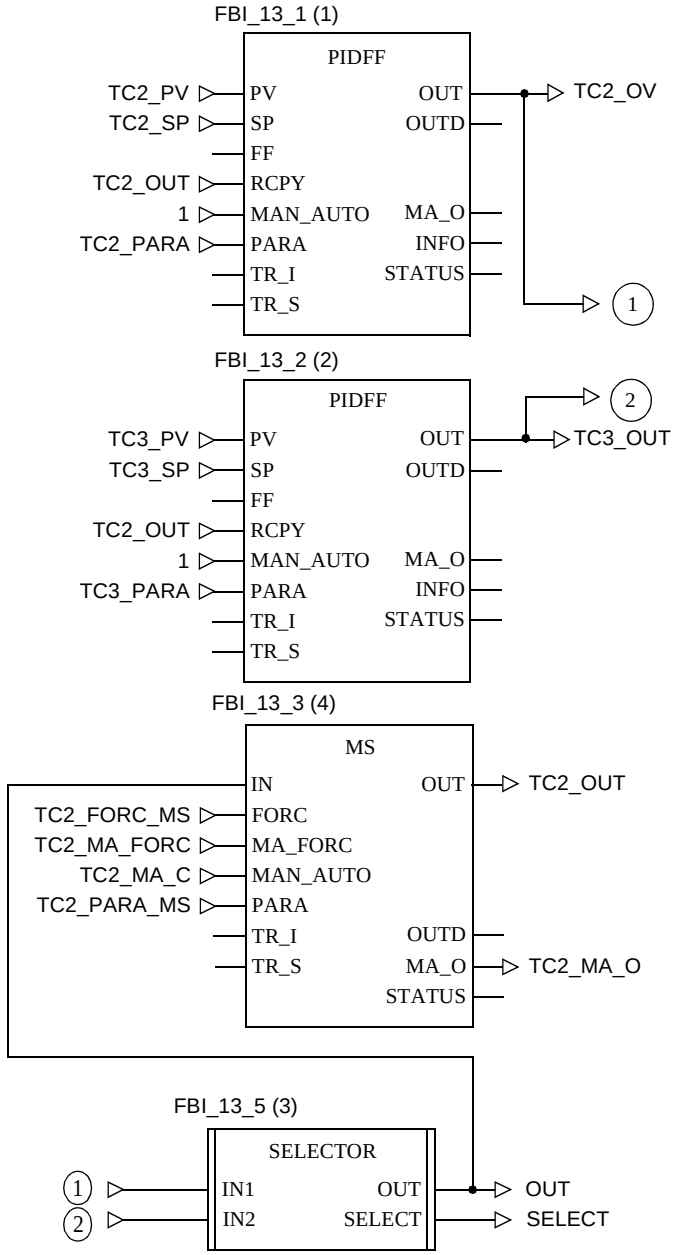


A representation of the function map, part 2, follows:



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

39

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:

$$G(s) = KP + \frac{KI}{s} + \frac{KD \times s}{s + \frac{1}{TD\_LAG}}$$

YD
YI
YP

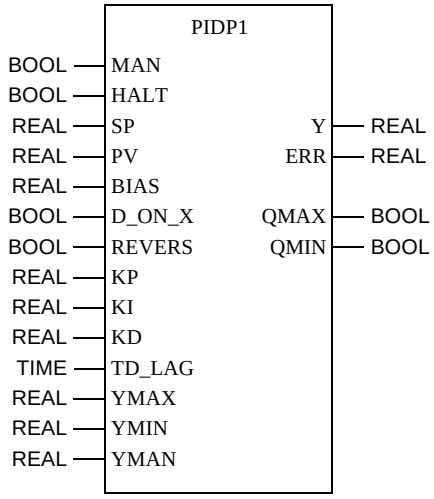
Explanation of the sizes:

Variable	Meaning
YD	D component
YI	I component
YP	P component

Representation

Symbol

Block representation:



Parameter description

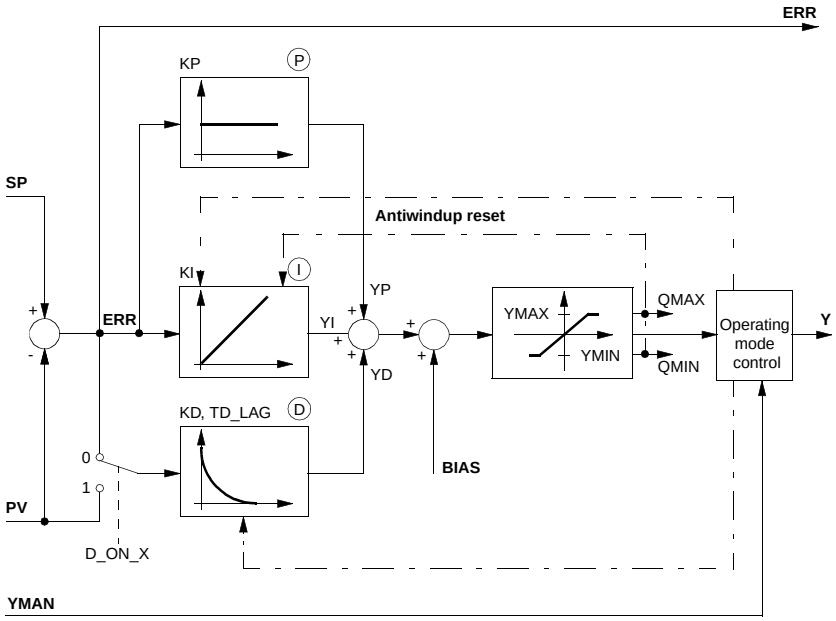
Block parameter description

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. 305*.
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:

$$YMIN - YP - BIAS \leq YI \leq YMAX - YP - BIAS$$

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. 260*).

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. Halt mode is also useful in allowing an external operator device to adjust control output Y, whereby the controller's internal components are correctly tracked.
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:

$$YMIN \leq Y \leq YMAX$$

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

40

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

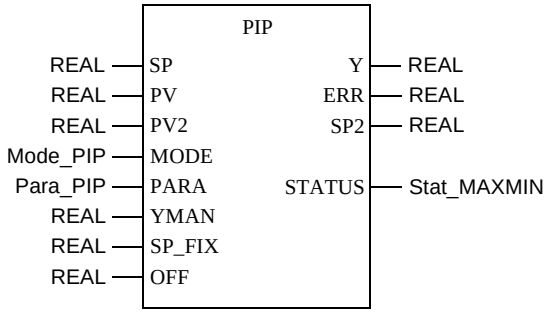
The proportional action coefficient of the master controller is determined as follows:

$$YP = gain1 \times ERR$$

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

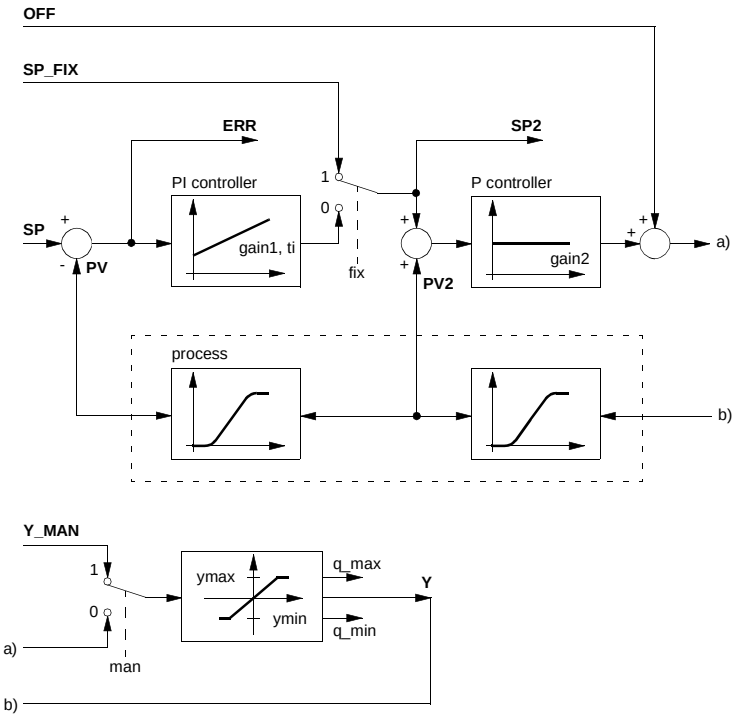
Data structure description

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

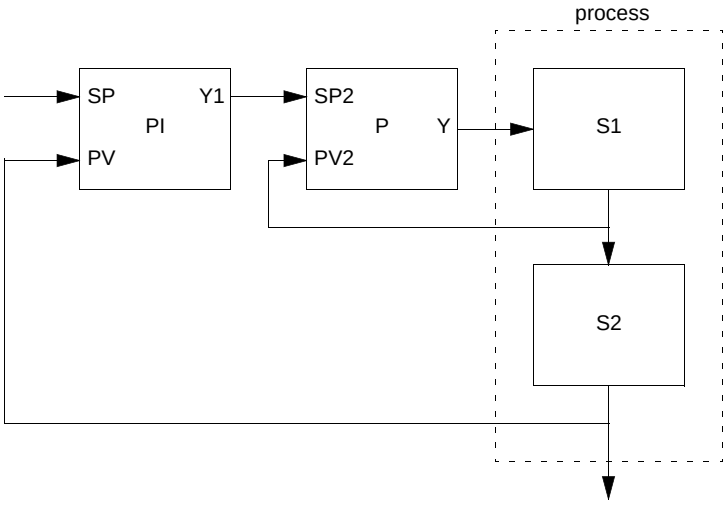
There follows now the structure diagram of the PIP block:



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. 316* .
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:
$ymin \leq Y \leq ymax$

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. 248*).

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

41

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

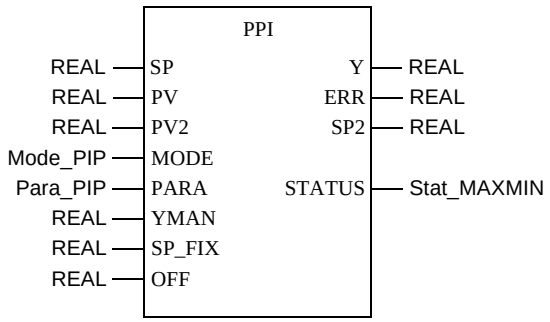
The proportional action coefficient is determined as follows:

$$YP = gain2 \times (SP2 - PV2)$$

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

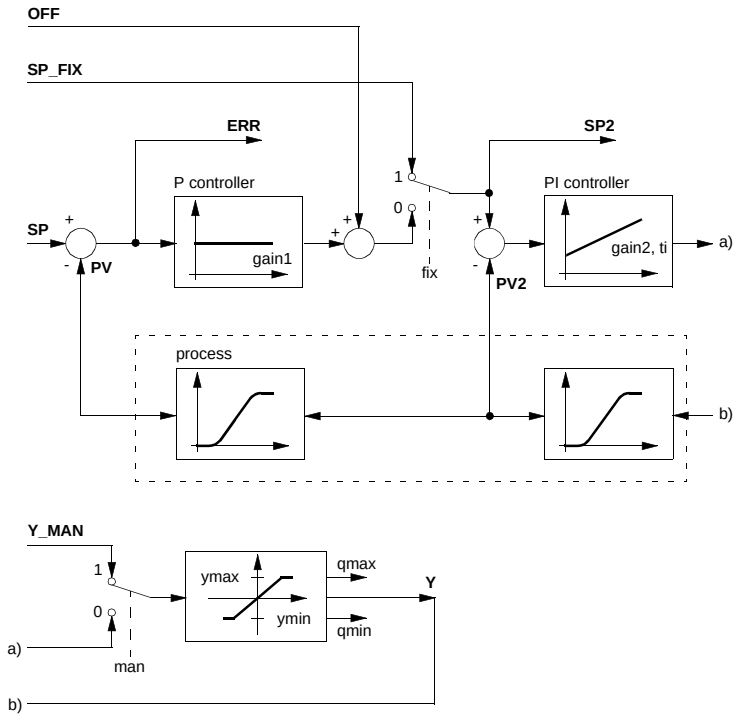
Data structure description

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

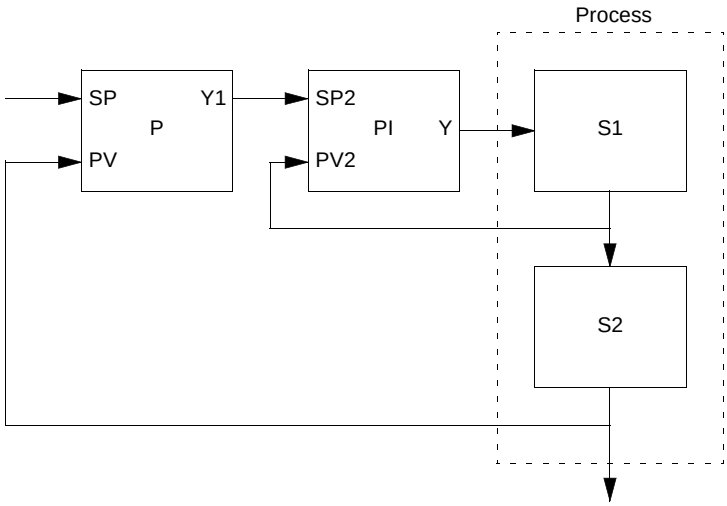
There follows now the structure diagram of the PPI block:



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. 326*.
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. 248*).

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

42

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.

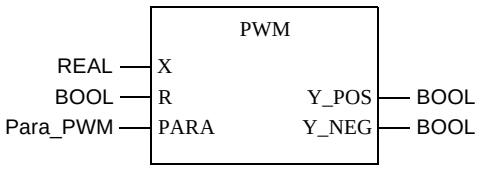
General information about the actuator drive

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.

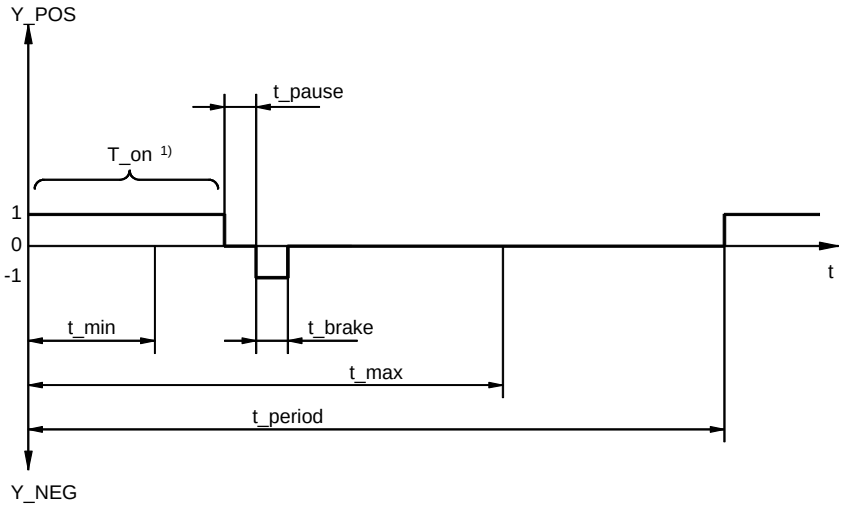
Detailed description

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. 333*" 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:

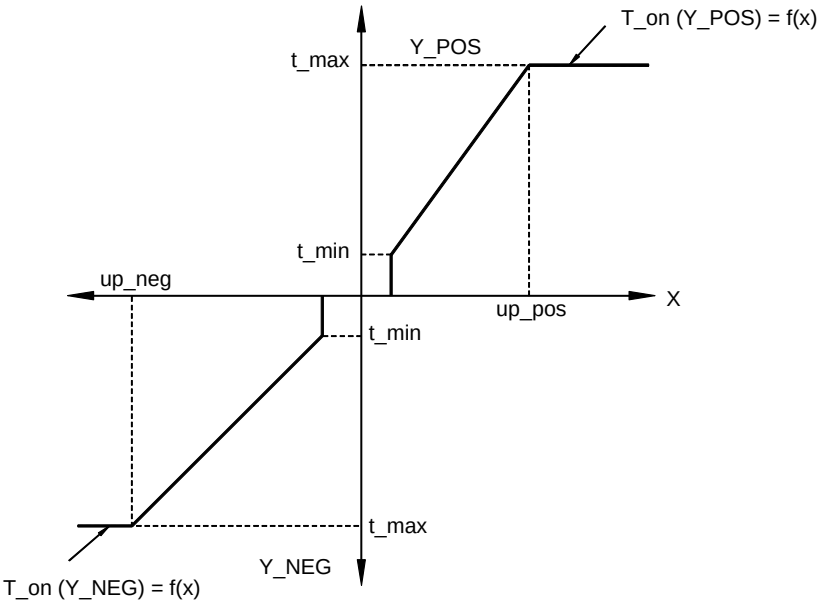


1 Variable turn-on time

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) ≥ 10

Example for the PWM block

Overview

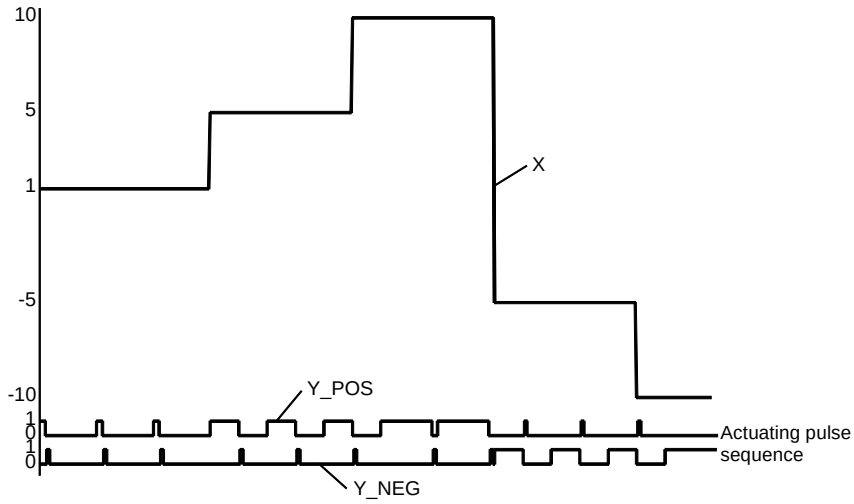
In the examples, the signal sequences on the outputs Y_POS and Y_NEG are shown for various X input signal values. The examples differ with respect to their selected parameter assignments.
The following examples on the PMW function block are to be found in this section

- 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



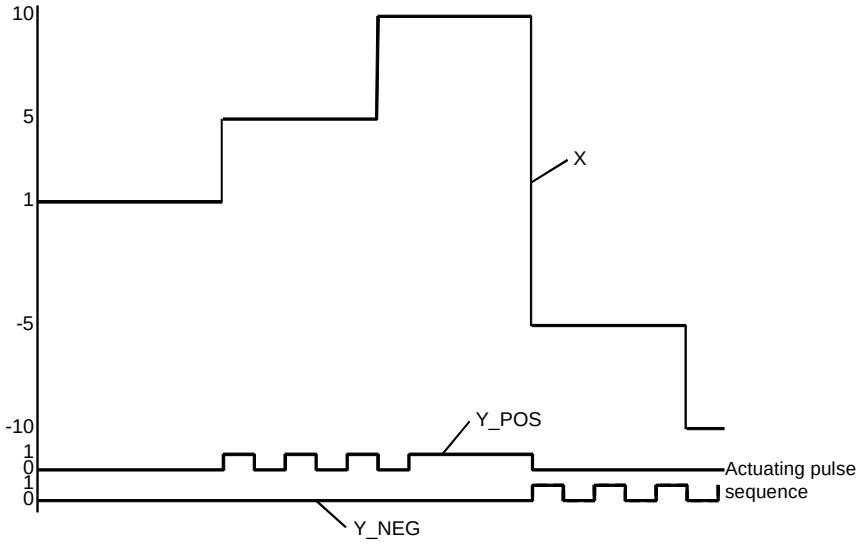
X analog signal

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

43

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.

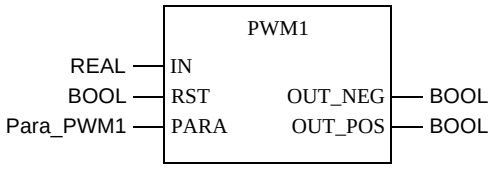
General information about the actuator drive

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	$T\_on = t\_period \times \frac{\lvert IN \rvert}{in\_max}$	$0 \le -IN \le in\_max$

Parametering rules

For correct operation the following rules should be observed:
$t\_min \le t\_period$

Detailed description

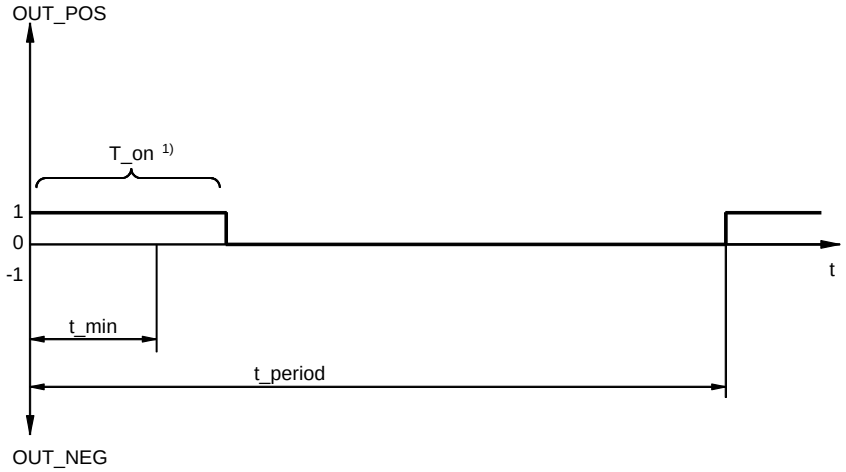
Block mode of operation

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. 343*" 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:

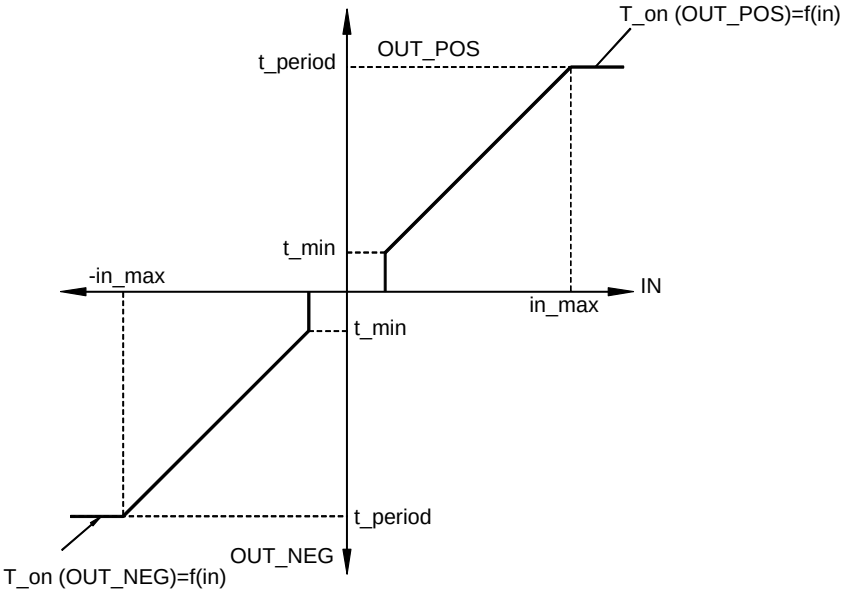


1 Variable turn-on time

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) ≥ 10

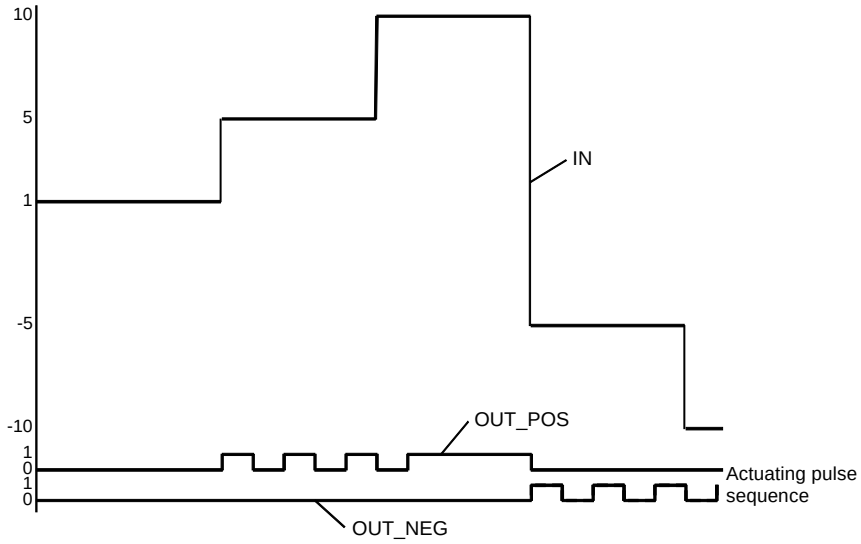
Example of the PWM1 block

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

44

Overview

At a glance

This chapter describes the QDTIME block.

What's in this chapter?

This chapter contains the following topics:

Brief description

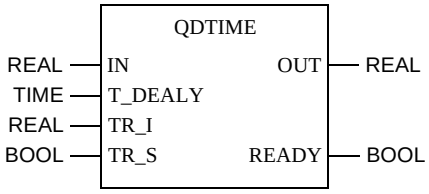
Function description

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

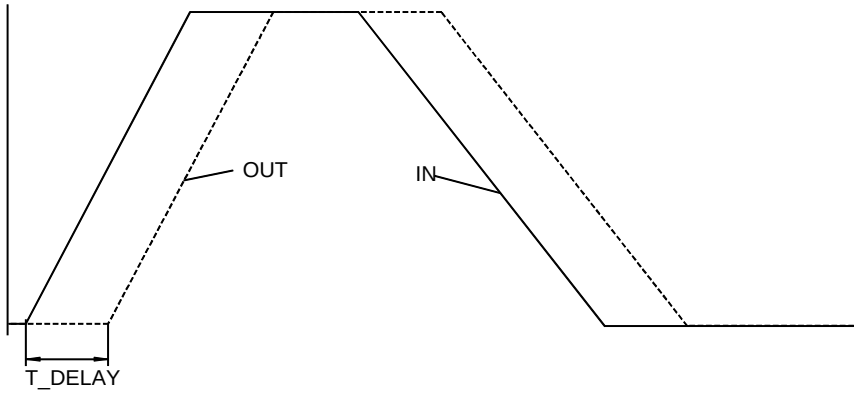
The following values are accepted:
Cycle time = 100 ms
T_DELAY = 10 s
tin = T_DELAY / 128 = 78 ms
As tin (reading time) is shorter than the cycle time, every IN value is accepted in the buffer. On the fourth performance of the function block (after 400 ms) the IN value will be saved twice rather than 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)

45

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.

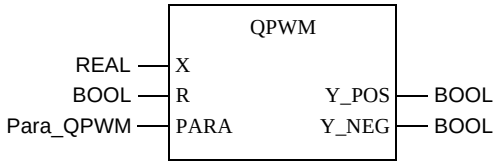
General information about the actuator drive

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	$T\_on = t\_period \times \frac{X}{x\_max}$	$0 \le X \le x\_max$
Y_NEG	$T\_on = t\_period \times \frac{\lvert X \rvert}{x\_max}$	$0 \le -X \le x\_max$

Parametering rules

For correct operation the following rules should be observed:
$t\_min \le t\_period$

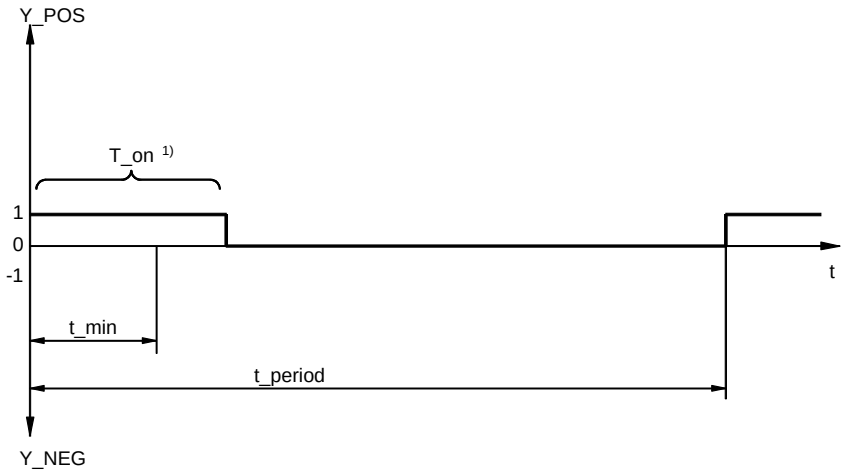
Detailed description

Block mode of operation

The period 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. 353*" 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:

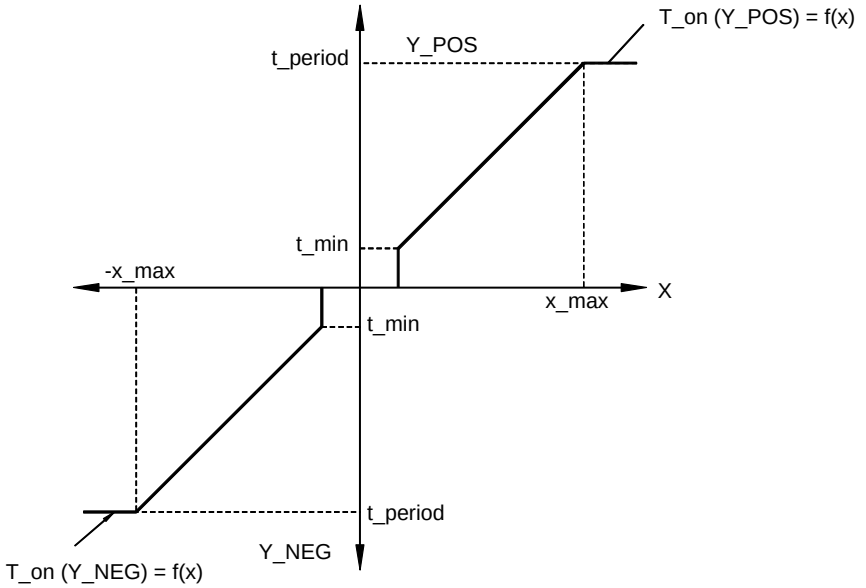


1 Variable turn-on time

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) ≥ 10

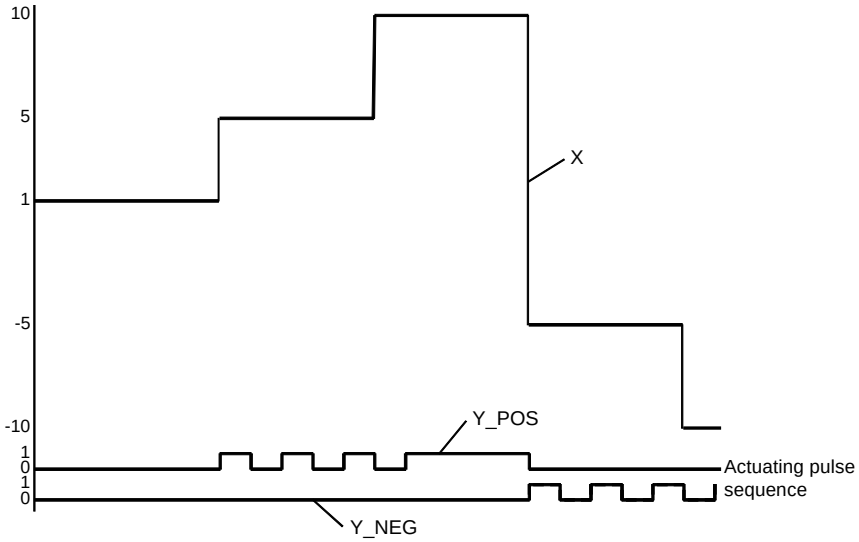
Example for the QPWM block

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

46

Overview

At a glance

This chapter describes the RAMP block.

What's in this chapter?

This chapter contains the following topics:

Brief description

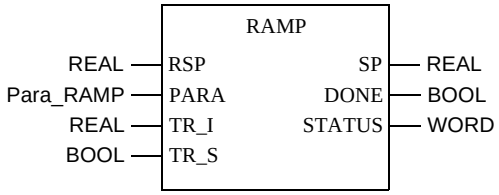
Function description

Der 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

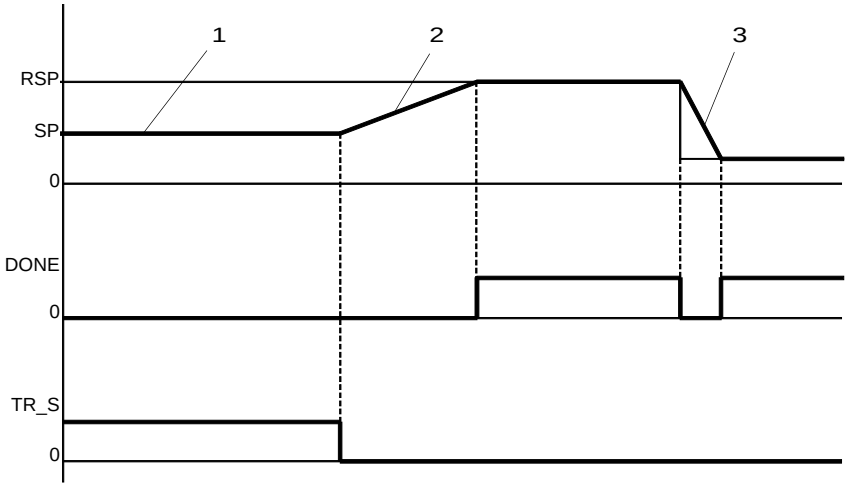
The tracking operation (TR_S = 1) allows for an initial value to be assigned to the SP output. They are as follows:

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

47

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

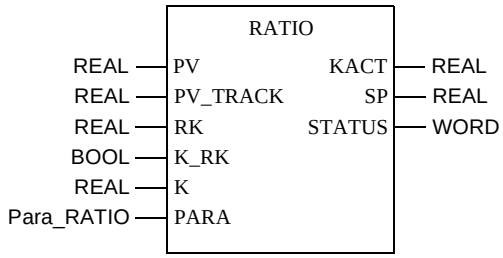
Calculation of the control setpoint

$$SP = K \times PV\_TRACK + bias$$

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

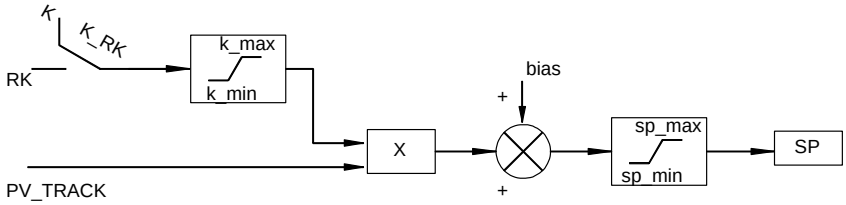
Data structure description

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

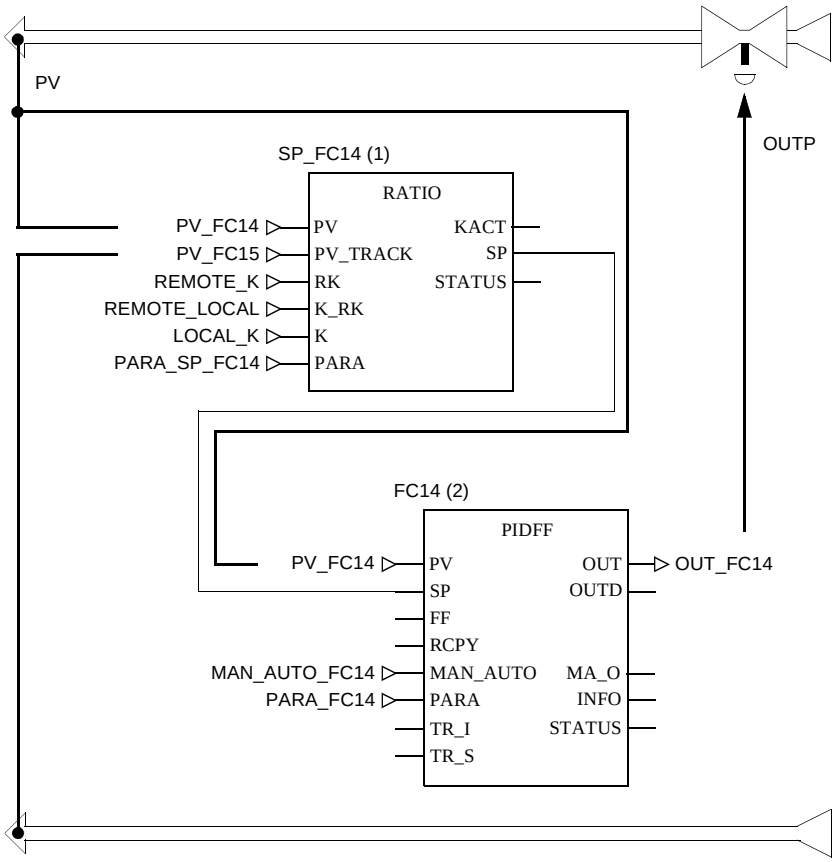
Structure diagram

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

48

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

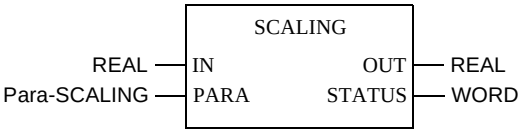
The function block carries out the following calculation:

$$OUT = (IN - in\_min) \times \frac{(out\_max - out\_min)}{(in\_max - in\_min)} + out\_min$$

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

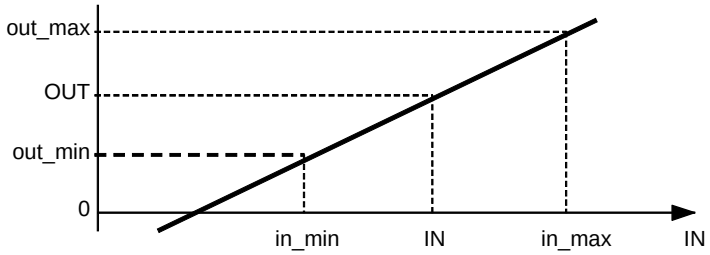
Data structure description

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

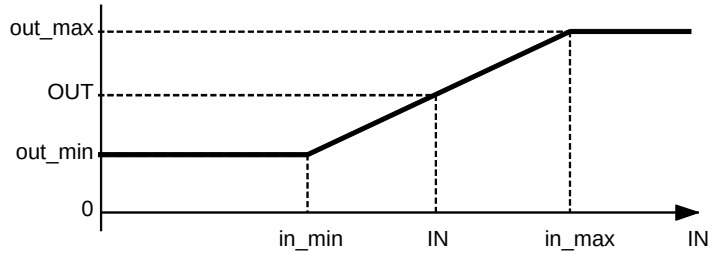
Without output limiting (clip = 0)

If the clip parameter is set to 0, then the scaling is independent of the value of the IN input.



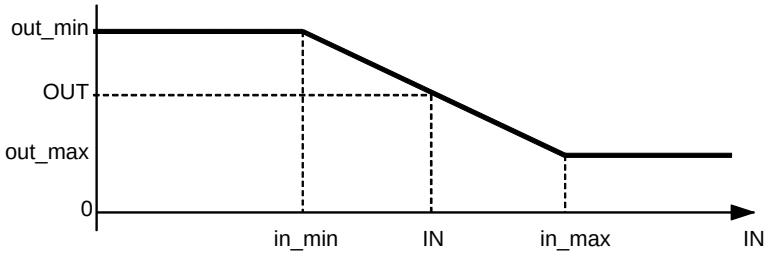
With output limiting (clip = 1)

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

49

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

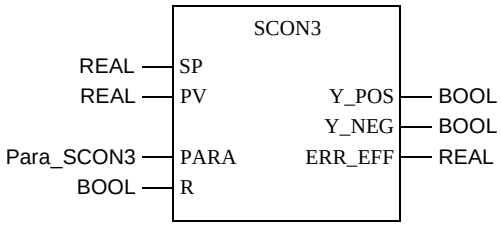
The function block SCON3 contains the following properties:

- 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

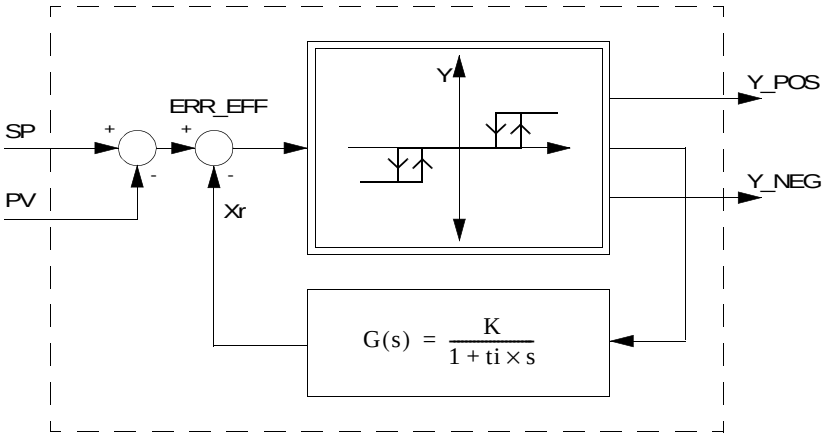
Data structure description

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:

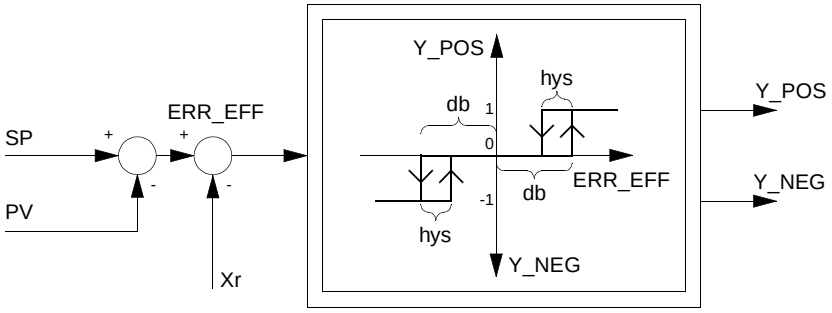
If…	Then…
Y = 1	Y_POS = 1 Y_NEG = 0
Y = 0	Y_POS = 0 Y_NEG = 0
Y = -1	Y_POS = 0 Y_NEG = 1

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. 374*. 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

50

Overview

At a glance

This chapter describes the SERVO block.

What's in this chapter?

This chapter contains the following topics:

Brief description

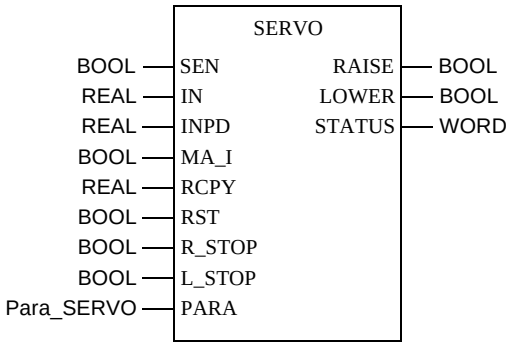
Function description

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. 380*
- *Without positional feedback (en_rcpy = 0), p. 380*
- *Actuator opening time (t_motor), p. 380*
- *Minimum impulse length (t_mini), p. 380*
- *Sweep / parameter SEN, p. 381*
- *Recording the end position, p. 381*

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. 382*").

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. 383*").

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. 383*
- *Example of operating mode automatic without positional feedback in manual mode, p. 387*

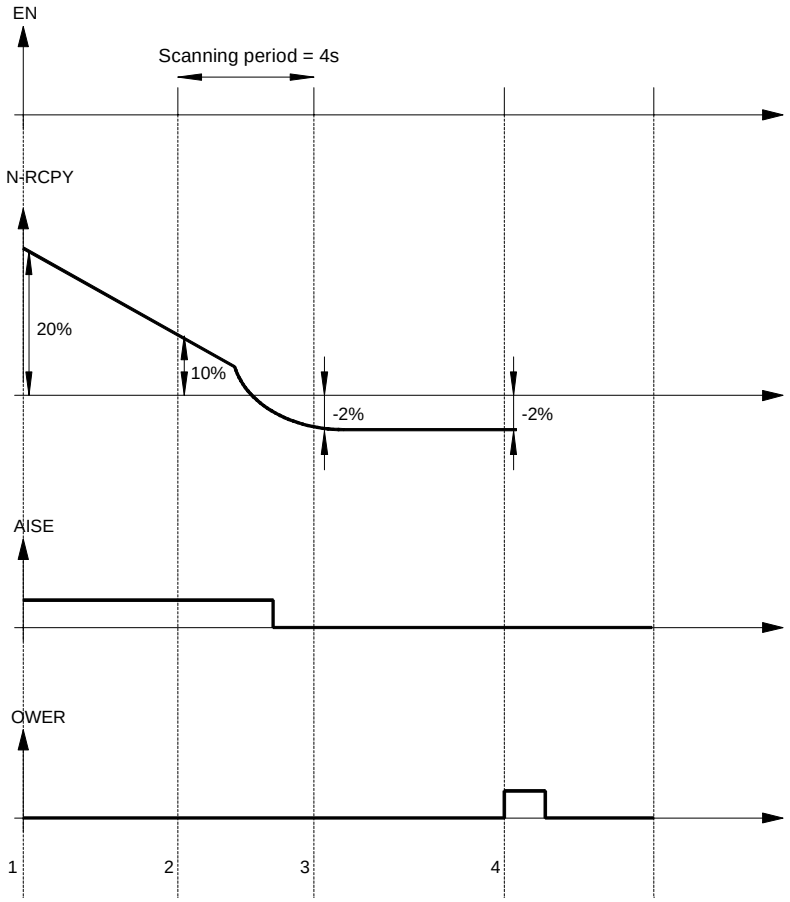
Automatic mode with positional feedback

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



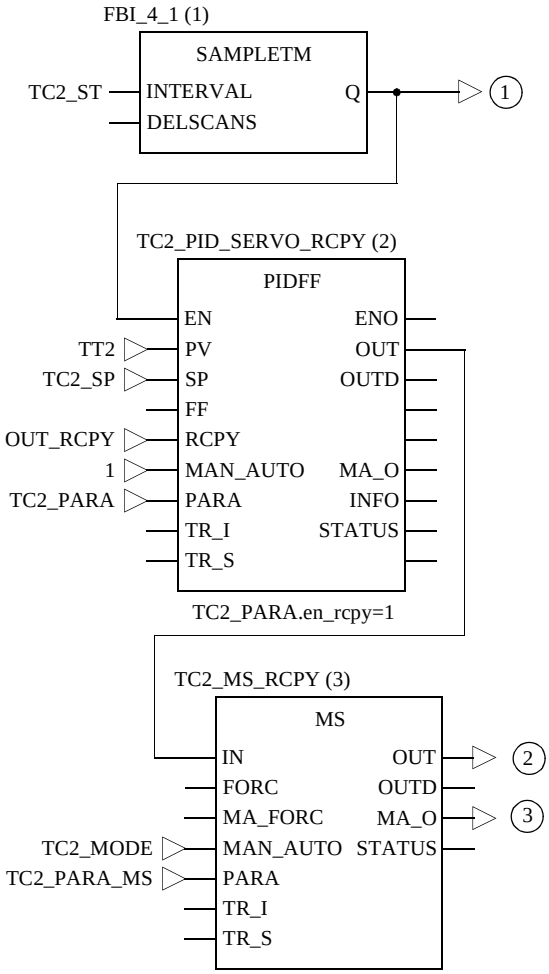
Explanation of the timings

Explanation of the marked positions:

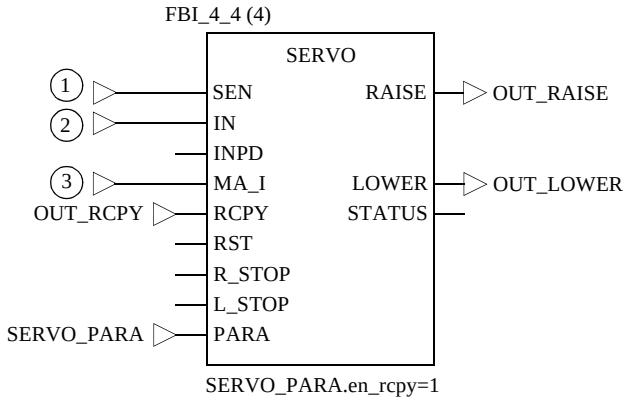
Position No.	Explanation
1	The variance IN-RCPY is 20%: a pulse of length 5 s (=20% of 25 s) was generated at the RAISE output.
2	The variance is still only 10%, a pulse of 2.5s (= 10% of 25 s) was generated at the RAISE output; the second left over from the previous pulse is not taken into account.
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-{}-

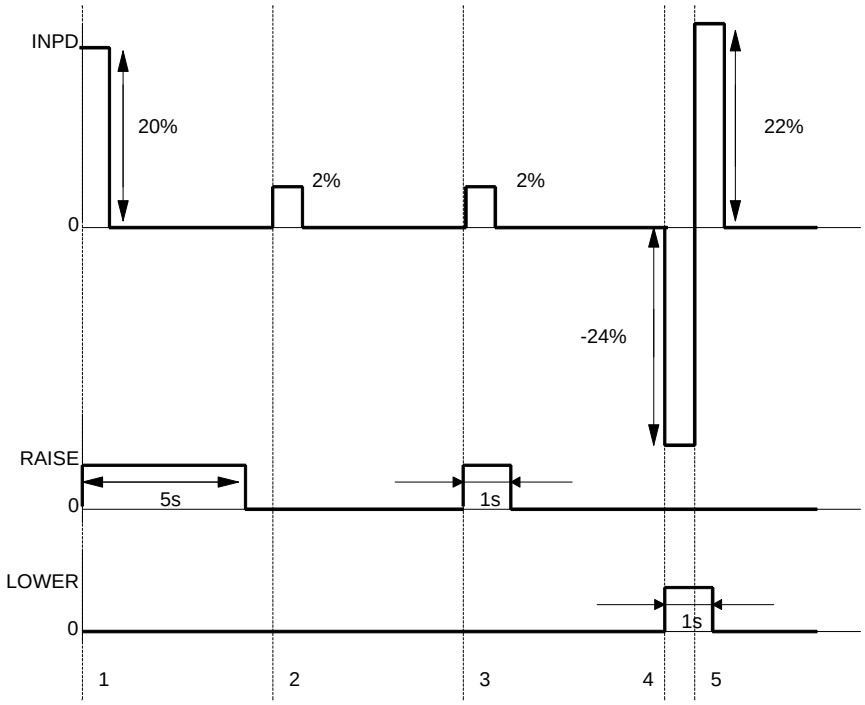
Example of operating mode automatic without positional feedback in manual mode

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



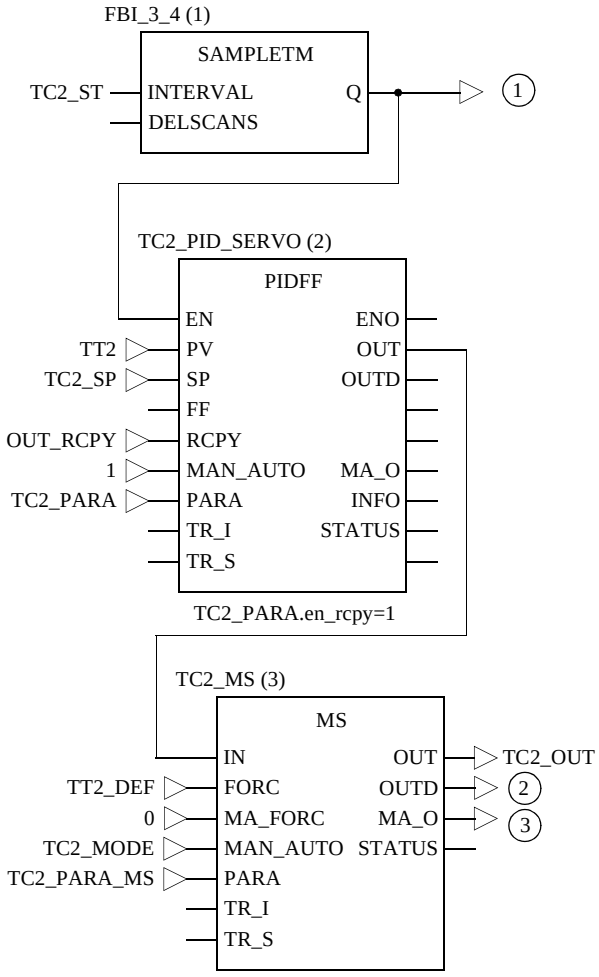
Explanation of the timings

Explanation of the marked positions:

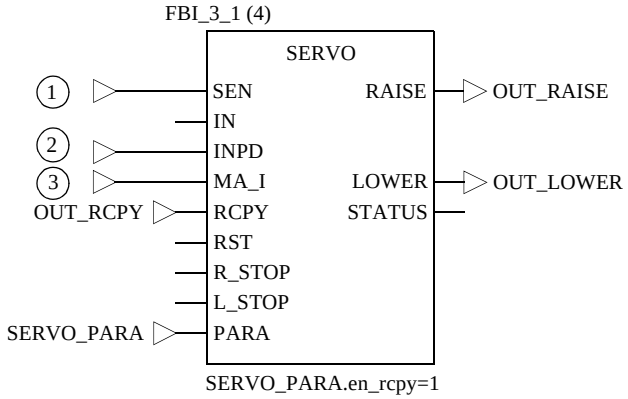
Position No.	Explanation
1	The modification of the PID control output is +20%, in this case the pulse affects the RAISE output and lasts 5 s (= 20% of 25 s).
2	The modification of the PID controller is +2% which corresponds to a pulse duration of 0.5 s. The pulse is less than t_mini (=1 s.) so it does 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

51

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:

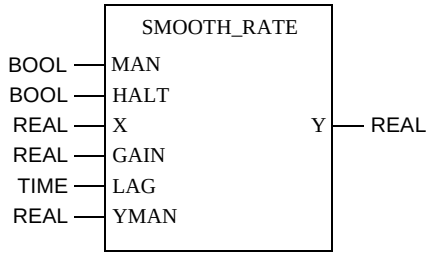
- Manual
- Halt
- Automatic

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.

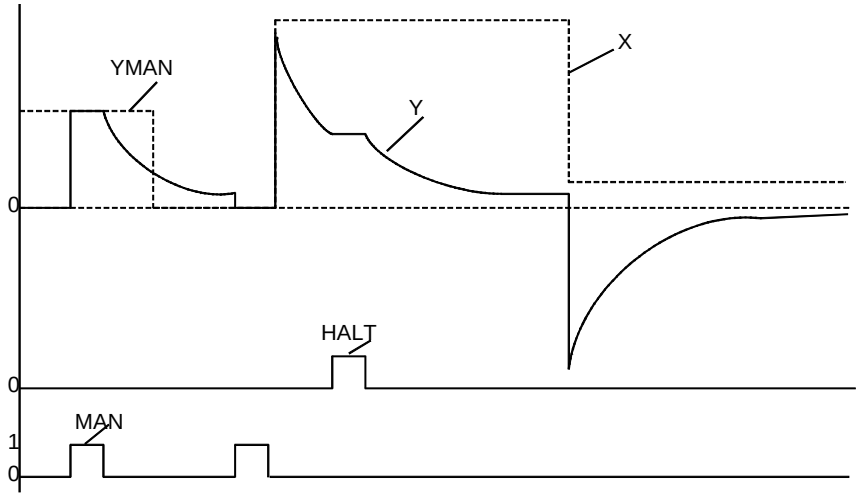
Operating mode

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. The output remains at this value, but can still be overwritten by the user.

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

52

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

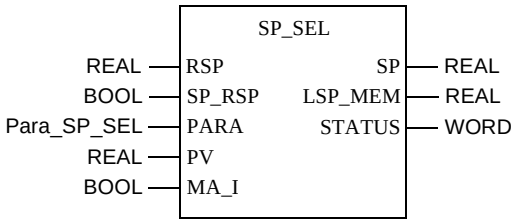
The function block SP_SEL has the following properties:

- 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. 401*").

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.

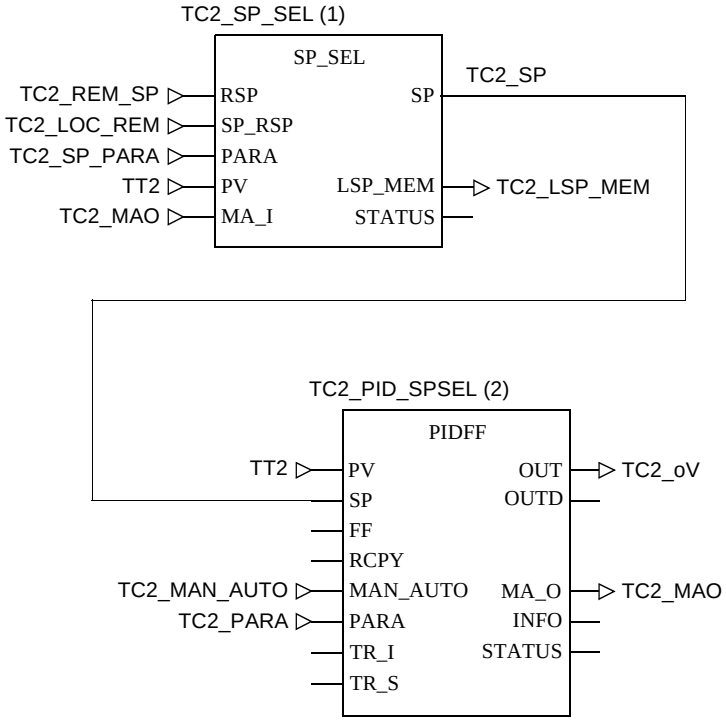
Function of the output LSP_MEM

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

53

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

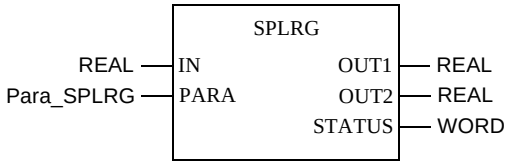
The function block SPLRG has the following properties:

- 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

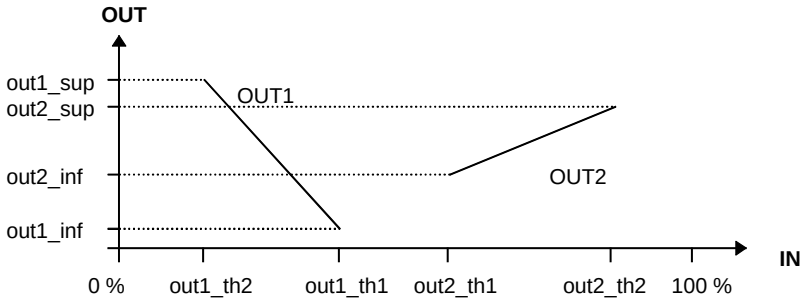
Parametering the function block consists of defining the properties of each actuator, i.e. in the kind of gradient modification of both control outputs in relation to the input IN.
The following points should be defined for the output OUT1:

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.

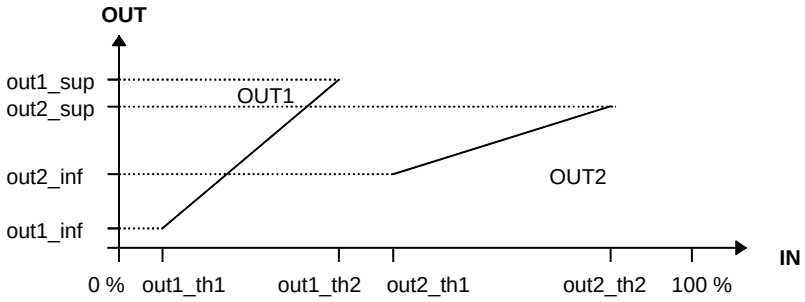
Three step step-control

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

54

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

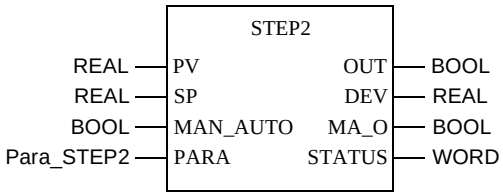
The control block has the following properties:

- 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" : Manual 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: Manual, 1: Automatic)
STATUS	WORD	Status word

Parameter description Para_STEP2

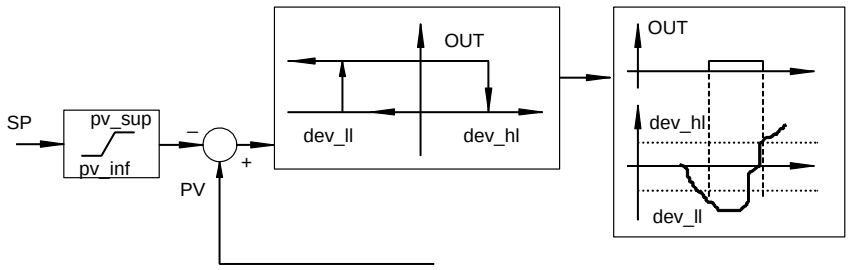
Data structure description

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

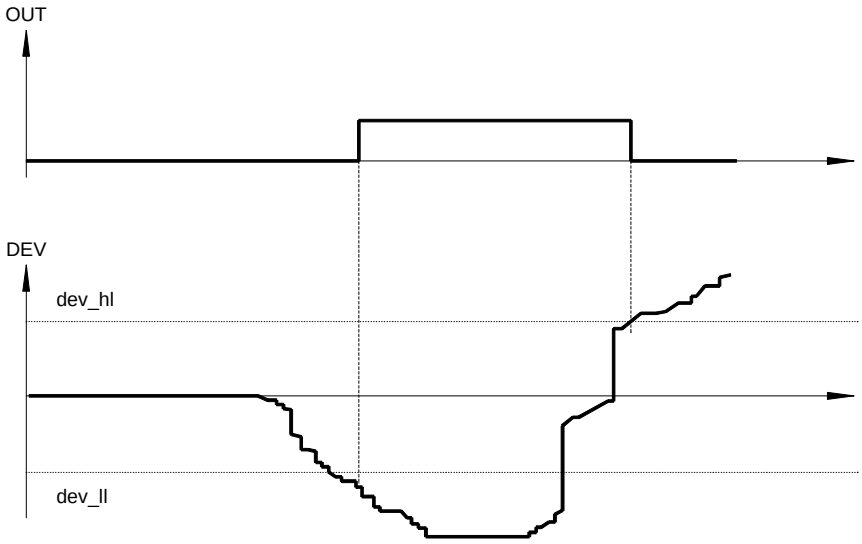
Structure diagram

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.
Manual	0	The controller does not set the output, but the operator can modify the value of the variables connected to the output OUT.

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

55

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

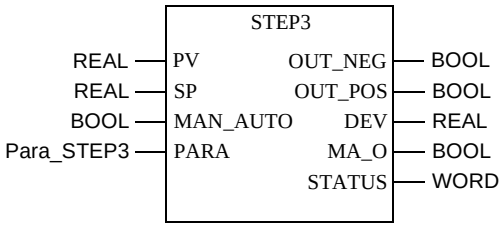
The control block has the following properties:

- 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" : Manual 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: Manual, 1: Automatic)
STATUS	WORD	Status word

Parameter description Para_STEP3

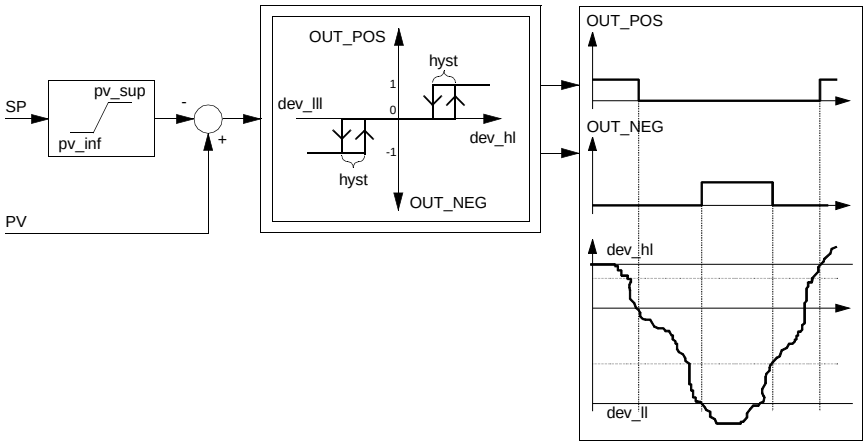
Data structure description

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

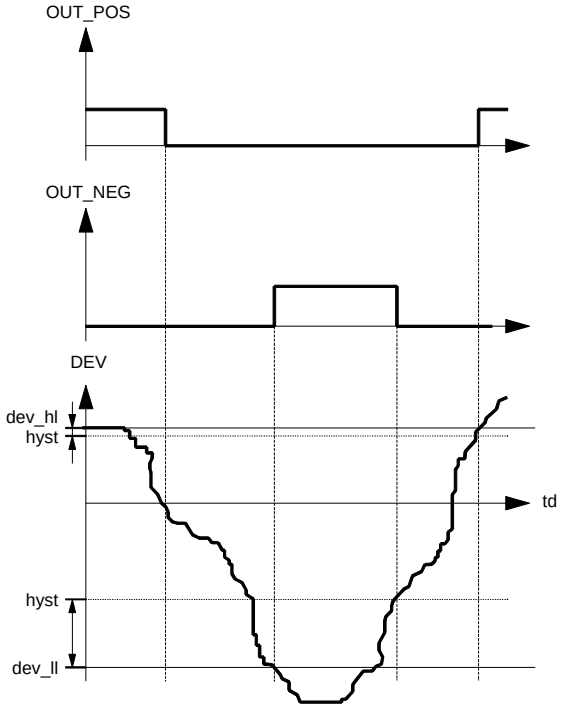
Structure diagram

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.
Manual	0	The configured outputs are not set by the control block; the operator can modify the value of the variables attached to the OUT_NEG and OUT_POS.

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

56

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

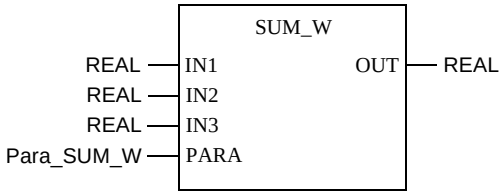
The block SUM_W operates as follows:

$$OUT = k1 \times IN1 + k2 \times IN2 + k3 \times IN3 + c1$$

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

57

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

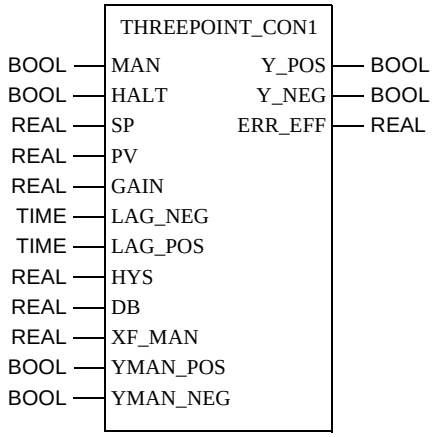
The function block THREEPOINT_CON1 contains the following properties:

- Manual, halt and automatic modes
- two internal feedback paths (1st Degree Delay)

Representation

Symbol

Block representation



Parameter description

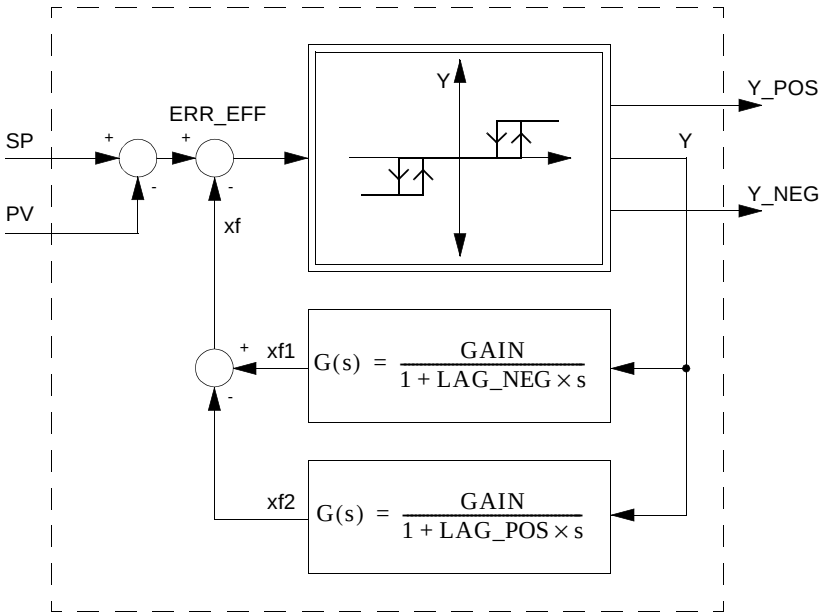
Block parameter description

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:

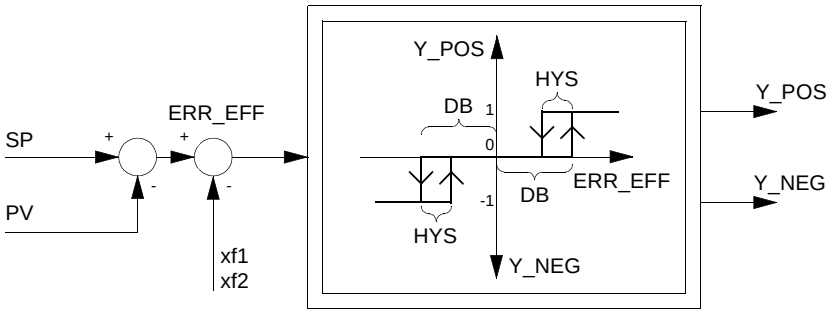


Dependency of outputs Y_POS and Y_NEG on variable Y:

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. 429*". 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

58

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

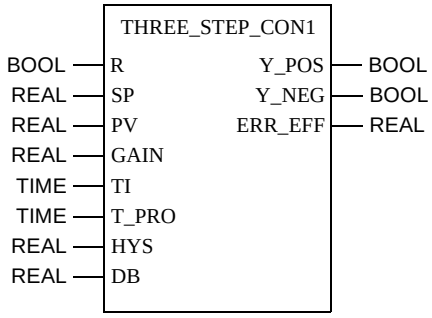
The function block THREE_STEP_CON1 has the following properties:

- Reset and automatic operating modes
- One internal feedback path (1st degree delay)

Representation

Symbol

Block representation



Parameter description

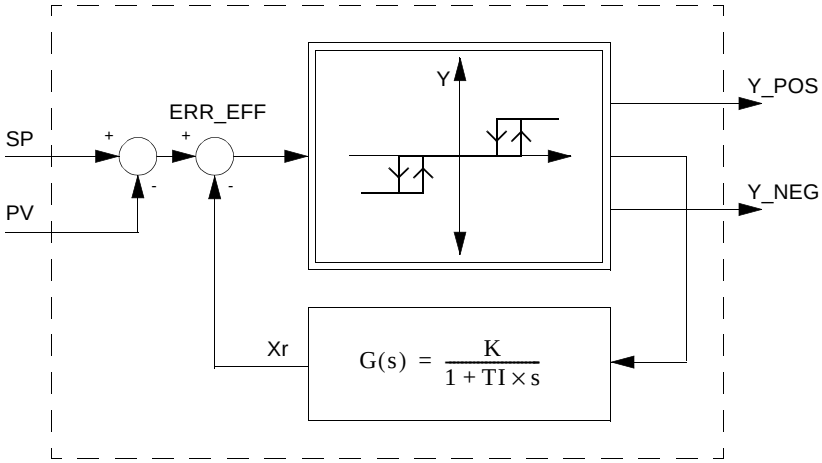
Block parameter description

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:

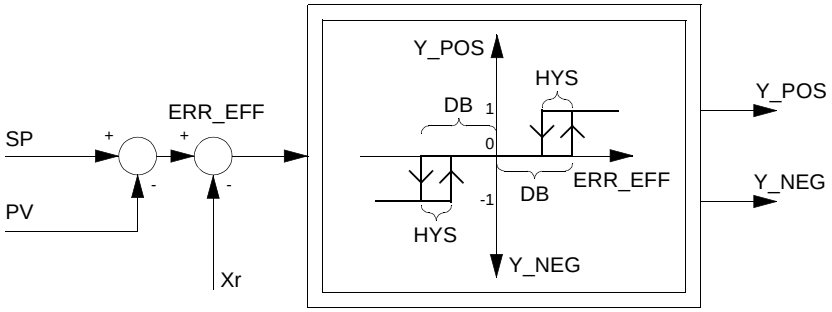
If…	Then…
Y = 1	Y_POS = 1 Y_NEG = 0
Y = 0	Y_POS = 0 Y_NEG = 0
Y = -1	Y_POS = 0 Y_NEG = 1

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. 437*". 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

59

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 an adjustable limit is reached (typically a volume).
EN and ENO can be configured as additional parameters.

Properties

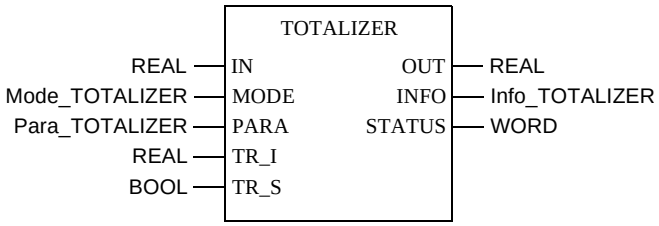
The function block has the following properties

- The integration can be temporarily paused and newly installed
- Equipment that can also consider very small input values
- Division whereby the low limit of the values of IN will no longer be considered
- Use in the mode "Reverse of the integral summation": the output OUT decreases from threshold value to zero (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

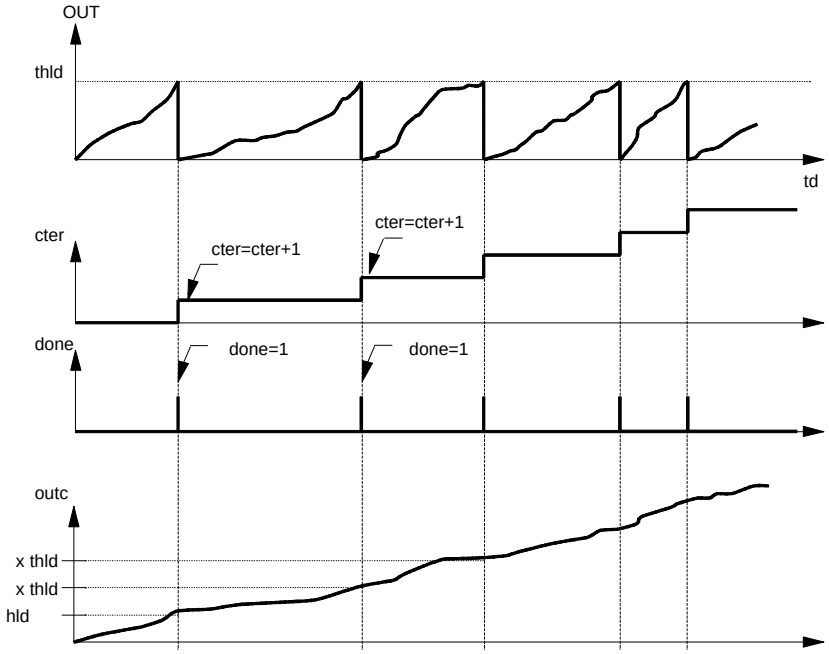
Setting the integral threshold thld

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

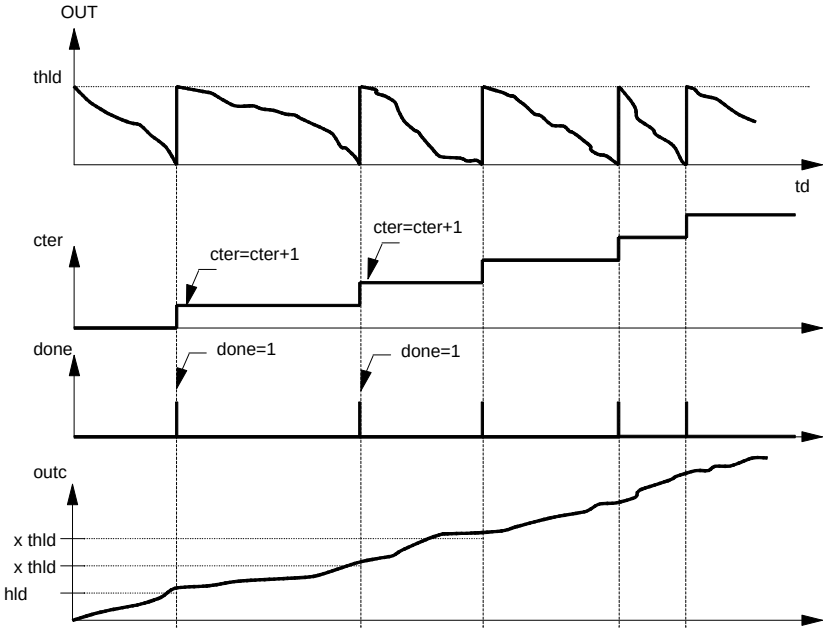
There are 3 individual operating modes for the TOTALIZER function block: Tracking, Reset and Halt:

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

60

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

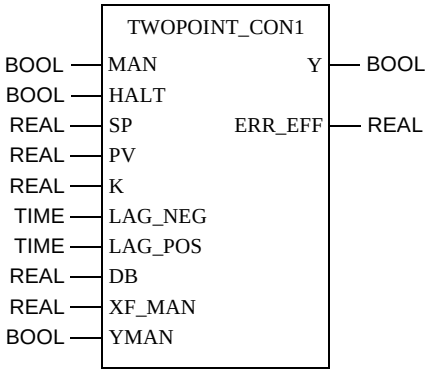
The function block TWOPOINT_CON1 has the following properties:

- Manual, halt and automatic modes
- Two internal feedback paths (1st Degree Delay)

Representation

Symbol

Block representation



Parameter description

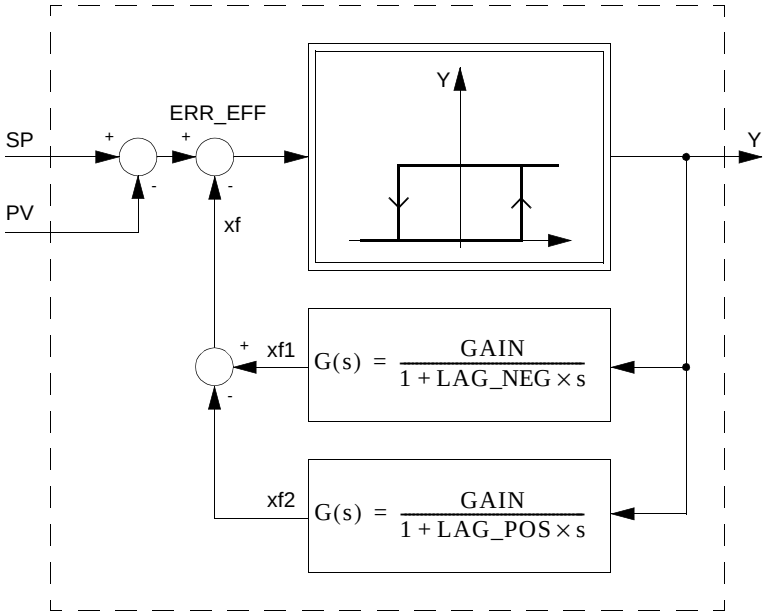
Block parameter description

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

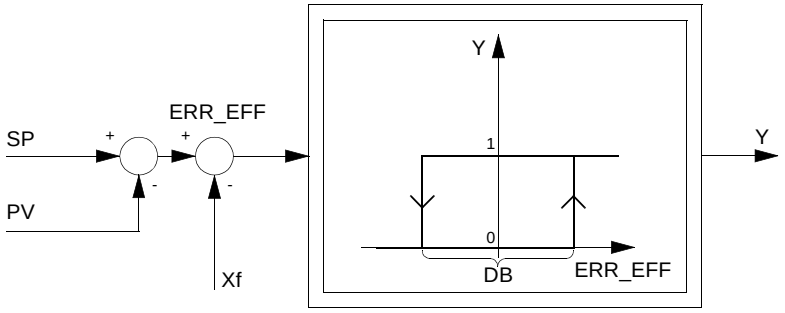
Structure of the controller

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. 451*". 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

61

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

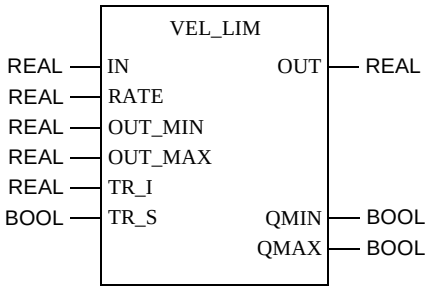
The function block has the following properties:

- 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.

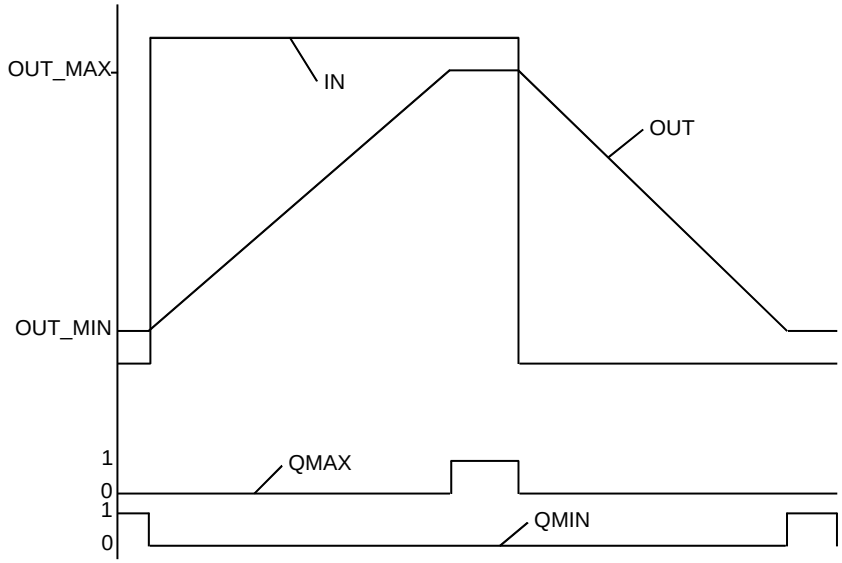
- QMAX = 1 if OUT ≥ OUT_MAX
- QMIN = 1 if OUT ≤ OUT_MIN

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

62

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.

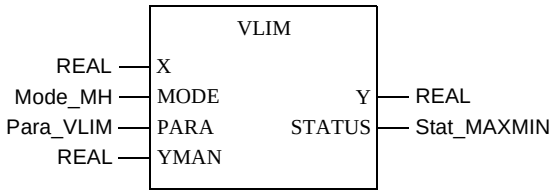
Properties

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

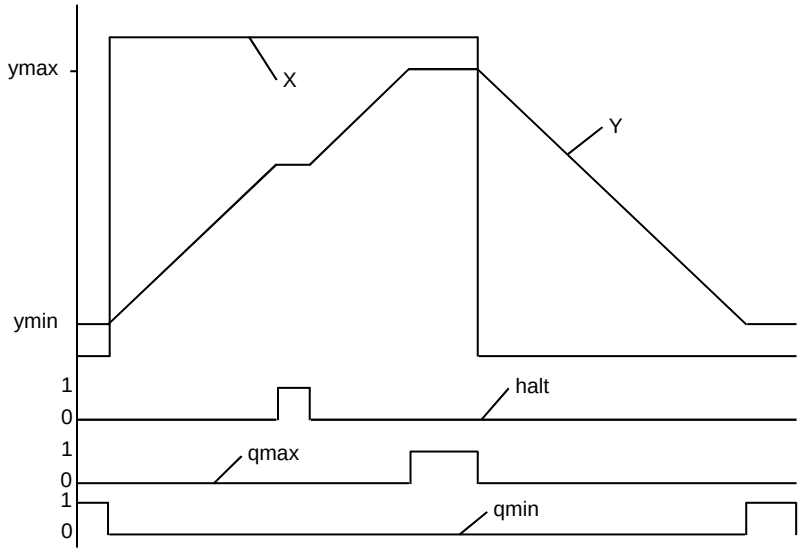
Operating mode

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.

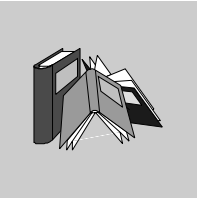
Rum-time error

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 in 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 data types 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 input module, which were specified in the I/O component list, are automatically assigned data types and should therefore only be occupied with Unlocated Variables.

ANY — In the above 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 above version "ANY_BIT" covers the BOOL, BYTE and WORD data types.

ANY_ELEM In the above version "ANY_ELEM" covers the BOOL, BYTE, DINT, INT, REAL, UDINT, UINT, TIME and WORD data types.

ANY_INT In the above version "ANY_INT" covers the DINT, INT, UDINT and UINT data types.

ANY_NUM In the above version "ANY_NUM" covers the DINT, INT, REAL, UDINT and UINT data types.

ANY_REAL In the above version "ANY_REAL" covers the REAL data type.

Application Window The window containing 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 (requiring 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 hexadecimalsystem. 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 dualsystem. 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 into 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.

Bitsequence 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 short on its left side, unchanged, to the horizontal short 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 invoke 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 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.

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 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 body of a derived function block type is designed using FBD language, LD language, ST language, IL language, however, this is only the case in the current version of the programming system. Furthermore, derived functions can not yet be defined in the current version.
A distinction is made between local and global DFBs.

Device Address

The device address is used to uniquely denote a network device in the routing path. The address is set on the device directly, e.g. using the rotary switch on the back of the modules.

DFB Code

The DFB code is the section’s DFB code, which can be executed. The size of the DFB code is mainly dependant upon the number of blocks in the section.

DFB instance data

The DFB instance data is internal data from the derived function block 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, basis 8 literal or base 16 literal. The length of the data element is 32 bits. The value range for variables of this datatype 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 section’s EFB code, which can be executed. 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.

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 invocation can be found in the "Functional block (instance)". In contrast to the invocation 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 invocation 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 = section number (current number)
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 occupied 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 EFB location 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 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 an input word at the address 201 in the State RAM.

Input/output marker bits (0x references)

An input/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.

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 = section number (current number)
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 commentary must, where available, be the last element of the line.

Instruction (LL984)

When programming electrical controls, the user should 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 the "commands" of the ST programming language. Instructions must be concluded 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, basis 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 decimalsystem. 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, through which an operation is carried out.

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 dependant 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 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 using direct values. These values can not be overwritten by the program logic (read only). A distinction is made between generic and standardized literals.
Literals are also used to allocate a constant, a value or a variable an initial value. Entries are made as base 2 literal, base 8 literal, basis 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 addresses 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 value 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.

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 carried out.
Output parameter (outputs):	A parameter, through which the result(s) of the evaluation of a FFB is/are returned.
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.

Print-out Expressions consist of operators and operands.

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 permissable 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 numbers, 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 = Coils
1x range = Discrete inputs
3x range = Input registers
4x range = Output registers
6x range = Register in the extended memory

Note: The x, which follows each 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.

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 of 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 with figures or timing errors with 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 mode of functioning 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, it is however 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 figure 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 figure 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 situated in the state RAM.

Status Bits For every device with global inputs or specific inp./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 occurs, regarding its inputs and outputs of 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 = section number (current number)
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 providing of I/O-map SY/MAX-I/O modules for remote contolling 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 labelling 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 TIME data type is 1 ms.

Time literals Permissable 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 device. The token passes round the devices in a rotating (increasing) address sequence. All devices 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, basis 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, basis 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

D

Q

R

S

T

V