SIEMENS

SIMATIC

Modular PID Control

Manual

Edition 11/2003

A5E00275589-01

Safety Guidelines

This manual contains notices intended to ensure personal safety, as well as to protect the products and connected equipment against damage. These notices are highlighted by the symbols shown below and graded according to severity by the following texts:

Danger

indicates that death, severe personal injury or substantial property damage will result if proper precautions are not taken.

Warning

indicates that death, severe personal injury or substantial property damage can result if proper precautions are not taken.

Caution

indicates that minor personal injury can result if proper precautions are not taken.

Caution

indicates that property damage can result if proper precautions are not taken.

Notice

draws your attention to particularly important information on the product, handling the product, or to a particular part of the documentation.

Qualified Personnel

Only **qualified personnel** should be allowed to install and work on this equipment. Qualified persons are defined as persons who are authorized to commission, to ground and to tag circuits, equipment, and systems in accordance with established safety practices and standards.

Correct Usage

Note the following:

Warning

This device and its components may only be used for the applications described in the catalog or the technical description, and only in connection with devices or components from other manufacturers which have been approved or recommended by Siemens.

This product can only function correctly and safely if it is transported, stored, set up, and installed correctly, and operated and maintained as recommended.

Trademarks

SIMATIC®, SIMATIC HMI® and SIMATIC NET® are registered trademarks of SIEMENS AG.

Third parties using for their own purposes any other names in this document which refer to trademarks might infringe upon the rights of the trademark owners.

Siemens AG
Bereich Automation and Drives
Geschaeftsgebiet Industrial Automation Systems
Postfach 4848, D- 90327 Nuernberg

Preface

Purpose of the Manual

This manual will help you when selecting, configuring, and assigning parameters to a controller block for your control task.

The manual introduces you to the functions of the controller block and explains how to use the Startup and Configuration tool.

Required Basic Knowledge

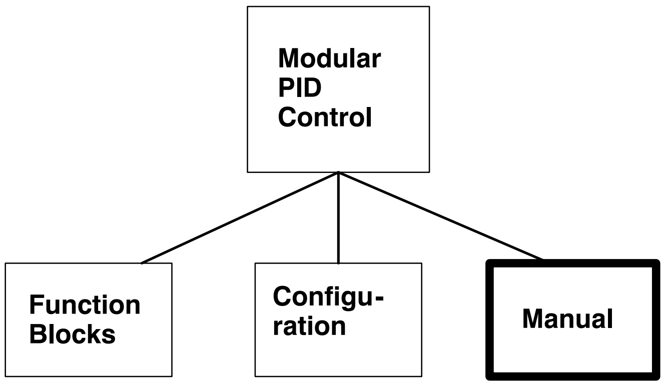
To understand this manual, you should be familiar with automation and process control engineering.

In addition, you should know how to use computers or devices with similar functions (e.g programming devices) under Windows operating systems. Since modular PID Control is based on the STEP 7 software, you should also know how to operate it. This is provided in the manual “Programming with STEP 7 V5.1”.

Where is this Manual valid?

This manual is valid for the software packages Modular PID Control V5.0 and Modular PID Control Tool V5.0.

Place of this Documentation in thr Information Environment



The Modular PID Control package includes three separate products:

- The “Modular PID Control FBs” product contains function blocks and examples.
- The “Modular PID Control FBs” product mainly contains tools for configuring controller blocks.

  The product will subsequently be referred to as “configuration tool”.

Audience

This manual is intended for the following readers:

- S7 programmers
- Programmers of control systems
- Operators
- Service personnel

Conventions in the Text

To make it easier for you to find information in the manual, certain conventions have been used:

- First glance through the titles in the left margin to get an idea of the content of a section.
- Sections dealing with a specific topic either answer a question about the functions of the tool or provide information about necessary or recommended courses of action.
- References to further information dealing with a topic are indicated by (*see Chapter* or *Section* x.y). References to other documentation are indicated by a number in slashes /.../. Based on these numbers, you can refer to the References in the Appendix if you require the full title of the documentation.
- You will find a glossary with important controller terms in the manual *"Standard PID Control"*

Further Support

If you have any technical questions, please get in touch with your Siemens representative or agent responsible.

You will find your contact person at:

`http://www.siemens.com/automation/partner`

Training Centers

Siemens offers a number of training courses to familiarize you with the SIMATIC S7 automation system. Please contact your regional training center or our central training center in D 90327 Nuremberg, Germany for details:

Telephone:	+49 (911) 895-3200.
Internet:	`http://www.sitrain.com`

A&D Technical Support

Worldwide, available 24 hours a day:

Worldwide (Nuernberg) **Technical Support**		
24 hours a day, 365 days a year Phone: +49 (180) 5050-222 Fax: +49 (180) 5050-223 E-Mail: adsupport@ siemens.com GMT: +1:00		
Europe / Africa (Nuernberg) **Authorization**	**United States** (Johnson City) **Technical Support and Authorization**	**Asia / Australia** (Beijing) **Technical Support and Authorization**
Local time: Mon.-Fri. 8:00 AM to 5:00 PM Phone: +49 (180) 5050–222 Fax: +49 (180) 5050-223 E-Mail: adsupport@ siemens.com GMT: +1:00	Local time: Mon.-Fri. 8:00 AM to 500 PM Phone: +1 (423) 262 2522 Fax: +1 (423) 262 2289 E-Mail: simatic.hotline@ sea.siemens.com GMT: –5:00	Local time: Mon.-Fri. 8:00 AM to 5:00 PM Phone: +86 10 64 75 75 75 Fax: +86 10 64 74 74 74 E-Mail: adsupport.asia@ siemens.com GMT: +8:00
The languages of the SIMATIC Hotlines and the authorization hotline are generally German and English.		

Service & Support on the Internet

In addition to our documentation, we offer our Know-how online on the internet at:

`http://www.siemens.com/automation/service&support`

where you will find the following:

- The newsletter, which constantly provides you with up–to–date information on your products.
- The right documents via our Search function in Service & Support.
- A forum, where users and experts from all over the world exchange their experiences.
- Your local representative for Automation & Drives.
- Information on field service, repairs, spare parts and more under “Services”.

Contents

Product Overview – Modular PID Control 1

1.1 The Product Modular PID Control

Concept of Modular PID Control

The “Modular PID Control” software product consists of a set of **function blocks** (FBs) and **functions** (FCs) containing the algorithms for creating controller functions. This is therefore purely a software controller in which you can implement the controller functions by interconnecting the blocks.

The block library is supplemented by a number of ready-to-use controller structures (single-loop fixed setpoint controller, ratio controller etc.) in the form of examples. You can copy and adapt these examples to suit your own control task.

When operating a large number of control loops, it is usually the case that some loops must be processed more often than others although each loop itself must be processed at equidistant intervals. For this situation, there is a **loop scheduler** (LP_SCHED) available with which extensive control systems can be configured clearly and simply. This also ensures that the load on the CPU is spread out.

To help you install and test individual control loops, the package also includes *the configuration tool “Modular PID Control Tool”.* This includes a loop monitor, a curve recorder for manipulating and monitoring process variables, and an algorithm for process identification and optimization of the PID parameters.

Overview of the Basic Functions

In many control tasks, the classic PID controller that influences the process is not the sole important element but great demands are also made on signal processing.

A controller created with the “Modular PID Control” software package therefore consists of a series of subfunctions for which you can select parameter values separately. In addition to the actual controller with the PID algorithm, functions are also available for processing the setpoint and process variable and for adapting the calculated manipulated variable.

1.2 The Components of Modular PID Control

Modular PID Control FB

The “Modular PID Control FB” package consists of a library with function blocks and 12 ready-to-use examples of controllers.

You can install the software on programming devices/PCs with the SETUP program. The online help system provides you with information about subfunctions and individual parameters while you are working.

Modular PID Control Tool

Using the “Startup and Test” tool, you can install, start up and test your controller structure and optimize the PID parameters.

The configuration tool includes a loop monitor, a curve recorder and an algorithm for setting or optimizing the PID controller parameters. *The configuration tool* is described in detail in Chapter 5.

Modular PID Control Manual

For details of the content of this manual, refer to the table of contents.

1.3 Environment and Applications

Hardware Environment

The controllers created with the “Modular PID Control” can be run on the programmable controllers (CPU with floating-point and cyclic interrupts) of the S7-300 and S7-400 family and Win AC.

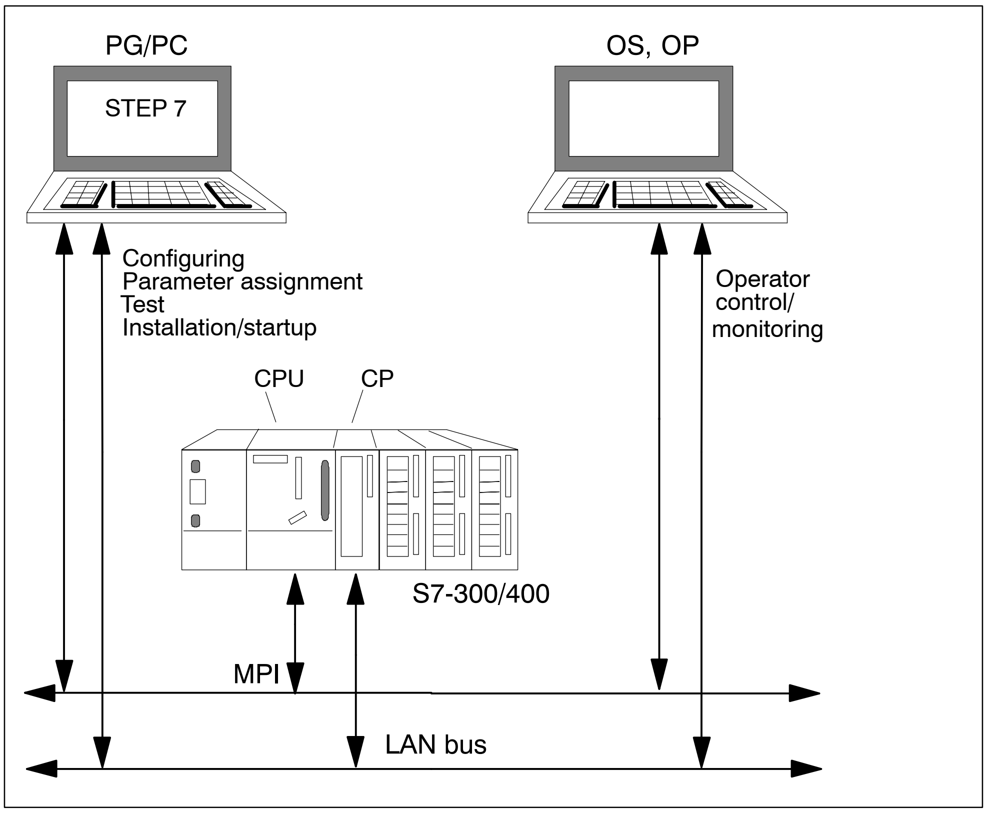


Figure 1-1 Environment of “Modular PID Control”

Software Environment

Modular PID Control is designed for use in the STEP 7 program group.

The configuration software for Modular PID Control can be installed locally on a programming device/PC or in a network on a central network drive.

Range of Functions of Modular PID Control

Both slow processes (temperatures, tank levels) and very fast processes (flow rate, motor speed) can be controlled. The following controller types can be implemented:

- Continuous PID controllers
- PID step controllers for integrating actuators
- Pulse-break controllers

They can be connected to create one of the following controller structures:

- Fixed setpoint controllers
- Cascade controllers
- Ratio controllers
- Blending controllers
- Split range controllers
- Override controllers
- Controllers with feedforward control
- Multiple variable controllers

Description of the Functions 2

2.1 General Information

Conventions Used with Parameter and Block Names in the Block Diagrams

The names of the parameters are a maximum of 8 characters long.

The following conventions were used to name the parameters:

First letter:

Q	general output of the type BOOL (Boolean variable)
SP	setpoint
PV	process variable
LMN	manipulated variable or analog output signal
DISV	disturbance variable

Following letters:

MAN	manual value
INT	internal
EXT	external
_ON	BOOLEAN variable to activate a function

Call Data

Most blocks in the Modular PID Control package require loop-specific call data such as the complete restart bit and sampling time. These values are transferred via the inputs COM_RST and CYCLE.

Notes on the block parameters (input, output and in/out parameters)

- **Default:** these are the default values used when an instance is created.
- **Permitted Values**: the values set for the input parameters should not exceed the permitted range of values. The range is **not** checked when the block is executed. The entry “technical range of values” means a physical variable with a value between approximately $\pm 10^{6}$.

2.1.1 A_DEAD_B: Adaptive Dead Band

Application

If the process variable is affected by noise and the controller is optimally set, the noise will also affect the controller output. Due to the high switching frequency (step controller), this increases wear and tear on the actuator. Suppressing the noise prevents oscillation of the controller output.

Block Diagram

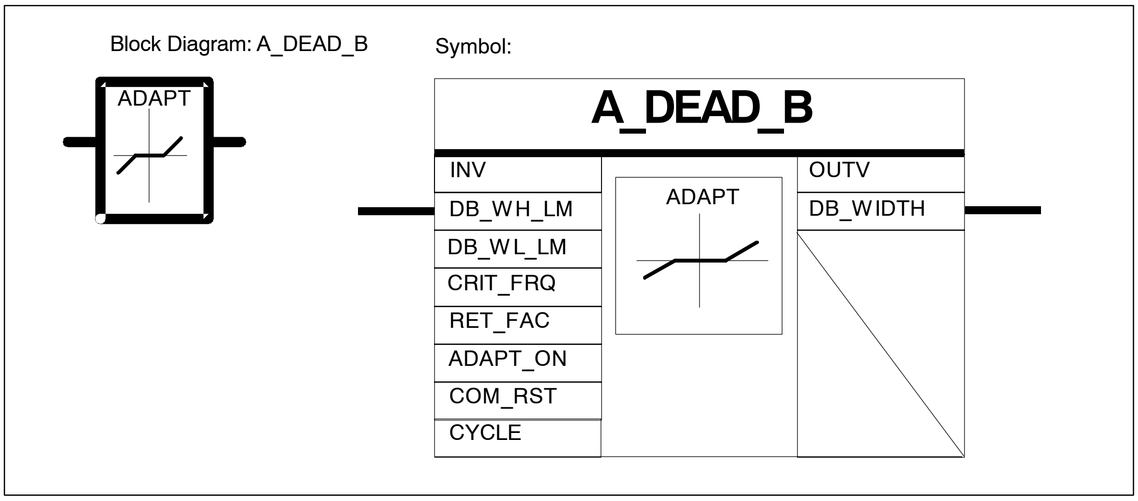


Figure 2-1 A_DEAD_B, Block Diagram and Symbol

Functional Description

This block filters high-frequency disturbance signals out of the error signal. It forms a dead band around the zero point. If the input variable is within this dead band, zero is applied to the output. The width of the dead band is automatically adapted to the amplitude of the noise signal.

The block operates according to the following function:

OUTV = INV + DB_WIDTH	when	INV < –DB_WIDTH
OUTV = 0.0	when	–DB_WIDTH ≤ INV ≤ +DB_WIDTH
OUTV = INV – DB_WIDTH	when	INV > +DB_WIDTH

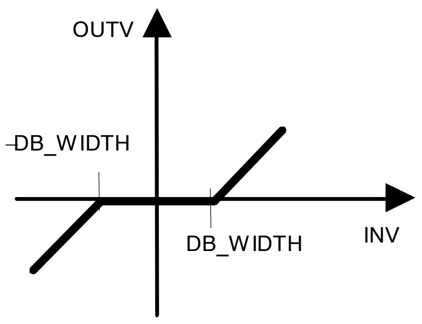


Figure 2-2 OUTV = f(INV)

Adaptation of the Dead Band

To ensure stability, the effective dead width band DB_WIDTH is limited downwards by the selectable input parameter DB_WL_LM. If the input signal INV affected by noise exceeds the currently set dead band width in the negative (1), positive (2), and then negative (3) direction again within the period 1/CRIT_FRQ, the effective band width is increased by the value 0.1. (see also Figure 2-4).This procedure is started whenever the dead band is exceeded in a positive or negative direction. Whenever the dead band is exceeded subsequently (3 –> 4), in the opposite direction within half the period, it is once again increased by 0.1. This procedure is repeated until the dead band width matches the amplitude of the measured noise. To prevent input signals of any magnitude being suppressed, the effective dead band width is limited upwards by the input DB_WH_LM. If, on the other hand, the dead band width is not exceeded within the time RET_FAC*1/CRIT_FRQ, the value is reduced by 0.1.

CRIT_FRQ specifies the critical frequency at which a signal component is detected as noise. It is limited upwards and downwards as follows:

0.01 CRIT_FRQ 1/(3*CYCLE) where CYCLE is the sampling time in seconds.

The RET_FAC parameter specifies the multiple of 1/CRIT_FRQ following which the dead band width is reduced again.

The adaptation logic only operates when the input variable without a noise component is close to zero.

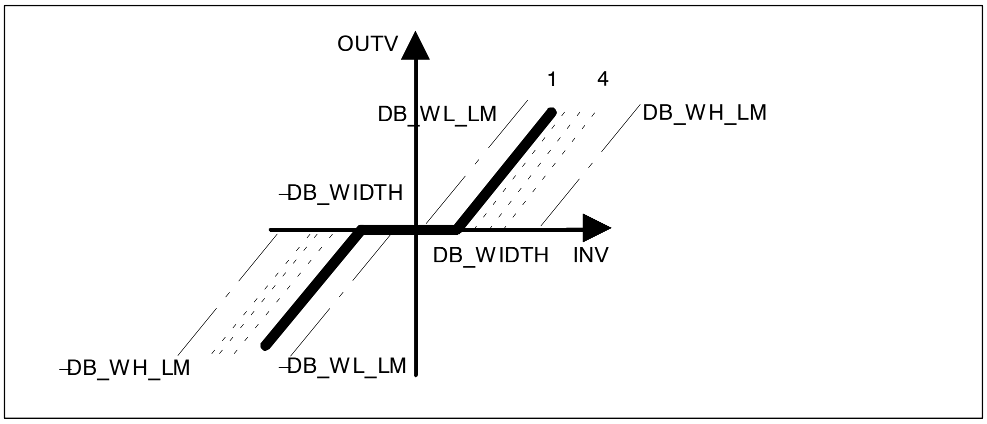


Figure 2-3 Adaptation of the Dead Band

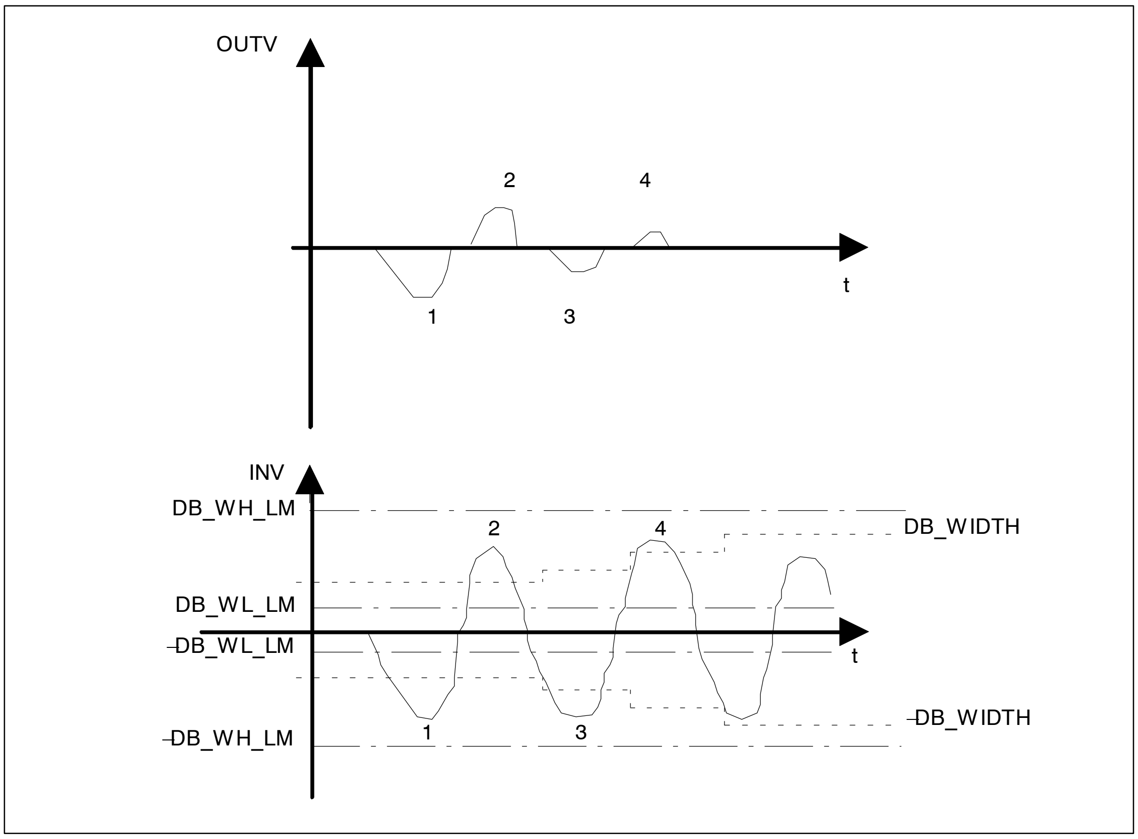


Figure 2-4 Adaptation of the Dead Band

Input Parameters

The following table shows the data type and structure of the input parameters of A_DEAD_B.

Table 2-1 Input Parameters of A_DEAD_B

Data Type	Parameter	Comment	Permitted Values	Default
REAL	INV	input variable	technical range of values	0.0
REAL	DB_WH_LM	dead band width high limit	tech. range > DB_WL_LM	5.0
REAL	DB_WL_LM	dead band width low limit	tech. range < DB_WH_LM	1.0
REAL	CRIT_FRQ	critical frequency	≥ 0.01 and ≤1/(3 ∗ CYCLE)	0.1
INT	RET_FAC	return factor	≥ 1	1
BOOL	ADAPT_ON	adaptive algorithm on		FALSE
BOOL	COM_RST	complete restart		FALSE
TIME	CYCLE	sampling time	≥ 1ms	T#1s

Output Parameters

The following table shows the data type and structure of the output parameters A_DEAD_B.

Table 2-2 Output Parameters of A_DEAD_B

Data Type	Parameter	Comment	Default
REAL	OUTV	output variable	0.0
REAL	DB_WIDTH	effective dead band width	0.0

Complete Restart

During a complete restart, OUTV is set to 0.0 and the effective dead band width is set so that DB_WIDTH = DB_WL_LM.

Normal Operation

The following conditions apply to the adaptation:

- Adaptation Off

  If adaptation is turned off (ADAPT_ON = FALSE), the last DB_WIDTH value is frozen and used as the effective dead band width DB_WIDTH.

- Adaptation On

  If ADAPT_ON = TRUE, an adaptation algorithm can be included that calculates the effective dead band width. This adapts the dead band width to the amplitude of the noise signal overlaying the input variable so that the noise component is suppressed even when its amplitude fluctuates.

  If the block call is acyclic, the adaptation must be turned off (ADAPT_ON = FALSE).

Block-Internal Limits

The values of the input parameters are not restricted in the block; the parameters are not checked.

Example

If the adaptation is turned on due to noise during startup and if a stable dead band width is established after a certain time, the adaptation can be turned off. The dead band width set by the adaptive function is retained until there is a complete restart.

2.1.2 CRP_IN: Change Range Peripheral Input

Application

The block adapts the range of values of the analog I/Os to the internal representation of the modular controller; it can, for example, be called in the process variable branch.

Block Diagram

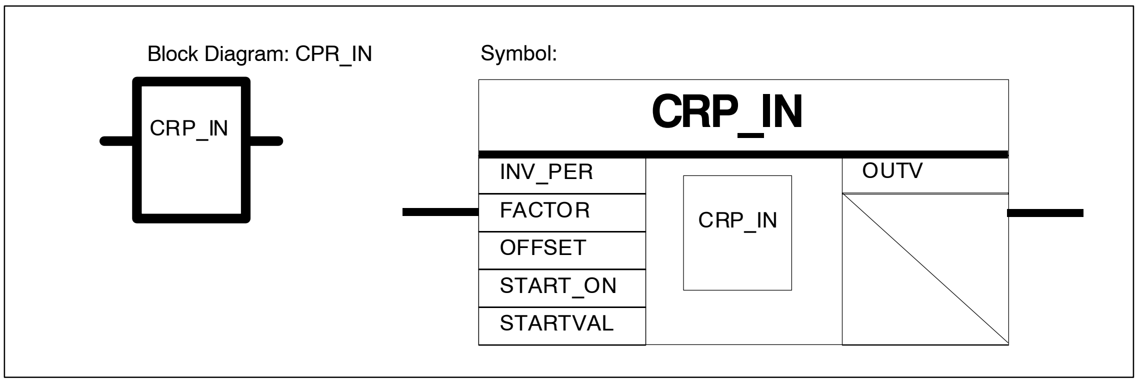


Figure 2-5 CRP_IN, Block Diagram and Symbo

Functional Description

CRP_IN converts an input value in peripheral format to a normalized floating-point value for the modular controller.

Peripheral Value	Output Value in %
32767	118.515
27648	100.000
1	0.003617
0	0.000
–1	–0.003617
–27648	–100.000
–32768	–118.519

The floating-point value can be adapted using a scaling factor and an offset. The output is obtained as follows:

OUTV = INV_PER * 100/27648 * FACTOR + OFFSET

During installation, testing or if problems occur in the periphery, it is possible to change to a startup value. If **START_ON** = TRUE is set, the value in **STARTVAL** is output at the **OUTV** output.

Note

There is no check for positive/negative overflow.

Input Parameters

The following table shows the data type and structure of the input parameters of CRP_IN.

Table 2-3 Input Parameters of CRP_IN

Data Type	Parameter	Comment	Permitted Values	Default
WORD	INV_PER	input variable peripheral	technical range of values	0
REAL	FACTOR	scaling factor		1.0
REAL	OFFSET	offset	technical range of values	0.0
BOOL	START_ON	startup value on		TRUE
REAL	STARTVAL	startup value	technical range of values	0.0

Output Parameters

The following table shows the data type and structure of the output parameters CRP_IN.

Table 2-4 Output Parameters of CRP_IN

Data Type	Parameter	Comment	Default
REAL	OUTV	output variable	0.0

Complete Restart

The block does not have a complete restart routine.

Normal Operation

The block has no modes other than normal operation.

Block-Internal Limits

The values of the input parameters are not restricted in the block; the parameters are not checked.

2.1.3 CRP_OUT: Change Range Peripheral Output

Application

The block adapts a floating-point value of the modular controller to the peripheral format.

Block Diagram

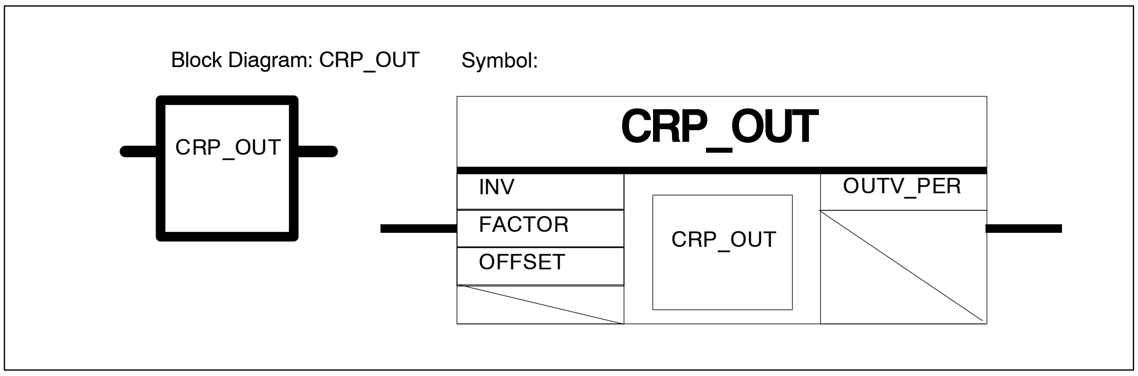


Figure 2-6 CRP_OUT, Block Diagram and Symbol

Functional Description

CRP_OUT converts an input value (normalized floating-point value of the modular controller) to the peripheral format of the analog I/Os.

Table 2-5 Input Value/Peripheral Value

Input Value in %	Peripheral Value
118.515	32767
100.000	27648
0.003617	1
0.000	0
–0.003617	–1
–100.000	–27648
–118.519	–32768

The floating-point value can be adapted using a scaling factor and an offset. The output is calculated as follows:

OUTV_PER = (INV * FACTOR + OFFSET) * 27648/100

Note

There is no check for positive/negative overflow.

Input Parameters

The following table shows the data type and structure of the input parameters of CRP_OUT.

Table 2-6 Input Parameters of CRP_OUT

Data Type	Parameter	Comment	Permitted Values	Default
REAL	INV	input variable	technical range of values	0.0
REAL	FACTOR	scaling factor		1.0
REAL	OFFSET	offset	technical range of values	0.0

Output Parameters

The following table shows the data type and structure of the output parameters CRP_OUT.

Table 2-7 Output Parameters of CRP_OUT

Data Type	Parameter	Comment	Default
WORD	OUTV_PER	output variable peripheral	0

Complete Restart

The block does not have a complete restart routine.

Normal Operation

The block has no modes other than normal operation.

Block-Internal Limits

The values of the input parameters are not restricted in the block; the parameters are not checked.

2.1.4 DEAD_T: Dead Time

Application

This block can be used in ratio controllers when the individual components have different distances to travel before they are brought together.

Block Diagram

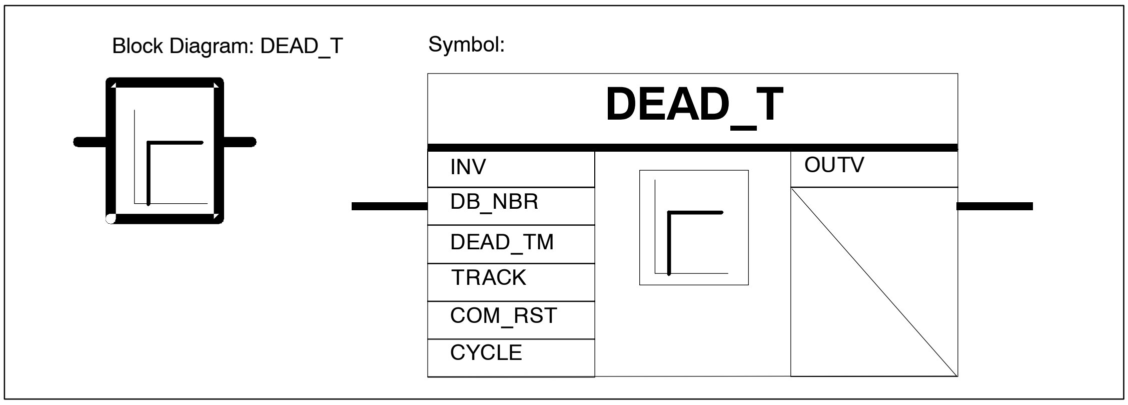


Figure 2-7 DEAD_T, Block Diagram and Symbol

Functional Description

The block delays the output of an input value by a selectable time (dead time). The input values are buffered in a shared data block. The maximum dead time depends on the length of this data block. The data in the shared data block DB_NBR are processed in the same way as in a ring buffer.

Table 2-8 Input Value

No.	Input Value	
0	INV[0]	⊓
1	INV[1]	| ↓
2	INV[2]	| ⇔ **OUTV**/**INV** read/write pointer
...	...	| ↓
...	...	| |
n	INV[n]	⊔ DEAD_TM = (n+1) • CYCLE
...	...	
m	INV[m]	

The location indicated by the read/write pointer is read and output to OUTV. Following this, INV is written to the same memory location. The memory location index for the read/write pointer is incremented by 1 each time the block is executed. When it reaches n, it returns to 0.

If the dead time DEAD_TM is specified and with a fixed sampling time CYCLE, the data block must allow

$$\frac{\text{DEAD\_TM}}{\text{CYCLE}}$$

save operations. A save operation (data type: REAL) occupies 4 bytes. DEAD_TM must be a whole multiple of CYCLE.

$$\text{DB length (in bytes) u=} \quad \frac{\text{DEAD\_TM}}{\text{CYCLE}} \quad * 4$$

If TRACK = TRUE, the input value is output directly.

Note

The block does not check whether or not a shared DB with the number DB_NBR really exists nor whether the parameters DEAD_TM (dead time) and CYCLE (sampling time) match the length of the data block. If the parameter assignment is incorrect, the CPU changes to STOP with an internal system error.

Input Parameters

The following table shows the data type and structure of the input parameters of DEAD_T.

Table 2-9 Input Parameters of DEAD_T

Data Type	Parameter	Comment	Permitted Values	Default
REAL	INV	input variable	technical range of values	0.0
BLOCK_DB	DB_NBR	data block number		DB 1
TIME	DEAD_TM	dead time	≥ CYCLE ≤ DB length/4∗CYCLE	10s
BOOL	TRACK	tracking OUTV = INV		FALSE
BOOL	COM_RST	complete restart		FALSE
TIME	CYCLE	sampling time	≥ 1ms	1s

Output Parameters

The following table shows the data type and structure of the output parameters DEAD_T.

Table 2-10 Output Parameters of DEAD_T

Data Type	Parameter	Comment	Default
REAL	OUTV	output variable	0.0

Shared Data Block DB_NBR

The following table shows the data type and Parameters of the shared data block.

Table 2-11 Parameters of the Shared Data Block

Data Type	Parameter	Comment	Permitted Values	Default
REAL	INV[0]	input variable [0]	technical range of values	0.0
REAL	INV[1]	input variable [1]	technical range of values	0.0
REAL	INV[2]	input variable [2]	technical range of values	0.0
REAL	INV[3]	input variable [3]	technical range of values	0.0

Complete Restart

During a complete restart, all the saved input values are deleted and OUTV = 0.0 is output.

Normal Operation

The input values are output delayed by the dead time. Online changes to the dead time setting can cause step changes in the output value.

- Tracking

  If tracking is turned on (TRACK = TRUE), the input value is transferred to OUTV without any delay. The buffering of the input values is not interrupted so that when tracking is turned off, the input values can still be output after the set dead time. If TRACK = FALSE, OUTV jumps to INV[DEAD_TM].

Block-Internal Limits

The values of the input parameters are not restricted in the block; the parameters are not checked.

Example

With a sampling time of CYCLE = 1 s and a dead time of DEAD_TM = 4 s, four input values must be buffered. The data area must then be 16 bytes long.

Table 2-12 Double Word/Input Value

Data Double Word	Input Value
0	INV[0]
4	INV[1]
8	INV[2]
12	INV[3]

2.1.5 DEADBAND: Dead Band

Application

If the process variable is affected by noise and the controller is optimally set, the noise will also affect the controller output. Due to the high switching frequency (step controller), this increases wear and tear on the actuator. Suppressing the noise prevents oscillation of the controller output.. When the dead band is used to form the error signal, the offset DEADB_O must be set to 0.0.

Block Diagram

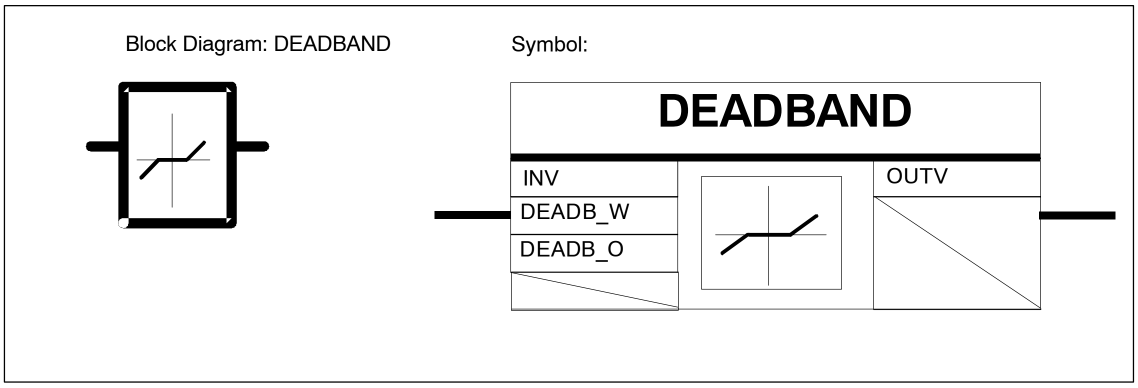


Figure 2-8 DEADBAND, Block Diagram and Symbol

Functional Description

The DEADBAND block suppresses small fluctuations in the input variable INV around a specified zero point. Outside this dead band, the output variable OUTV rises proportionally to the input variable. The block operates according to the following function:

OUTV = INV + DEADB_W – DEADB_O
when INV < DEADB_O – DEADB_W

OUTV = 0.0 when DEADB_O – DEADB_W ≤ INV
and INV ≤ DEADB_O + DEADB_W

OUTV = INV – DEADB_W – DEADB_O
when INV > DEADB_O + DEADB_W

The signal is falsified by the amount of the value DEADB_W. The mid point of the dead band is specified by DEADB_O.

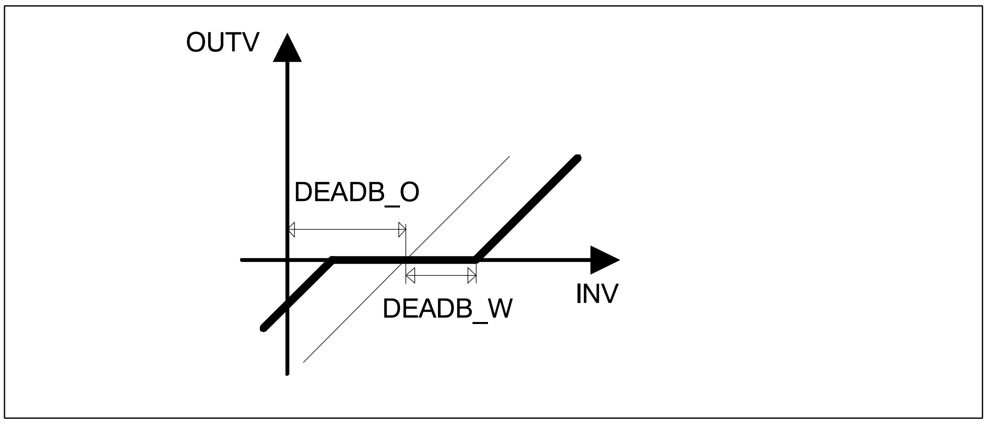


Figure 2-9 OUTV = f(INV)

The dead band width DEADB_W and dead band offset DEADB_O can be selected.

Input Parameters

The following table shows the data type and structure of the input parameters of DEADBAND.

Table 2-13 Input Parameters of DEADBAND

Data Type	Parameter	Comment	Permitted Values	Default
REAL	INV	input variable	technical range of values	0.0
REAL	DEADB_W	dead band width	tech. range ≥ 0.0	1.0
REAL	DEADB_O	dead band offset	technical range of values	0.0

Output Parameters

The following table shows the data type and structure of the output parameters DEADBAND.

Table 2-14 Output Parameters of DEADBAND

Data Type	Parameter	Comment	Default
REAL	OUTV	output variable	0.0

Complete Restart

The block has no complete restart routine.

Normal Operation

The block has no modes other than normal operation.

Block-Internal Limits

The values of the input parameters are not restricted in the block; the parameters are not checked. The dead band width can only have positive values.

Example

Figure 2-10 shows the suppression of noise using the offset.

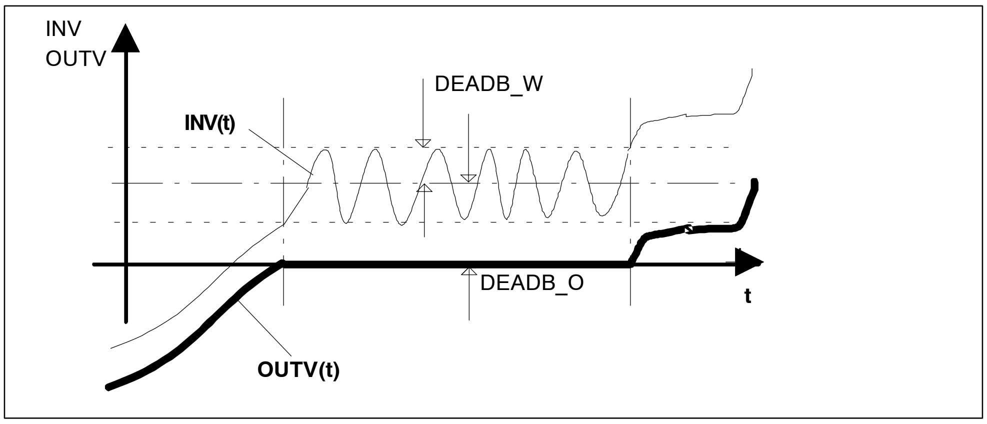


Figure 2-10 Suppression of Noise Using the Offset

2.1.6 DIF: Differentiator

Application

Process variables are differentiated dynamically. This means, for example, that the speed can be calculated from the distance traveled. The differentiator can be used for feedforward control, as a precontroller and to configure a controller.

Block Diagram

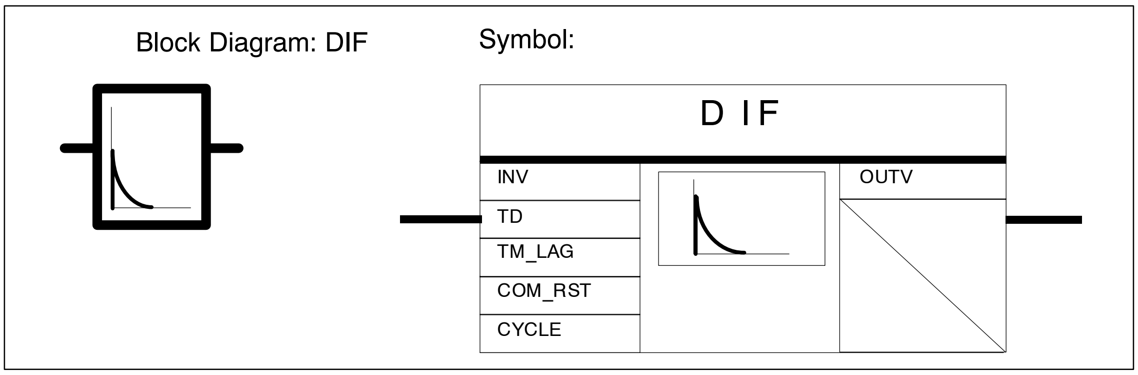


Figure 2-11 DIF, Block Diagram and Symbol

Functional Description

The block differentiates the input value over time and filters the signal with a 1st order lag.

Input Parameters

The following table shows the data type and structure of the input parameters of DIF.

Table 2-15 Input Parameters of DIF

Data Type	Parameter	Comment	Permitted Values	Default
REAL	INV	input variable	technical range of values	0.0
TIME	TD	derivative time value	≥ CYCLE	T#25s
TIME	TM_LAG	time lag		T#5s
BOOL	COM_RST	complete restart		FALSE
TIME	CYCLE	sampling time	≥ 1ms	T#1s

Output Parameters

The following table shows the data type and structure of the output parameters DIF.

Table 2-16 Output Parameters of DIF

Data Type	Parameter	Comment	Default
REAL	OUTV	output variable	0.0

Complete Restart

During a complete restart, all the signal outputs are set to 0. Internally, the differentiator is assigned the current input value **INV**. The transition to normal operation therefore does not cause any step change if the input variable remains the same.

Normal Operation

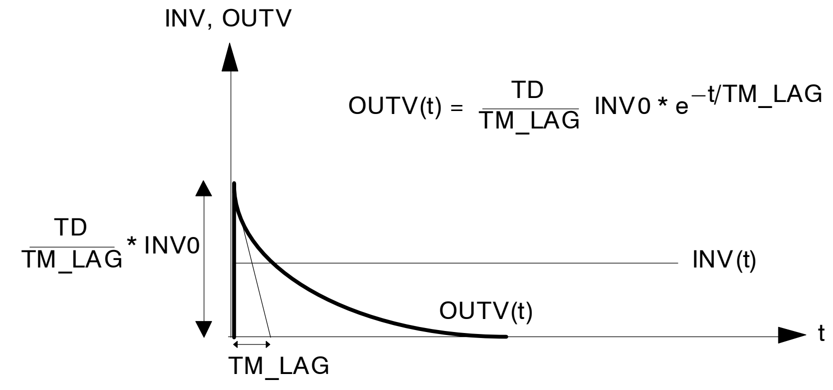
During differentiation, the block operates according to the following transfer function:

in the Laplace range: OUTV(s) / INV(s) = TD / (1+TM_LAG*s)

The time response of the differentiator is specified by the derivative time TD and the time lag TM_LAG. The corresponding step response is illustrated in the following diagram.

Step Response

Figures 2-12 and 2-13 show the step response of DIF (with and without lag).



INV, OUTV
$OUTV(t) = \frac{TD}{TM\_LAG} INV0 * e^{-t/TM\_LAG}$
$\frac{TD}{TM\_LAG} * INV0$
INV(t)
OUTV(t)
t
TM_LAG

where:		
	TD:	Derivative time
	TM_LAG:	Time lag constant
	INV0:	Input step change
	t:	Time
	INV:	Input variable
	OUTV:	Output variable

Figure 2-12 Step Response of DIF

If the value assigned for TM_LAG is less than or equal to CYCLE/2, the differentiator works without time lag. An input step change is applied to the output with the factor TD/CYCLE. After one cycle, the output returns to 0.0 again.

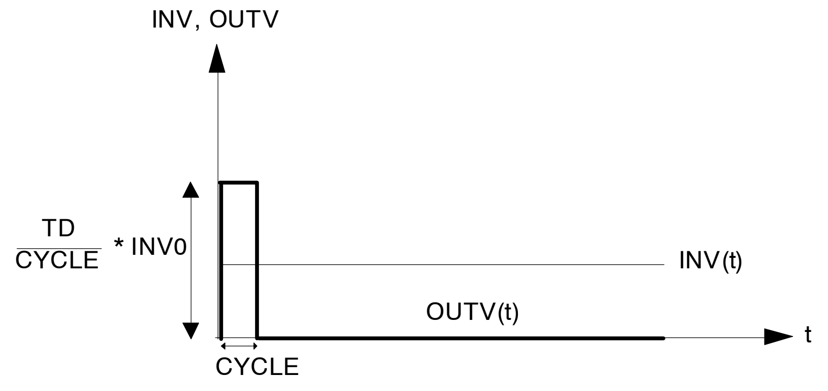


Figure 2-13 Step Response of DIF without Lag

Block-Internal Limits

The derivative time is limited downwards to the sampling time. The time lag is limited downwards to half the sampling time.

$TD_{\textbf{intern}}$ = CYCLE	when TD < CYCLE
$TM\_LAG_{\textbf{intern}}$ = CYCLE/2	when TM_LAG < CYCLE/2

The values of the other input parameters are not restricted in the block; the parameters are not checked.

2.1.7 ERR_MON: Error Signal Monitoring

Application

The block is used to form and monitor the error signal.

Block Diagram

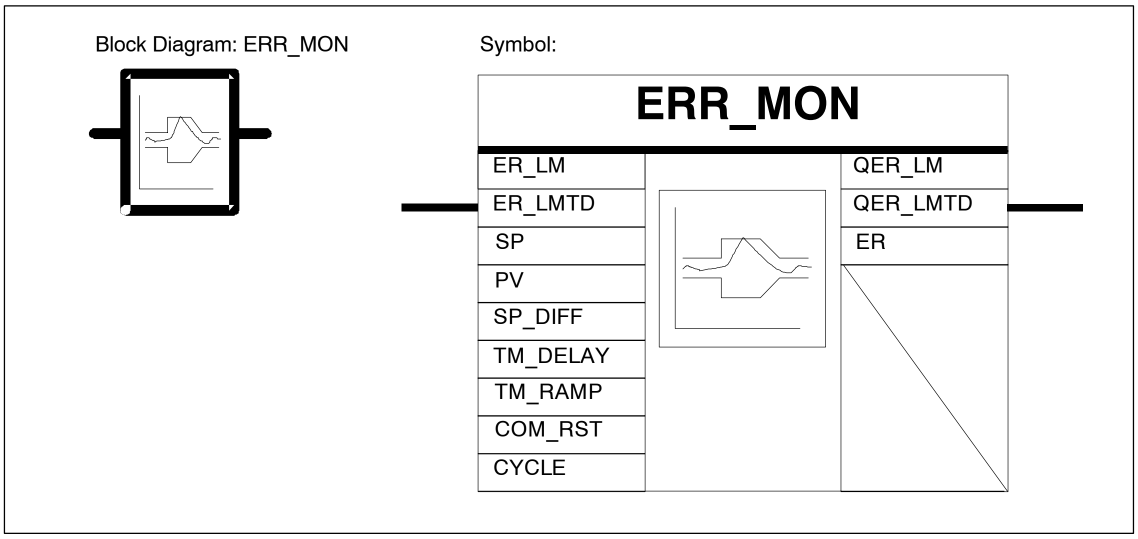
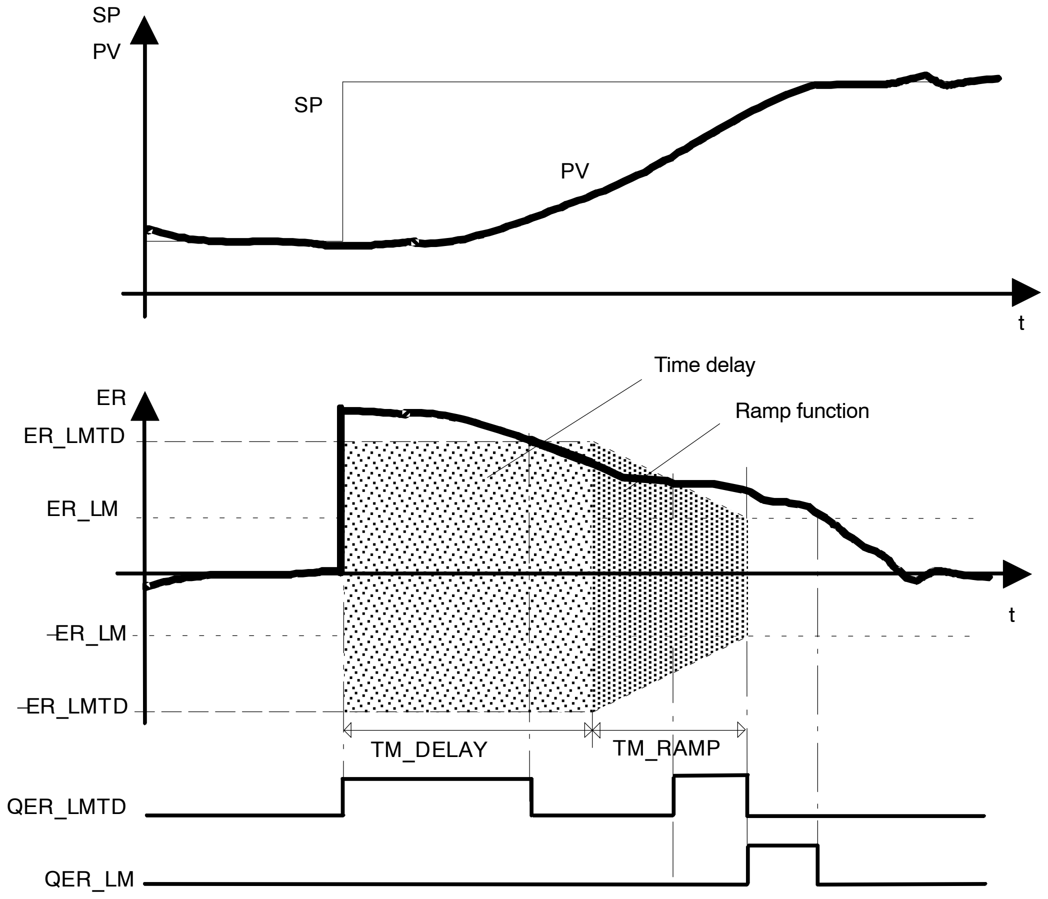


Figure 2-14 ERR_MON, Block Diagram and Symbol

Functional Description

The block calculates the error signal ER = SP – PV and monitors it for selectable limits. If there is a change in the setpoint greater than SP_DIF, the activation of the limit value signal ER_LM is suppressed for a selectable time (TM_DELAY+TM_RAMP); during this time the higher limit value ER_LMTD of ER is monitored. If ER_LMTD is exceeded, QER_LMTD = TRUE is output. Once the delay time has expired, ER_LMTD changes to ER_LM according to a ramp function. The on delay is started by a setpoint change. The slope of the ramp can be selected with the TM_RAMP parameter.



PV:	Process variable	TM_DELAY:	Time delay of the monitoring signal
SP:	Setpoint	TM_RAMP:	Ramp time constant
ER:	Error signal	QER_LM:	Error signal limit reached
t:	Time	QER_LMTD:	Error signal limit during time delay and ramp reached
ER_LM:	Error signal limit		
ER_LMTD:	Error signal limit during the time delay		

Figure 2-15 How ERR_MON Functions

Input Parameters

The following table shows the data type and structure of the input parameters of ERR_MON.

Table 2-17 Input Parameters of ERR_MON

Data Type	Parameter	Comment	Permitted Values	Default
REAL	ER_LM	error variable limit	tech. range > 0.0 und < ER_LMTD	10.0
REAL	ER_LMTD	error signal limit during time delay	tech. range > ER_LM	100.0
REAL	SP	setpoint variable	technical range of values	0.0
REAL	PV	process variable	technical range of values	0.0
REAL	SP_DIFF	setpoint difference	tech. range > 0.0	10.0
TIME	TM_DELAY	time delay of the monitoring signal		T#60s
TIME	TM_RAMP	time constant of ramp		T#60s
BOOL	COM_RST	complete restart		FALSE
TIME	CYCLE	sampling time	≥ 1ms	T#1s

Output Parameters

The following table shows the data type and structure of the output parameters ERR_MON.

Table 2-18 Output Parameters of ERR_MON

Data Type	Parameter	Comment	Default
BOOL	QER_LM	error signal limit reached	FALSE
BOOL	QER_LMTD	error signal limit during time delay reached	FALSE
REAL	ER	error signal	0.0

Complete Restart

During a complete restart, the QER_LM and QER_LMTD signals and the error signal output ER are reset.

Normal Operation

The block has no modes other than normal operation.

Block-Internal Limits

The values of the input parameters are not restricted in the block; the parameters are not checked.

2.1.8 INTEG: Integrator

Application

Process variables are integrated dynamically. This means, for example, that the distance traveled is calculated from the speed. The integrator can be used to configure a controller.

Block Diagram

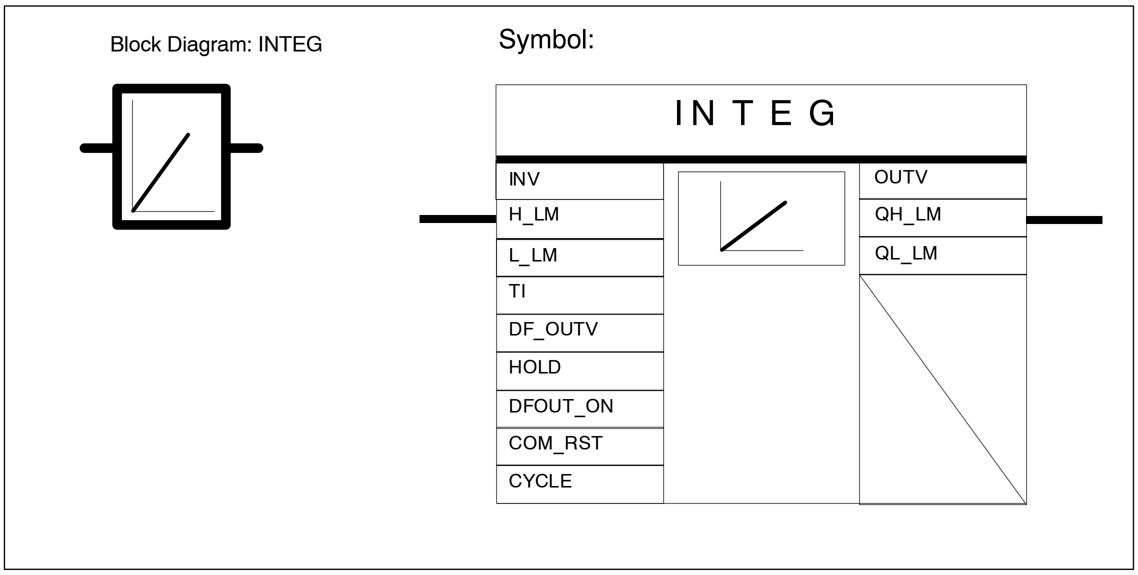


Figure 2-16 INTEG, Block Diagram and Symbol

Functional Description

The block integrates the input variable over time and restricts the integral to a selectable upper and lower limit. The limit of the output variable is indicated by signal bits.

Input Parameters

The following table shows the data type and structure of the input parameters of INTEG.

Table 2-19 Input Parameters of INTEG

Data Type	Parameter	Comment	Permitted Values	Default
REAL	INV	input variable	technical range of values	0.0
REAL	H_LM	high limit	tech. range > L_LM	100.0
REAL	L_LM	low limit	tech. range < H_LM	0.0
TIME	TI	reset time	≥ CYCLE	T#25s
REAL	DF_OUTV	default output variable	technical range of values	0.0
BOOL	HOLD	integrator hold		FALSE
BOOL	DFOUT_ON	default output variable on		FALSE
BOOL	COM_RST	complete restart		FALSE
TIME	CYCLE	sampling time	≥ 1ms	T#1s

Output Parameters

The following table shows the data type and structure of the output parameters INTEG.

Table 2-20 Output Parameters of INTEG

Data Type	Parameter	Comment	Default
REAL	OUTV	output variable	0.0
BOOL	QH_LM	high limit reached	FALSE
BOOL	QL_LM	low limit reached	FALSE

Complete Restart

During a complete restart, the OUTV output is reset to 0.0. If DFOUT_ON = TRUE is set DF_OUTV is output. The limiting of the output remains effective during a complete restart and the limit signal bits are also effective. When the controller changes to normal operation, the block integrates starting at OUTV.

If you want the integrator to start at a particular operating point when a complete start is executed, the operating point must be entered at the input DF_OUTV. When the block is called during the complete restart routine, DFOUT_ON = TRUE must be set and then reset to DFOUT_ON = FALSE at the cyclic interrupt priority level.

Normal Operation

In addition to normal operation, the block has the following modes:

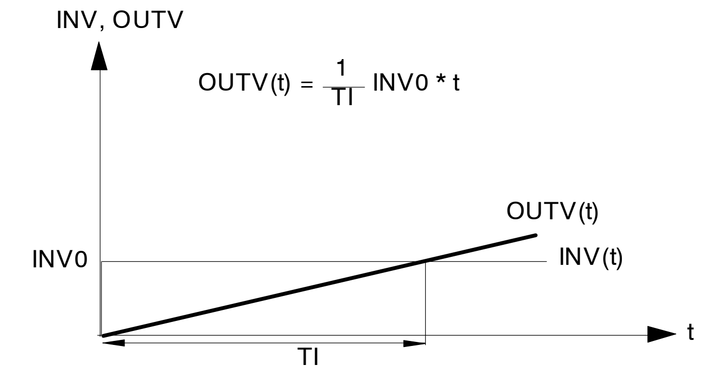
Modes	DFOUT_ON	HOLD
Integrate	FALSE	FALSE
Integrator hold	FALSE	TRUE
Default output variable	TRUE	any

- **Integration**

When integrating, the block operates according to the following transfer function:

in the Laplace range:
OUTV(s) / INV(s) = 1 / (TI∗s)

The time response of the integrator is specified by the reset time TI. The corresponding step response is illustrated in the following diagram.



where : TI: Reset time
INV0: Input step change
t: Time
INV: Input variable
OUTV: Output variable

Figure 2-17 Step Response of INTEG

The output and the integrator buffer are restricted to the selectable limit values H_LM and L_LM. If the output is within the limits, this is indicated by the signal bits QH_LM and QL_LM.

- **Integrator Hold**

If HOLD is set to TRUE, the integrator remains at its current output value OUTV. When HOLD is reset to FALSE, the integrator continues to integrate starting at the current output value OUTV.

- **Default Value at the Output**

If DFOUT_ON = TRUE is set, DF_OUTV is applied to the output. The limit is effective. If this is reset so that DF_OUTV_ON = FALSE is set, the integrator begins to integrate starting at the value DF_OUTV.

Block-Internal Limits

The reset time is limited downwards by the sampling time:

$TI_{intern} = CYCLE$ when $TI < CYCLE$

The values of the other input parameters are not restricted in the block; the parameters are not checked.

Example

Figure 2-18 shows an example of DFOUT_ON, HOLD and limitation.

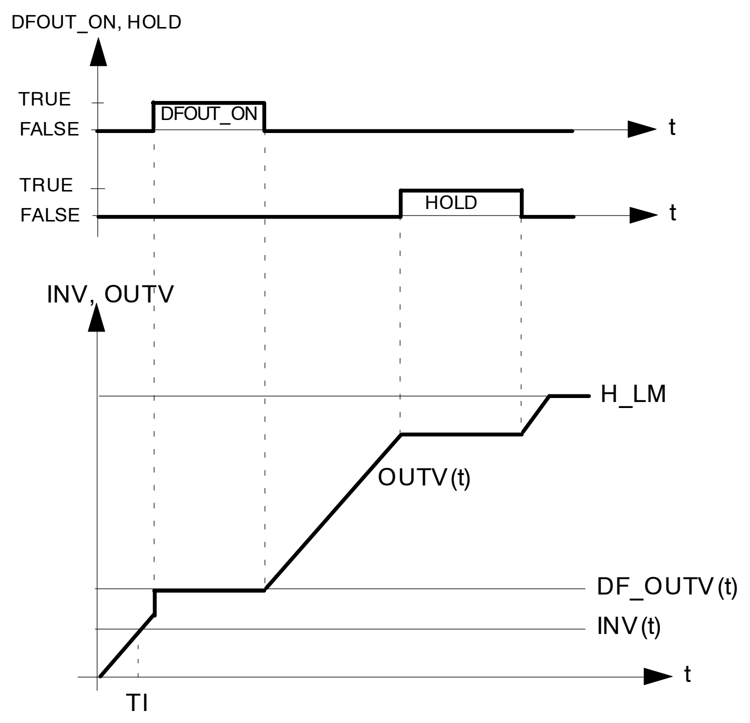


Figure 2-18 Example of DFOUT_ON, HOLD and limitation

2.1.9 LAG1ST: First-Order Lag Element

Application

The block can be used as a delay or filter element.

Block Diagram

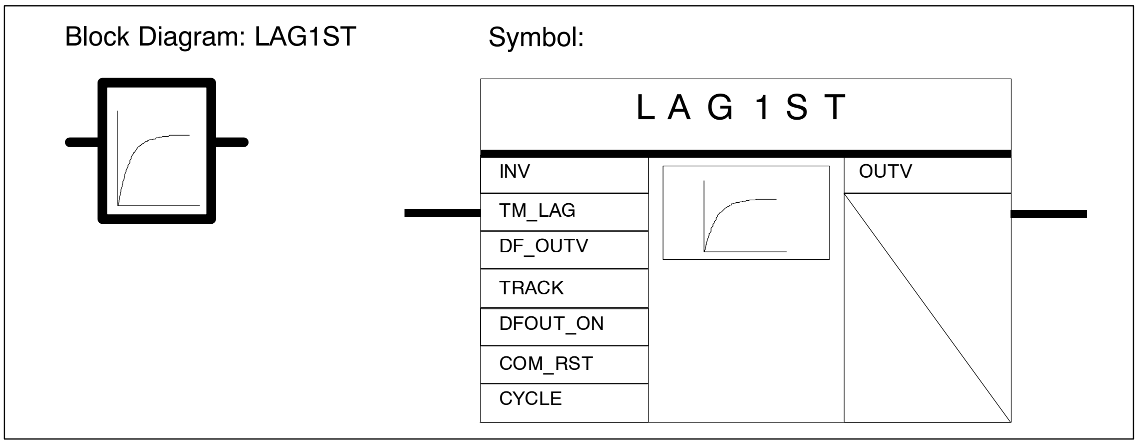


Figure 2-19 LAG1ST, Block Diagram and Symbol

Functional Description

The block filters the input variable with a 1st order time lag. The time lag can be selected.

Input Parameters

The following table shows the data type and structure of the input parameters of LAG1ST.

Table 2-21 Input Parameters of LAG1ST

Data Type	Parameter	Comment	Permitted Values	Default
REAL	INV	input variable	technical range of values	0.0
TIME	TM_LAG	time lag		T#25s
REAL	DF_OUTV	default output variable	technical range of values	0.0
BOOL	TRACK	tracking OUTV = INV		FALSE
BOOL	DFOUT_ON	default output variable on		FALSE
BOOL	COM_RST	complete restart		FALSE
TIME	CYCLE	sampling time	≥1ms	T#1s

Output Parameters

The following table shows the data type and structure of the output parameters LAG1ST.

Table 2-22 Output Parameters of LAG1ST

Data Type	Parameter	Comment	Default
REAL	OUTV	output variable	0.0

Complete Restart

During a complete restart, the output OUTV is reset to 0.0. If DFOUT_ON = TRUE is set, DF_OUTV is output. When the controller changes to normal operation, the block continues to operate starting from OUTV.

Normal Operation

Apart from normal operation, the block has the following modes:

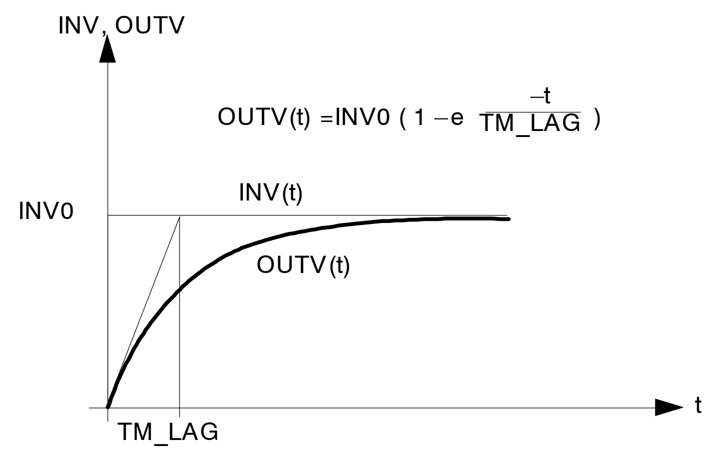
Modes	DFOUT_ON	HOLD
Filtering	FALSE	FALSE
Tracking	FALSE	TRUE
Default output variable	TRUE	any

- **Filtering**

When filtering, the block operates according to the following transfer function:

in the Laplace range: OUTV(s) / INV(s) = 1 / (1+TM_LAG∗s)

The time response of the lag element is specified by the time lag TM_LAG. The corresponding step response is shown in the following figure.



where:	INV:	Input variable
	OUTV:	Output variable
	INV0:	Input step change
	TM_LAG:	Time lag constant

Figure 2-20 Step Response of LAG1ST

- **Tracking**

If TRACK = TRUE, the input value INV is switched to the output OUTV.

- **Default Value at the Output**

If DFOUT_ON = TRUE is set, DF_OUTV is applied to the output. If this is reset so that DF_OUTV_ON = FALSE is set, the lag element filters starting from the value DF_OUTV.

Block-Internal Limits

The time lag is limited downwards to half the sampling time.

$TM\_LAG_{intern} = CYCLE/2$ when $TM\_LAG < CYCLE/2$

The values of the other input parameters are not restricted in the block; the parameters are not checked.

Example

Figure 2-21 shows an example with DFOUT_ON and TRACK.

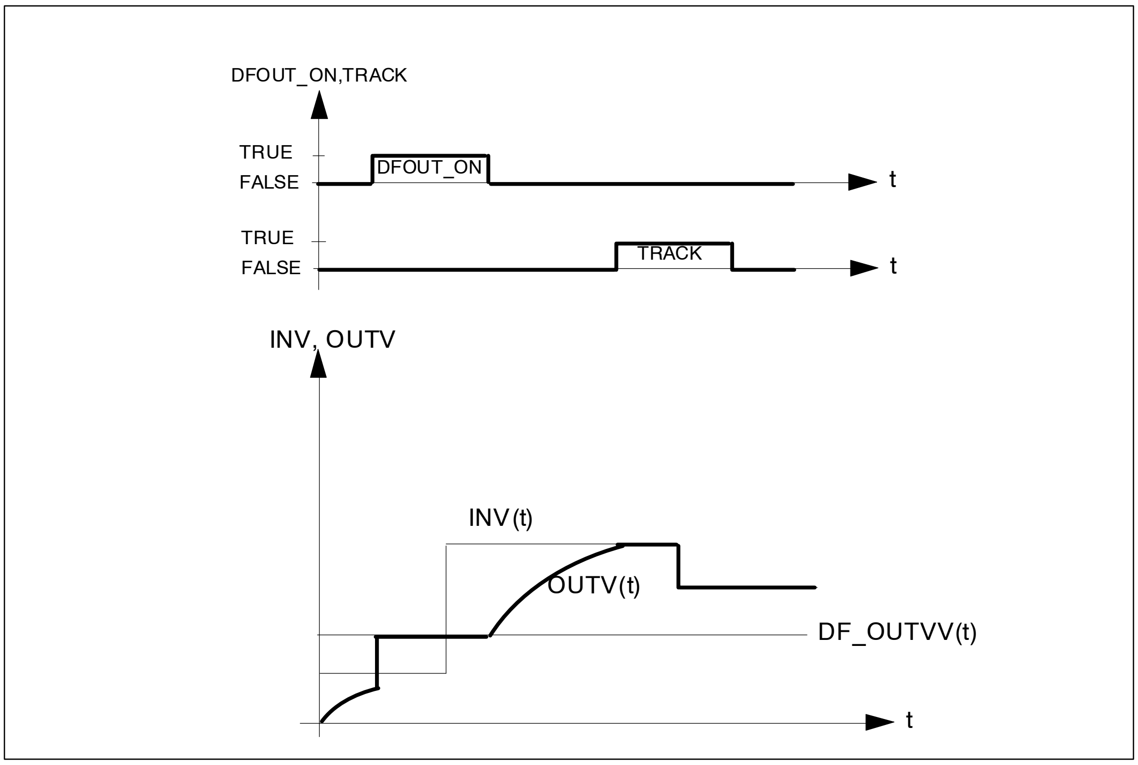


Figure 2-21 Example with DFOUT_ON and TRACK

2.1.10 LAG2ND: Second-Order Lag Element

Application

The block is used to simulate system components for precontrollers and two-loop controllers.

Block Diagram

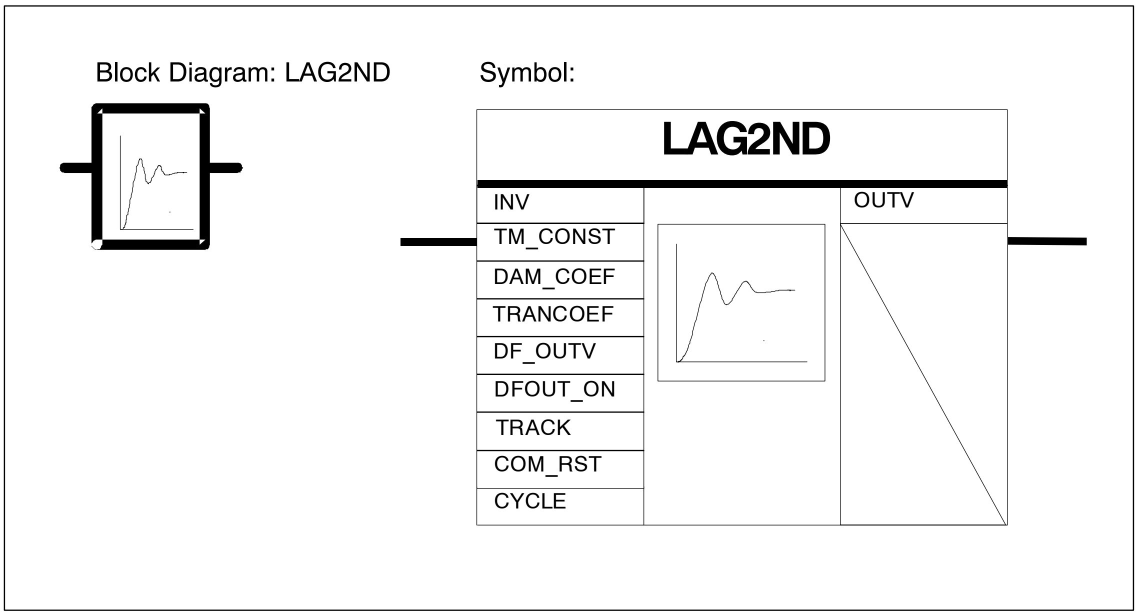


Figure 2-22 LAG2ND, Block Diagram and Symbol

Functional Description

The block implements a 2nd order lag capable of oscillation.

Transfer Function

The transfer function in the Laplace range is:

OUTV(s) / INV(s) = TRANCOEF / (1 + 2∗DAM_COEF*TM_CONST∗s + TM_CONST^2∗s^2)

If DAM_COEF >= 1 (aperiodic case), the transfer element can be represented as a series circuit with two PT1 elements.

OUTV(s) / INV(s) = TRANCOEF / (1 + T1*s) * 1 / (1 + T2 * s)

The time constants are recalculated as follows:

$$T1 = TM\_CONST\ (DAM\_COEF + \sqrt{DAM\_COEF^2 - 1}\ )$$

$$T2 = TM\_CONST\ (DAM\_COEF - \sqrt{DAM\_COEF^2 - 1}\ )$$

Block Diagram

Figure 2-23 shows the block diagram of the LAG2ND.

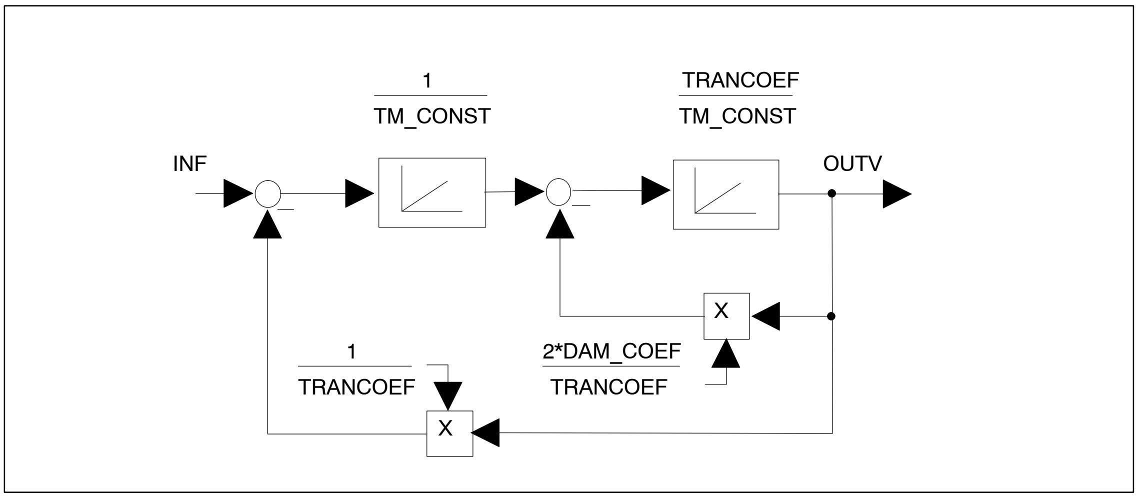
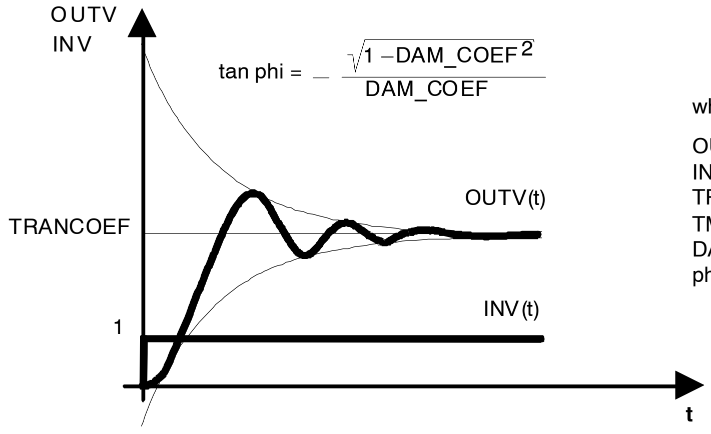


Figure 2-23 Structure of LAG2ND Made Up of Elementary Transfer Elements

Step Response

$$OUTV(t) = TRANCOEF + \frac{TRANCOEF}{\sqrt{1 - DAM\_COEF^2}}\, e^{-\frac{DAM\_COEF}{TM\_CONST}t} \sin\left(\frac{\sqrt{1 - DAM\_COEF^2}}{TM\_CONST}\, t - phi\right)$$

$$\tan phi = -\frac{\sqrt{1 - DAM\_COEF^2}}{DAM\_COEF}$$



where:

OUTV(t):	Output variable
INV(t):	Input variable
TRANCOEF:	Transfer coefficient
TM_LAG:	Time lag constant
DAM_COEF:	Damping coefficient
phi:	Supplementary angle

Figure 2-24 Step Response of the LAG2ND Element Capable of Oscillation

Input Parameters

The following table shows the data type and structure of the input parameters of LAG2ND.

Table 2-23 Input Parameters of LAG2ND

Data Type	Parameter	Comment	Permitted Values	Default
REAL	INV	input variable	technical range of values	0.0
TIME	TM_CONST	time constant		T#10s
REAL	DAM_COEF	damping coefficient		1.0
REAL	TRANCOEF	transfer coefficient		1.0
REAL	DF_OUTV	default output variable	technical range of values	0.0
BOOL	DFOUT_ON	default output variable on		FALSE
BOOL	TRACK	tracking OUTV = INV		FALSE
BOOL	COM_RST	complete restart		FALSE
TIME	CYCLE	sampling time	≥ 1ms	T#1s

Output Parameters

The following table shows the data type and structure of the output parameters LAG2ND.

Table 2-24 Output Parameters of LAG2ND

Data Type	Parameter	Comment	Default
REAL	OUTV	output variable	0.0

Complete Restart

During a complete restart, output OUTV is set to 0.0. If DFOUT_ON = TRUE, then DF_OUTV is output.

Normal Operation

Apart from normal operation, the block has the following modes:

- Tracking

If TRACK = TRUE is set, then OUTV = INV; the internal historical values are set to INV.

- Default at the Output

If DFOUT_ON = TRUE is set, then DF_OUTV is output, the internal historical values are set to DF_OUTV. DFOUT_ON has higher priority than TRACK.

Block-Internal Limits

The time constant TM_CONST is limited downwards to half the sampling time.

$TM\_CONST_{intern} = CYCLE/2$ when $TM\_CONST < CYCLE/2$

The values of the other input parameters are not restricted in the block; the parameters are not checked.

2.1.11 LIMALARM: Limit Alarm

Application

Illegal or dangerous states can occur in a system if process values (for example, motor speed, temperature or pressure) exceed or fall below critical values. Such limit violations must be detected and signaled to allow an appropriate reaction.

Block Diagram

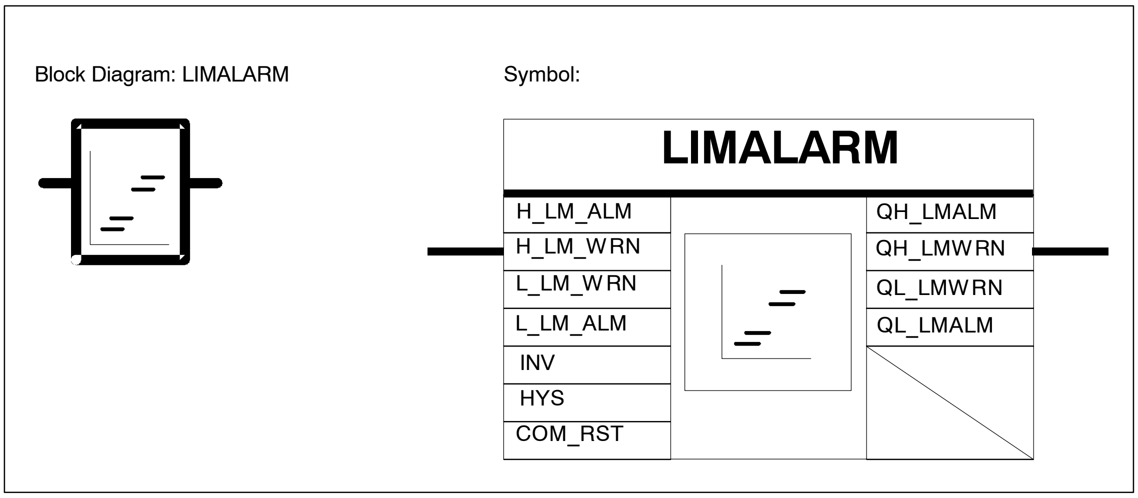


Figure 2-25 LIMALARM, Block Diagram and Symbol

Functional Description

Four limit values can be selected for the input variable INV. If one of these limits is reached and exceeded, a limit signal is output. A hysteresis can be set for the off threshold.

Figure 2-26 shows how the LIMALARM block functions:

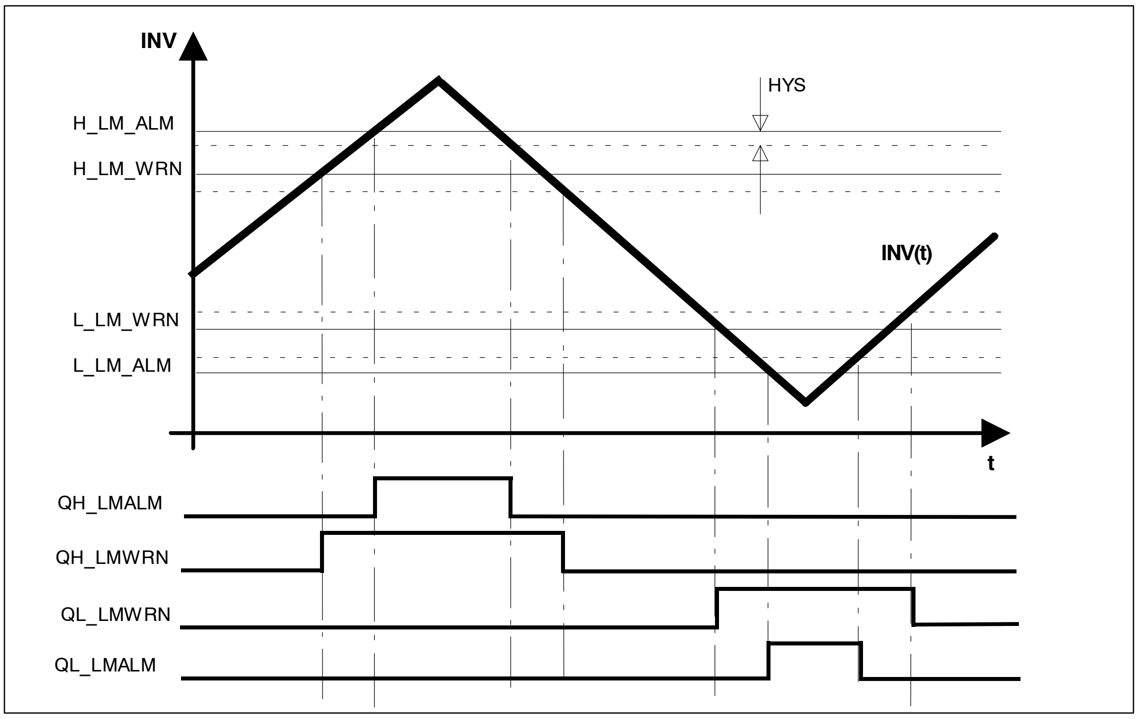


Figure 2-26 How the LIMALARM Block Functions

Input Parameters

The following table shows the data type and structure of the input parameters of LIMALARM.

Table 2-25 Input Parameters of LIMALARM

Data Type	Parameter	Comment	Permitted Values	Default
REAL	H_LM_ALM	high limit alarm	tech. range > H_LM_WRN	100.0
REAL	H_LM_WRN	high limit warning	tech. range > L_LM_WRN	90.0
REAL	L_LM_WRN	low limit warning	tech. range > L_LM_ALM	10.0
REAL	L_LM_ALM	low limit alarm	tech. range < L_LM_WRN	0.0
REAL	INV	input variable	technical range of values	0.0
REAL	HYS	hysteresis	technical range of values	1.0
BOOL	COM_RST	complete restart		FALSE

Output Parameters

The following table shows the data type and structure of the output parameters LIMALARM.

Table 2-26 Output Parameters of LIMALARM

Data Type	Parameter	Comment	Default
BOOL	QH_LMALM	high limit alarm reached	FALSE
BOOL	QH_LMWRN	high limit warning reached	FALSE
BOOL	QL_LMWRN	low limit warning reached	FALSE
BOOL	QL_LMALM	low limit alarm reached	FALSE

Complete Restart

During a complete restart, all the signal outputs are set to FALSE.

Normal Operation

The block operates according to the following functions:

QH_LMALM = TRUE	if	INV rises	and	INV >= H_LM_ALM
	or	INV falls	and	INV >= H_LM_ALM – HYS
QH_LMWRN = TRUE	if	INV rises	and	INV >= H_LM_WRN
	or	INV falls	and	INV >= H_LM_WRN – HYS
QL_LMWRN = TRUE	if	INV falls	and	INV <= L_LM_WRN
	or	INV rises	and	INV <= L_LM_WRN + HYS
QL_LMALM = TRUE	if	INV falls	and	INV <= L_LM_ALM
	or	INV rises	and	INV <= L_LM_ALM + HYS

The block can only function properly when:

L_LM_ALM < L_LM_WRN < H_LM_WRN < H_LM_ALM

The limits are set at the inputs H_LM_ALM, H_LM_WRN, L_LM_WRN and L_LM_ALM. If the input variable INV exceeds the limits, the output signal bits QH_LMALM, QH_LMWRN, QL_LMWRN and QL_LMALM are set. To avoid fast setting and resetting of the signal bits, the input value must also overcome a hysteresis HYS before the outputs are reset.

Block-Internal Limits

The values of the parameters are not limited in the block; the parameters are not checked.

2.1.12 LIMITER: Limiter

Application

If the parameters are set dynamically (for example setpoints calculated from process variables) these can be set to values that are not permitted for the process. With LIMITER, you can keep values within the permitted range.

Block Diagram

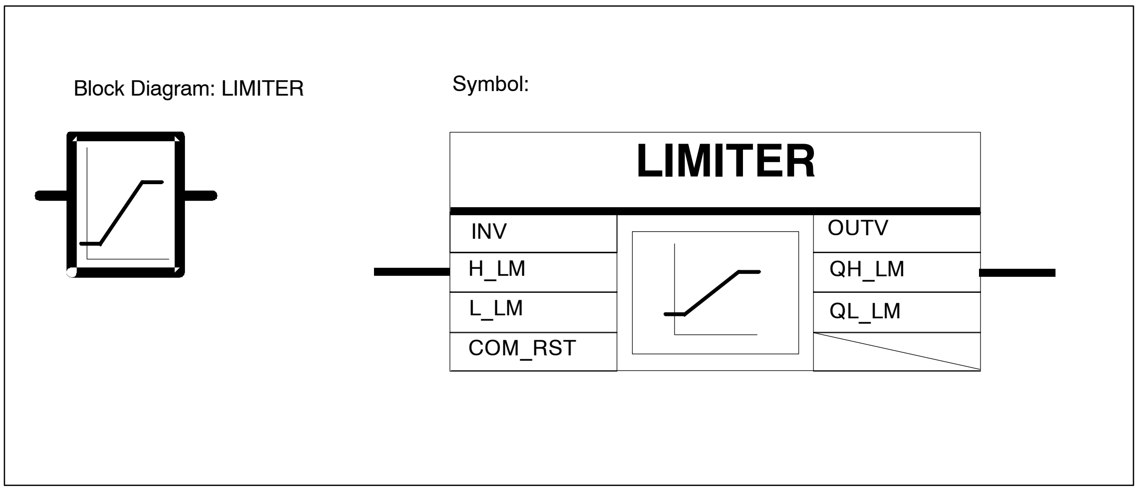


Figure 2-27 LIMITER, Block Diagram and Symbol

Functional Description

The block restricts the output variable OUTV to a selectable high and low limit H_LM and L_LM when the input variable INV is outside these limits. The limits of OUTV are signaled via outputs QH_LM and QL_LM.

Figure 2-28 shows how the LIMITER block functions:

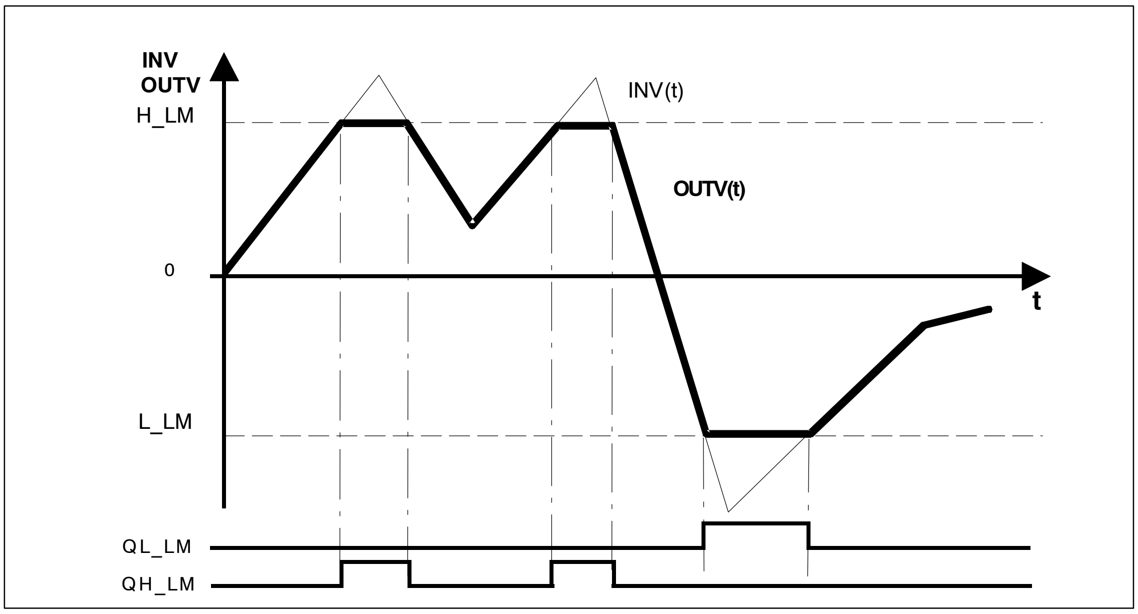


Figure 2-28 Functions of the LIMITER Block

Input Parameters

The following table shows the data type and structure of the input parameters of LIMITER.

Table 2-27 Input Parameters of LIMITER

Data Type	Parameter	Comment	Permitted Values	Default
REAL	INV	input variable	technical range of values	0.0
REAL	H_LM	high limit	technical range of values > L_LM	100.0
REAL	L_LM	low limit	technical range of values < H_LM	0.0
BOOL	COM_RST	complete restart		FALSE

Output Parameters

The following table shows the data type and structure of the output parameters LIMITER.

Table 2-28 Output Parameters of LIMITER

Data Type	Parameter	Comment	Default
REAL	OUTV	output variable	0.0
BOOL	QH_LM	high limit reached	FALSE
BOOL	QL_LM	low limit reached	FALSE

Complete Restart

During a complete restart, all signal outputs are set to FALSE; 0.0 is output at OUTV.

Normal Operation

The block operates according to the following functions:

OUTV = H_LM,	QH_LM = TRUE, QL_LM = FALSE if	INV >= H_LM
OUTV = L_LM,	QH_LM = FALSE, QL_LM = TRUE if	INV <= L_LM
OUTV = INV,	QH_LM = FALSE, QL_LM = FALSEif	L_LM < INV < H_LM

The block can only operate correctly when: L_LM < H_LM.

Block-Internal Limits

The values of the parameters are not limited in the block; the parameters are not checked.

2.1.13 LMNGEN_C: Output Continuous PID Controller

Application

The block is used to structure a continuous PID controller. It contains the manipulated value processing of the controller.

Block Diagram

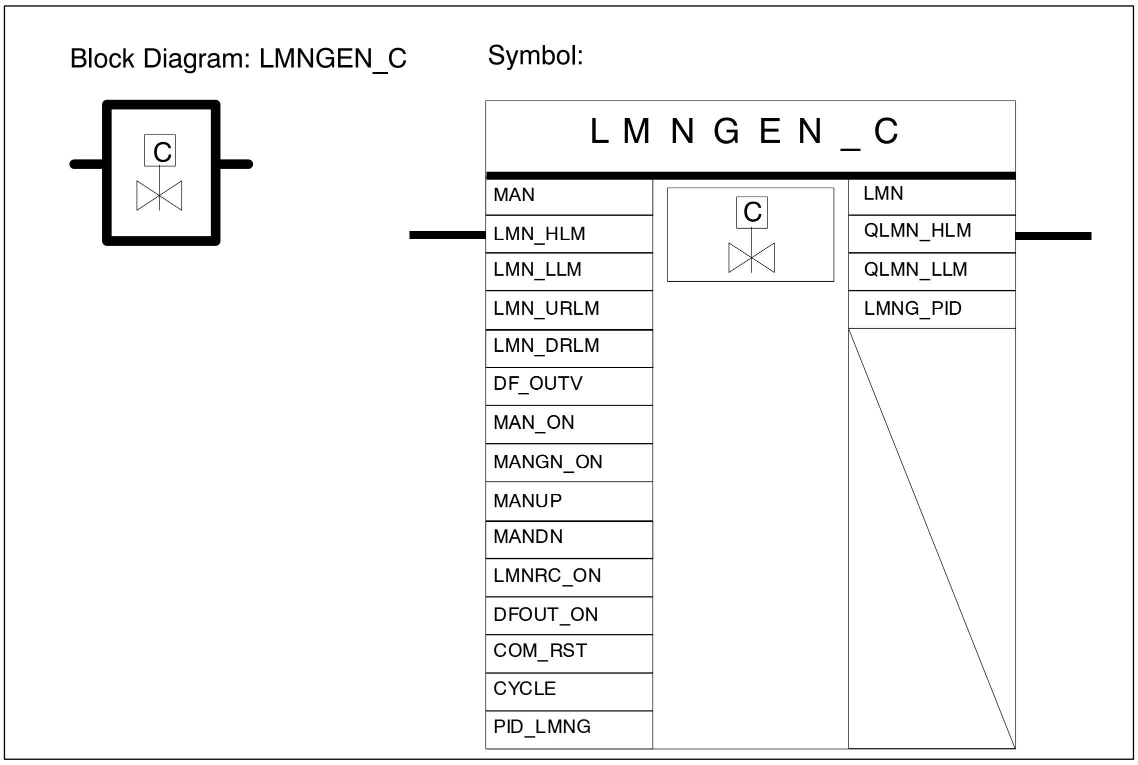


Figure 2-29 LMNGEN_C, Block Diagram and Symbol

Functional Description

The block includes the manual-automatic switchover. In the manual mode, you can specify an absolute value or increase or reduce the value with switches. The manipulated value and the rate of change of the manipulated value can be restricted to selectable limits.

The block is always used in conjunction with the PID algorithm block.

- **Continuous PID controller: PID + LMNGEN_C**

Figure 2-30 shows the connection of the PID controller.

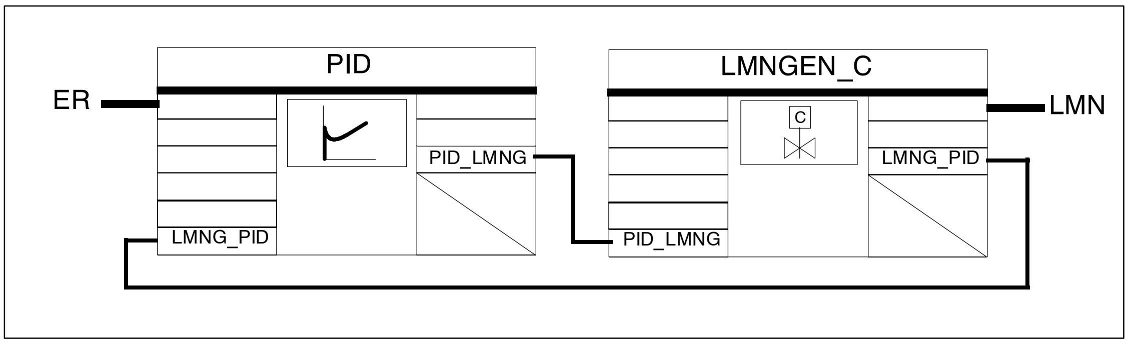


Figure 2-30 Connection of the Continuous PID Controller

While the PID algorithm is located in a cyclic interrupt priority class, with a cycle time adapted to the dominant system time constant, the LMNGEN_C block, that influences the actuator, can be located in a faster cyclic interrupt priority class to allow manual interventions. The block is connected using the structured input-output parameters PID_LMNG and LMNG_PID.

Input Parameters

The following table shows the data type and structure of the input parameters of LMNGEN_C.

Table 2-29 Input Parameters of LMNGEN_C

Data Type	Parameter	Comment	Permitted Values	Default
REAL	MAN	manual value	technical range of values	0.0
REAL	LMN_HLM	manipulated value high limit	tech. range > LMN_LLM	100.0
REAL	LMN_LLM	manipulated value low limit	tech. range < LMN_HLM	0.0
REAL	LMN_URLM	manipulated value up rate limit [1/s]	> 0.0	10.0
REAL	LMN_DRLM	manipulated value down rate limit [1/s]	> 0.0	10.0
REAL	DF_OUTV	default output variable	technical range of values	0.0
BOOL	MAN_ON	manual value on		TRUE
BOOL	MANGN_ON	manual value generator on		FALSE
BOOL	MANUP	manual value up		FALSE
BOOL	MANDN	manual value down		FALSE
BOOL	LMNRC_ON	manipulated value rate of change on		FALSE

Table 2-29 Input Parameters of LMNGEN_C, continued

Data Type	Parameter	Comment	Permitted Values	Default
BOOL	DFOUT_ON	default output variable on		FALSE
BOOL	COM_RST	complete restart		FALSE
TIME	CYCLE	sampling time	≥ 1ms	T#1s
STRUC	PID_LMNG	PID-LMNGEN interface		

Output Parameters

The following table shows the data type and structure of the output parameters LMNGEN_C.

Table 2-30 Output Parameters of LMNGEN_C

Data Type	Parameter	Comment	Default
REAL	LMN	manipulated value	0.0
BOOL	QLMN_HLM	high limit of manipulated value reached	FALSE
BOOL	QLMN_LLM	low limit of manipulated value reached	FALSE
STRUC	LMNG_PID	PID-LMNGEN interface	

Complete Restart

During a complete restart, the default value DF_OUTV is switched to the LMN output regardless of the default bit DFOUT_ON. The limitation of the output and the limit signal bits are also effective in a complete restart. When the controller changes to normal operation, the block continues to operate starting with DF_OUTV.

Normal Operation

Apart from normal operation, the block has the following modes:

Modes	DFOUT_ON	MAN_ON
Automatic	FALSE	FALSE
Manual	FALSE	TRUE
Default output variable	TRUE	any

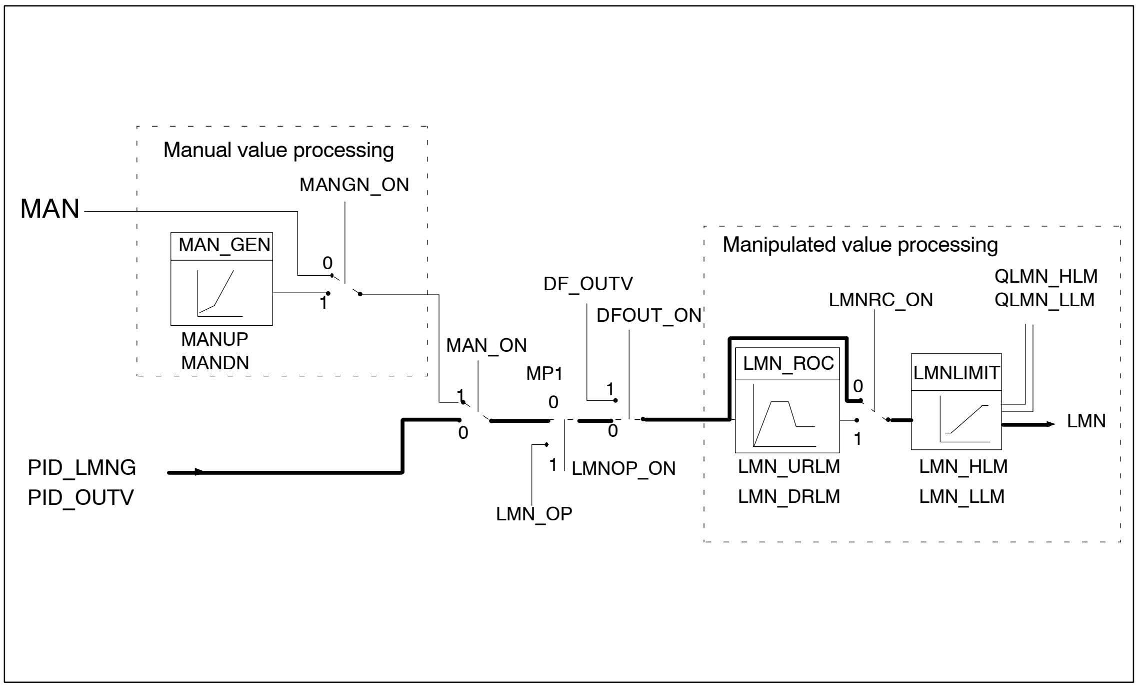


Figure 2-31 Block Diagram Manipulated Value Processing of the Continuous PID Controller

- **Automatic Mode**

If no other mode is selected, the value calculated by the PID algorithm is transferred to manipulated value processing. The switchover to the automatic mode does not cause any sudden change if the manipulated value rate of change function is activated (LMNRC_ON = TRUE).

- **Manual Mode**

Using the MAN_ON switch, you can switch over to manual operation and interrupt the control loop. If MANGN_ON = TRUE, you can increase or reduce the manipulated value starting from the current value using the switches MANUP and MANDN within the limits LMN_HLM and LMN_LLM. The rate of change depends on the limits:

During the first 3 seconds after setting MANUP or MANDN:

dLMN/dt = (LMN_HLM – LMN_LLM) / 100 s

then: dLMN/dt = (LMN_HLM – LMN_LLM) / 10 s

If MANGN_ON = FALSE and MAN_ON = TRUE, the input value MAN is switched through as the manipulated value.

The upper and lower limits of the manipulated value must be applied to inputs LMN_HLM and LMN_LLM. The rate of change of the manipulated value can also be limited. The up and down rate of change limits or the manipulated value are set at inputs LMN_URLM and LMN_DRLM and activated with the switch LMNRC_ON. The manipulated value appears at the LMN output. The limiting of the manipulated value by the LMN_HLM and LMN_LLM limits is signaled by the bits QLMN_HLM and QLMN_LLM.

- **Default Output Variable**

If DFOUT_ON = TRUE is set, the default value DF_OUTV is applied to the LMN output. The manipulated value limits are effective and signaled. The switchover from or to “default output variable” does not cause a sudden change providing the rate of change function (LMNRC_ON = TRUE) is activated.

Block-Internal Limits

The values of the parameters are not limited in the block; the parameters are not checked.

Influencing the Manipulated Value with the *configuration tool*

LMN Display and Setting in the Loop Monitor

Die configuration tool has its own interface to the LMNGEN_C block. It is therefore always possible to interrupt the manipulated variable branch (for example for test purposes when working on a programming device or PC on which the *configuration tool* is loaded) and to set your own manipulated values (LMN_OP). (Figure 2-32).

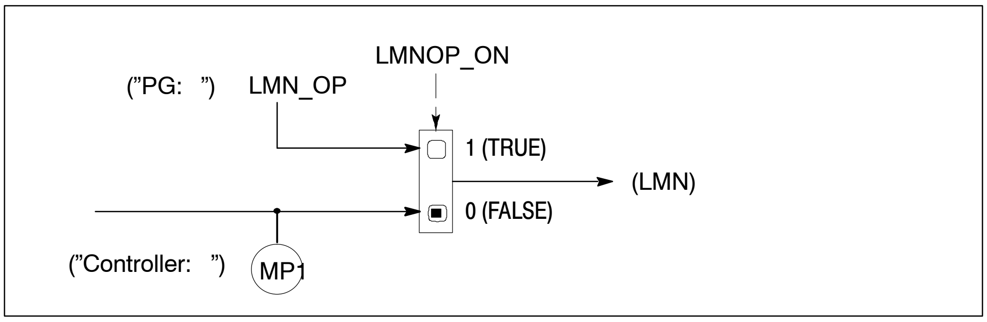


Figure 2-32 Intervention in the Manipulated Value Branch Using the *configuration tool*

The loop monitor has a field labeled “manipulated value”. Here, in the upper field (”Controller: ”), the manipulated value currently applied to measuring point MP1 is displayed. In the field below (”PG: ”), you can set the parameter LMN_OP.

Switching over to Manual Manipulation Value with the *configuration tool*

If the switch in the *configuration tool* is set to “PG: ”, the switching signal of the structure switch LMNOP_ON is set to TRUE in the controller FB and LMN_OP is switched through to the manipulated value LMN.

If the rate of change limit LMN_ROC is activated in the manipulated variable branch , it is possible to switch between “PG: ” and “Controller: ” without causing any sudden change. The value to which the manipulated variable switches back (MP1) can be seen in the “Controller: ” display field of the loop monitor. LMN then returns to this value at the rate of change set at LMN_ROC.

These interventions only affect the process after you transfer them to the programmable controller by clicking the “Send” button in the loop monitor.

The parameters LMNOP_ON, LMN_OP, and MP1 are static variables and are not available as input/output parameters for the block. The parameters should not be connected and should only be used and monitored with the *configuration tool*.

2.1.14 LMNGEN_S: Output PID Step Controller

Application

The block is used to structure a PID step controller for actuators with an integral component, (for example motor-driven valves). It contains the manipulated value processing of the controller. The step controller can operate both with and without a position feedback signal.

Block Diagram

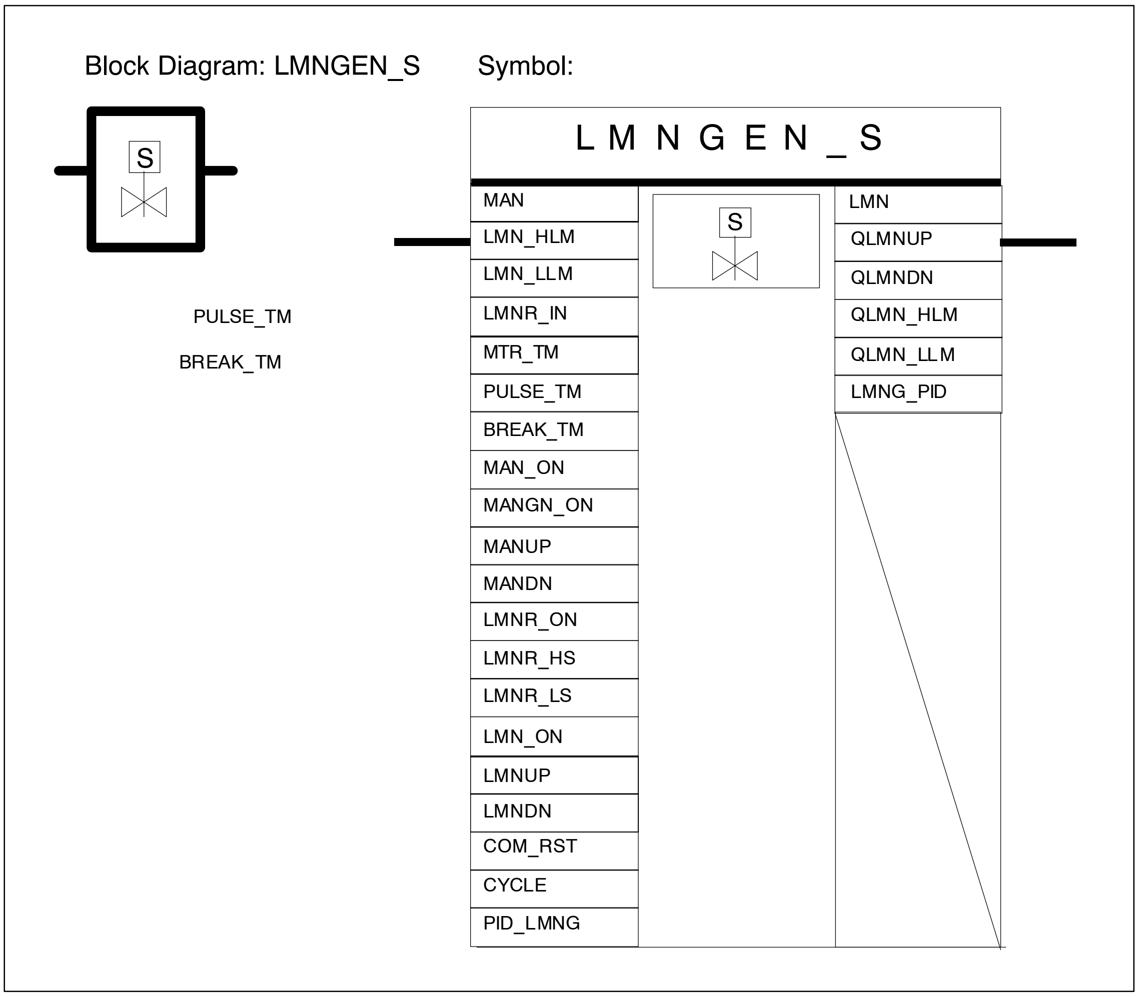


Figure 2-33 LMNGEN_S, Block Diagram and Symbol

Functional Description

If a position feedback signal is available, the block can be used as a positioning controller. With the manual/automatic option, you can change over between the manipulated variable supplied by the PID algorithm and a manual value. You can enter a manual value as an absolute value or use the manual value generator to increase and decrease the manual value. From the difference between the manipulated variable and the position feedback signal, the block generates the pulses for controlling the actuator via a three-step element and the pulse generator. By adapting the response threshold of the three-step element, the switching frequency of the controller is reduced.

The block also functions without a position feedback signal. The I action of the PID algorithm and the position feedback signal are calculated in an integrator and compared with the remaining PD actions. The difference is then applied to the three-step element and the pulse generator that generates the pulses for the final control element. By adapting the response threshold of the three-step element, the switching frequency of the controller is reduced.

The block is always used in conjunction with the PID algorithm block.

- **PID Step Controller: PID + LMNGEN_S**

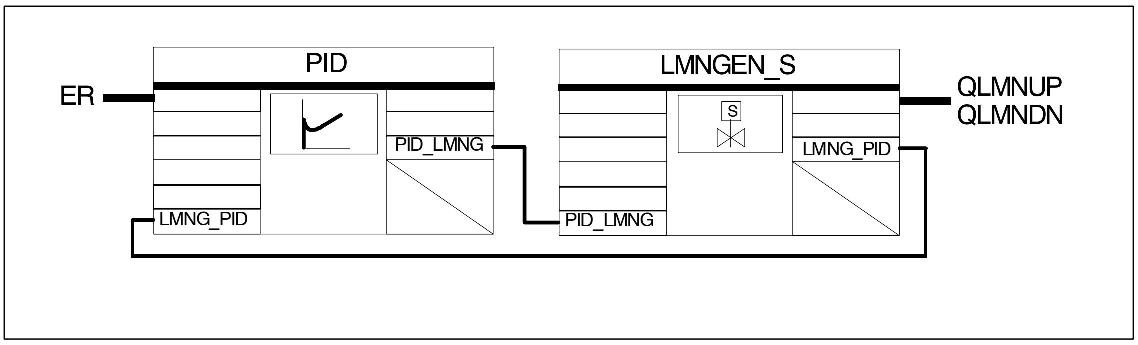


Figure 2-34 Connection of the PID Step Controller

While the PID algorithm is located in a cyclic interrupt priority class, with a cycle time adapted to the dominant system time constant, the LMNGEN_S block, that influences the actuator, can be located in a faster cyclic interrupt priority class to allow manual interventions. The block is connected using the structured input-output parameters PID_LMNG and LMNG_PID.

Input Parameters

The following table shows the data type and structure of the input parameters of LMNGEN_S.

Table 2-31 Input Parameters of LMNGEN_S

Data Type	Parameter	Comment	Permitted Values	Default
REAL	MAN	manual value	technical range of values	0.0
REAL	LMN_HLM	manipulated value high limit	tech. range > LMN_LLM	100.0
REAL	LMN_LLM	manipulated value low limit	tech. range < LMN_HLM	0.0
REAL	LMNR_IN	position feedback signal	technical range of values	0.0
TIME	MTR_TM	motor actuating time	≥ CYCLE	T#30s
TIME	PULSE_TM	minimum pulse time	≥ CYCLE	T#3s
TIME	BREAK_TM	minimum break time	≥ CYCLE	T#3s
BOOL	MAN_ON	manual value on		TRUE
BOOL	MANGN_ON	manual value generator on		FALSE
BOOL	MANUP	manual value up		FALSE
BOOL	MANDN	manual value down		FALSE
BOOL	LMNR_ON	position feedback signal on		FALSE
BOOL	LMNR_HS	high limit signal of position feedback signal		FALSE
BOOL	LMNR_LS	low limit signal of position feedback signal		FALSE
BOOL	LMNS_ON	manipulated value signals on		FALSE
BOOL	LMNUP	manipulated value signal up		FALSE
BOOL	LMNDN	manipulated value signal down		FALSE
BOOL	COM_RST	complete restart		FALSE
TIME	CYCLE	sampling time	≥ 1ms	T#1s
STRUC	PID_LMNG	PID-LMNGEN interface		

Output Parameters

The following table shows the data type and structure of the output parameters LMNGEN_S.

Table 2-32 Output Parameters of LMNGEN_S

Data Type	Parameter	Comment	Default
REAL	LMN	manipulated value	0.0
BOOL	QLMNUP	manipulated value signal up	FALSE
BOOL	QLMNDN	manipulated value signal down	FALSE
BOOL	QLMN_HLM	high limit of manipulated value reached	FALSE
BOOL	QLMN_LLM	low limit of manipulated value reached	FALSE
STRUC	LMNG_PID	PID-LMNGEN interface	

Complete Restart

During a complete restart, all signal outputs are set to zero.

Normal Operation

Apart from normal operation, the block also has the following modes:

Modes		LMNR_ON	LMN_ON	MAN_ON
Step controller with position feedback signal	Automatic mode	TRUE	FALSE	FALSE
	Manipulated value manual mode	TRUE	FALSE	TRUE
	Manual mode manipulated value output signals	TRUE	TRUE	any
Step controller without position feedback signal	Automatic mode	FALSE	FALSE	FALSE
	Manual mode manipulated value output signals	FALSE	TRUE	any

In the mode “step controller without position feedback signal”, the controller has no information about the value of the valve position, it is therefore point-
less to specify a manual value. In step controllers without position feedback signal, there is no “manipulated value manual mode”.

Warning

If there are no limit position signals, the controller cannot recognize a valve limit stop and it is, for example, possible that the controller outputs signals to open the valve although it is already fully open.

If there are no limit position signals, the inputs LMNR_HS and LMNR_LS must be set to FALSE.

Step Controller with Position Feedback Signal

If a position feedback signal is available, LMNR_ON = TRUE must be set. The block then operates as a positioning controller.

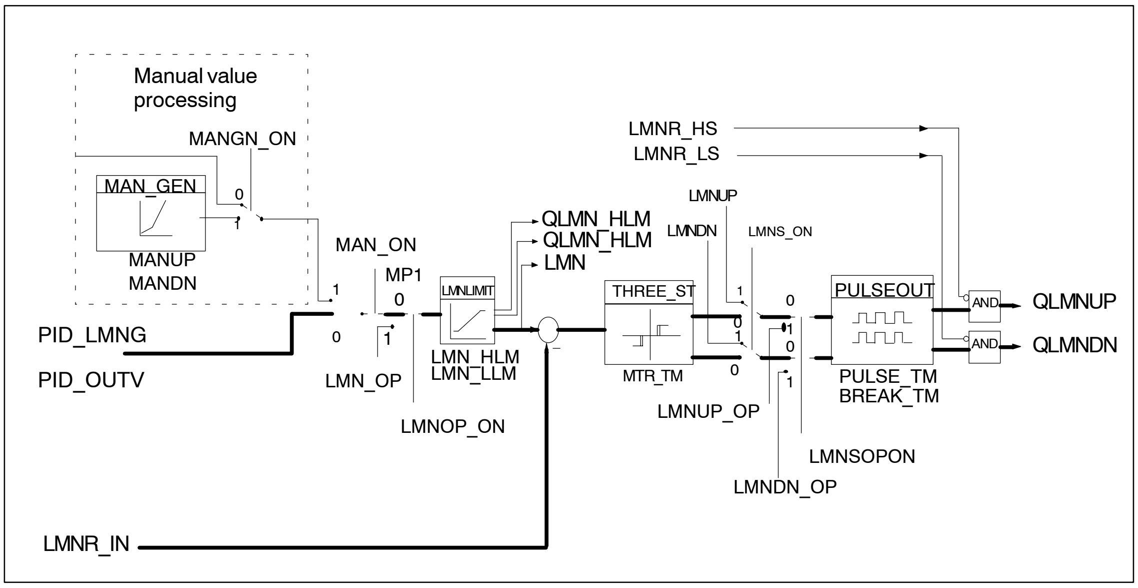


Figure 2-35 Step Controller with Position Feedback Signal

- **Manipulated Value Manual Mode**

In the positioning controller mode, you can switch over to the manual mode with the switch MAN_ON and enter a manipulated value as an absolute value at input MAN or use the manual value generator MAN_GEN.

If MANGN_ON = TRUE is set, the manipulated value can be increased or decreased starting from its current value using the switches MANUP and MANDN within the limits LMN_HLM and LMN_LLM. The rate of change depends on the limits as follows:

During the first 3 seconds after setting MANUP or MANDN:

$$dLMN/dt = (LMN\_HLM - LMN\_LLM) / 100\ s$$

then: $$dLMN/dt = (LMN\_HLM - LMN\_LLM) / 10\ s$$

If MANGN_ON = FALSE and MAN_ON = TRUE is set, the input value MAN is switched through as the manipulated value.

- **Manual Mode of the Manipulated Value Output Signals**

If LMNS_ON = TRUE is set, the binary output signals can be influenced directly. The actuating signal outputs QLMNUP and QLMNDN are set with the switches LMNUP and LMNDN. The minimum pulse time PULSE_TM and minimum break time BREAK_TM are taken into account. If one of the limit position switches LMNR_HS or LMNR_LS is set, the corresponding output signal QLMNUP or QLMNDN is disabled.

- **Automatic Mode**

The manipulated value from the PID algorithm PID_LMNG.PID_OUTV is limited to the selectable values LMN_HLM and LMN_LLM. The difference between the manipulated value and position feedback signal is switched to a three-step element with hysteresis. The off threshold is calculated from the motor actuating time. To reduce switching frequency, the on threshold is adapted. The pulse generator PULSEOUT ensures that the minimum pulse and break times are kept to. If one of the limit position switches LMNR_HS or LMNR_LS is set, the corresponding output signal QLMNUP or QLMNDN is disabled.

Step Controller without Position Feedback Signal

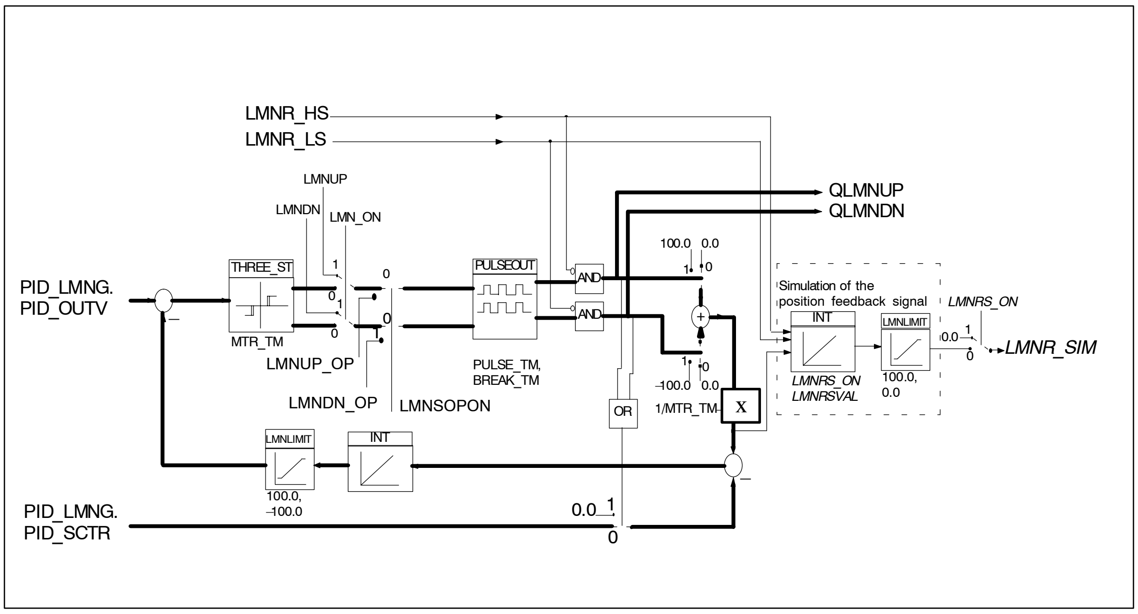


Figure 2-36 Step Controller without Position Feedback Signal

If no position feedback signal is available, LMNR_ON = FALSE is set.

- **Manual Mode for Manipulated Value Output Signals**

If LMNS_ON = TRUE is set, the binary output signals can be influenced directly. The actuating signal outputs QLMNUP and QLMNDN are set with the switches LMNUP and LMNDN. The minimum pulse time PULSE_TM and minimum break time BREAK_TM are taken into account. If one of the limit position switches LMNR_HS or LMNR_LS is set, the corresponding output signal QLMNUP or QLMNDN is disabled.

- **Automatic Mode**

The difference of the I action of the PID algorithm and an internal position feedback signal is integrated in the integrator INT. The input of the integrator is the difference between 100.0 / 0.0 /. –100.0 (depending on the state of the output signals QLMNUP and QLMNDN) relative to the motor actuating time MTR_TM and the error signal multiplied by the gain relative to the reset time TI of the PID algorithm (PID_LMNG.PID_SCTR). The output of the integrator forms the feedback signal that is compared with the remaining PD action of the PID algorithm (PID.LMNG.PID_OUTV). The difference is switched to a three-step element with hysteresis. The off threshold is calculated from the motor actuating time.

To reduce switching frequency, the on threshold is adapted. The pulse generator PULSEOUT ensures that the minimum pulse and break times are kept to. If one of the limit position switches LMNR_HS or LMNR_LS is set, the corresponding output signal QLMNUP or QLMNDN is disabled.

Simulation of the Position Feedback Signal

The position feedback signal is simulated by an integrator with the motor actuating time MTR_TM as reset time. Changing the parameter LMNRS_ON from FALSE to TRUE starts the simulation with the initial value LMNRSVAL. If LMNRS_ON is set to FALSE, then the initial value LMNRSVAL is output for the parameter LMNR_SIM. The parameters LMNRS_ON, LMNRSVAL and LMNR_SIM are located in the static variable area. They are supplied with values by the *configuration tool*. If LMNR_HS = TRUE, the upwards integration is disabled. If LMNR_LS = TRUE, the downwards integration is disabled.

Warning

The position feedback signal is only simulated. It does not necessarily correspond to the real position feedback signal of the final control element.

The PID process identification function of the *configuration tool* requires the position feedback signal as an input variable.

If a real position feedback signal exists, it should be used.

Block-Internal Limits

No values are limited internally in the block; the parameters are not checked.

Influencing the Manipulated Value with the *configuration tool*

LMN Display and Setting in the Loop Monitor

Die configuration tool has its own interface to the LMNGEN_S block. It is therefore always possible to interrupt the manipulated variable branch when working on a programming device or PC and to set your own manipulated values (Figure 2-37).

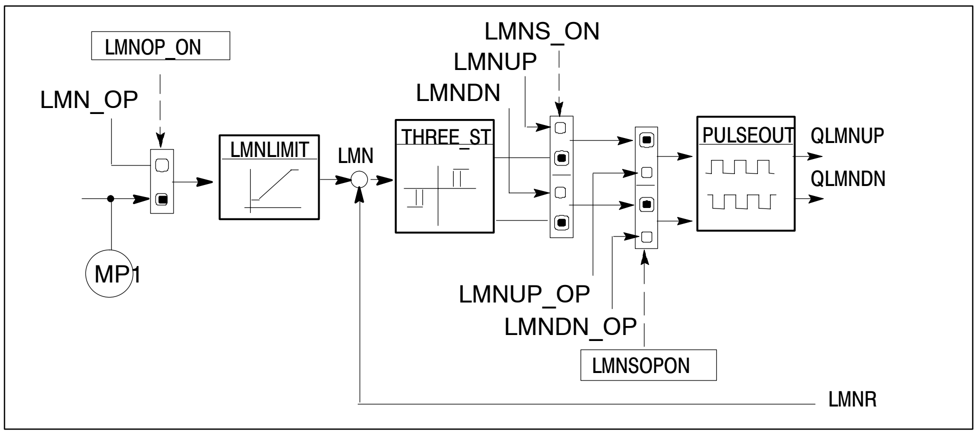


Figure 2-37 Interventions in the Manipulated Value Branch Using the *configuration tool*

The loop monitor has a field labeled “manipulated value”. Here, in the upper field (”Controller:”), the manipulated value currently applied to measuring point MP1 is displayed. In the field below (”PG:”), you can set the parameter LMN_OP.

Switching over to Manual Manipulated Value with the *configuration tool*

If the switch in the *configuration tool* is set to “PG: ”, the switching signal of the structure switch LMNOP_ON is set to TRUE in the controller FB and LMN_OP is switched through to the manipulated value LMN.

If the rate of change limit LMN_ROC is activated in the manipulated variable branch , it is possible to switch between “PG: ” and “Controller: ” without causing any sudden change. The value to which the manipulated variable switches back (MP1) can be seen in the “Controller: ” display field of the loop monitor. LMN then returns to this value at the rate of change set at LMN_ROC.

If the switch “Controller:/PG: ” in the manipulated values field is set to “PG: ”, the parameter LMNSOPON = TRUE is set and the actuating signal outputs can be controlled with the parameters LMNUP_OP (up) or LMNDN_OP (down) in the loop monitor. This applies to a step controller with or without position feedback signal.These interventions only affect the process after you transfer them to the programmable controller by clicking the “Send” button in the loop monitor.The parameters LMNOP_ON, LMN_OP, MP1, LMNSOPON, LMNUP_OP and LMNDN_OP are static variables and are not available as input/output parameters for the block. The parameters should not be connected and should only be used and monitored with the *configuration tool*.

2.1.15 LP_SCHED: Loop Scheduler

Application

When calling a large number of control loops with different sampling times, and with control loops with large sampling times, the range of the priority classes for cyclic interrupts is not adequate. Using the loop scheduler LP_SCHED, several control loops with different sampling times can be called at equal intervals in one cyclic interrupt priority class.

The use of the loop scheduler is not obligatory. The controller FBs can also be called directly by the OB without the scheduler function.

Block Diagram

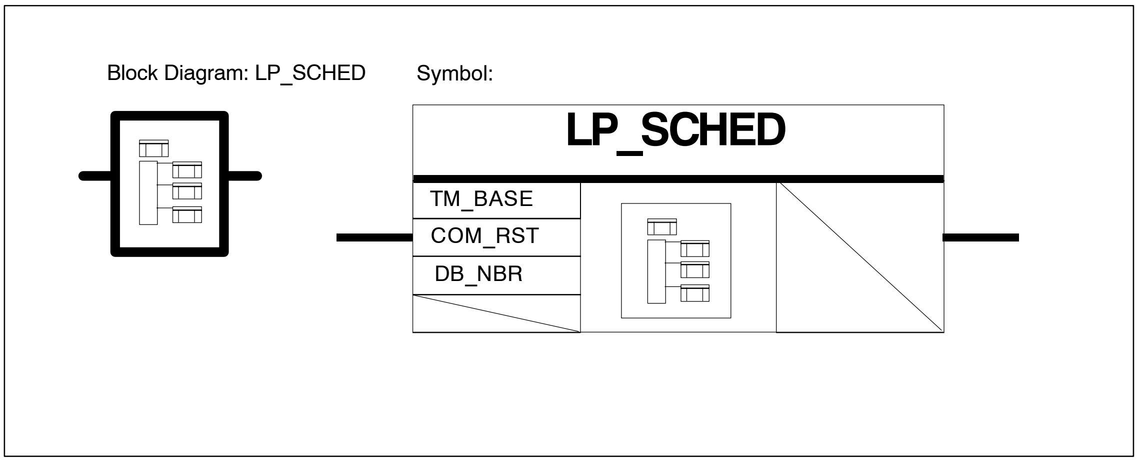
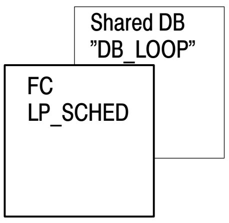


Figure 2-38 LP_SCHED, Block Diagram and Symbol

Functional Description

The scheduling of several controllers in one cyclic interrupt priority class is implemented in the LP_SCHED function. The block must be called before all the control loops. The data for the individual block calls are stored in a shared data block (DB_LOOP).



The LP_SCHED block processes the shared data block and sets the ENABLE bits in keeping with the sampling times of the control loops. It supports the time base of the cyclic interrupt priority class and calls the control loops according to their different sampling times. The individual loops in this priority class are called and processed according to their sampling time. After the block call, the ENABLE bit must be reset. The block calls and the resetting of the ENABLE bits must be programmed.

The calls for individual control loops can be disabled manually. Individual loops can also be reset (complete restart).

Note

The block does not check whether or not a shared DB with the number DB_NBR exists or whether the “highest loop number” parameter GLP_NBR fits in the DB length. If the parameter assignment is incorrect, the CPU changes to STOP with an internal system error.

Input Parameters

The following table shows the data type and structure of the input parameters of LP_SCHED.

Table 2-33 Input Parameters of LP_SCHED

Data Type	Parameter	Comment	Permitted Values
TIME	TM_BASE	time base	>=1ms
BOOL	COM_RST	complete restart	
BLOCK_DB	DB_NBR	data block number	

Output Parameters

The following table shows the data type and structure of the output parameters LP_SCHED.

Table 2-34

Data Type	Parameter	Comment
The block does not have any output parameters.		

Neustart

Bei COM_RST = TRUE werden folgende Vorbelegungen eingestellt:

Aktuelle Regelkreisnummer: ALP_NBR =	*0*

*Die Aufrufdaten aller Regelkreise bei **GLP_NBR** werden wie folgt vorbelegt:*

Freigabe:	*ENABLE =*	*NOT MAN_DIS*
Abtastzeit:	*CYCLE =*	*GV(MAN_CYC)*
Neustart:	*COM_RST =*	*TRUE*
Lokale Aufrufnummer:	*ILP_COU =*	*0*

*GV(MAN_CYC) = MAN_CYC wird gerundet auf ein ganzzahliges Vielfaches von TM_BASE*GLP_NBR*

Shared Data Block “DB_LOOP”

The following table shows the shared data block DB_LOOP for the controller calls.

Table 2-35

Data Type	Parameter	Comment	Permitted Values	Default
INT	GLP_NBR	greatest loop number	1 – 256	2
INT	ALP_NBR	actual loop number	0 – 256	0
TIME	LOOP_DAT[1]. MAN_CYC	loop data [1] manual sampling time	>=1ms	T#1s
BOOL	LOOP_DAT[1]. MAN_DIS	loop data [1] manual loop disable		FALSE
BOOL	LOOP_DAT[1]. MAN_CRST	loop data [1] manual complete restart		FALSE
BOOL	LOOP_DAT[1]. ENABLE	loop data [1] enable loop		FALSE
BOOL	LOOP_DAT[1]. COM_RST	loop data [1] complete restart		FALSE
INT	LOOP_DAT[1]. ILP_COU	loop data [1] internal loop counter		0
TIME	LOOP_DAT[1]. CYCLE	loop data [1] sampling time	>=1ms	T#1s

Calling a Control Loop Using LP_SCHED

The LP_SCHED function is integrated in the call strategy of the CPU using three input parameters. The time base of the cyclic interrupt class is specified at the TM_BASE input. The control loops are called using a conditional block call in **one** cyclic interrupt class (for example OB35) and the ENABLE bits are queried in the shared data block.

If the call is made at the complete restart level, the input COM_RST = TRUE is set. This call must be reset to FALSE in the cyclic interrupt class. The shared data block (Table 2-35) with the time data for the control loops in the cyclic interrupt class is assigned using the input parameter DB_NBR.

Programming a Control Loop Call (Shared DB)

The loop scheduler must be programmed <u>without</u> the support of the *configuration tool*.

The data block (DB_LOOP) contains both a parameter for the total number of control loops to be processed in the cyclic interrupt class (a maximum of 256) and a parameter to indicate the control loop currently being processed, as follows:

GLP_NBR	Highest control loop number
ALP_NBR	Number of the control loop being processed in the cycle

The number of each control loop is decided by the position of its call in the sequence of entries in the DB.

The call data of the individual control loops are in structured form in the LOOP_DAT field. If you want to add control loops, the field length of LOOP_DAT must be adapted by modifying the ARRAY data type in the declaration view. With, for example 10 loops, you would enter ARRAY[1..10]. You must also adapt the parameter GLP_NBR in the data view. It must never be higher than the field length.

The parameters COM_RST and CYCLE must be linked to the corresponding input parameters of the FB belonging to the called control loop. This connection must be programmed by the user. If the ENABLE parameter is set, the corresponding control loop will be called. After the controller has been called, the ENABLE bit must be reset. The user must program the conditional controller call and the ENABLE bit reset.

Using the parameters MAN_CYC / MAN_DIS / MAN_CRST that can be set manually, you can decide whether or not a control loop is called. You can change these calls online (in other words during operation) as long as you only overwrite parameters and do not regenerate the entire DB. The meaning of the parameters is as follows:

MAN_CYC	Sampling time of the corresponding controller (rounded up to a whole multiple of TM_BASE * GLP_NBR in CYCLE).
MAN_DIS	Disable the controller call.
MAN_CRST	Complete restart for the controller.

Processing Calls

Each loop is processed according to the parameters set in the DB if the ENABLE signal of the controller call data is set.

The data block is worked through from top to bottom. Per cycle, the loop scheduler moves on one loop number (ALP_NBR) further in the order in which they are entered in the DB. The internal counter ILP_COU is then decremented by one. If ILP_COU = 0 is set, the loop scheduler sets the ENABLE bit of the corresponding loop. The ENABLE bit must be reset after the call and this must be programmed by the user.

When CYCLE is processed, the parameter MAN_CYC is transferred:

CYCLE = GV (MAN_CYC), GV = whole multiple

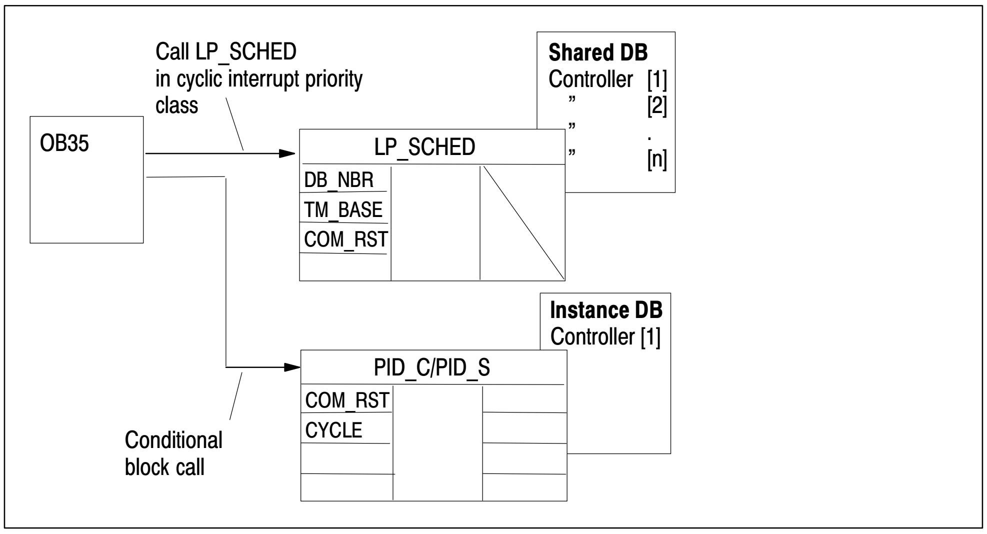


Figure 2-39 Principle of the Controller Call Using the Loop Scheduler LP_SCHED

- Disabling selected loops:

  If you set the “MAN_DIS” bit in the DB, the ENABLE bit is reset to FALSE and the loop involved is excluded from processing in the loop scheduler.

- Resetting selected loop (complete restart):

  If you set the “MAN_CRST” bit in the DB, COM_RST = TRUE is set and then MAN_CRST is reset. The complete restart routine of the control loop is then processed. In the following call cycle, COM_RST is then set to FALSE.

Note

If you want to insert or delete a control loop and regenerate the entire DB without a complete restart of the loop scheduler, the internal loop counters (ILP_COU[n]) and the parameter for the current loop number ALP_NBR must be set to zero.

Conditions for Calling a Loop With LP_SCHED

To ensure that the intervals between the calls of a particular controller remain constant and to spread the load on the CPU, only one control loop can be processed per time base unit of the cyclic interrupt class. When you assign the sampling times MAN_CYC, remember the following conditions with respect to the time base (TM_BASE):

- The processing times of the individual loops must be shorter than the time base (TM_BASE) of the cyclic interrupt class.
- The sampling time of a control loop (MAN_CYC) must be a whole multiple (GV) of the product of the time base and number of controllers to be processed (GLP_NBR):

  MAN_CYC = GV (TM_BASE $*$ GLP_NBR).

Example of Loop Scheduling

The following example illustrates the sequence of calls of four loops in one cyclic interrupt class (Figure 2-40). Only one loop is processed per time base unit. The sequence in which the loops are called and with it the time displacement (TD1 to TD5) result from the sequence of the call data within the shared data block.

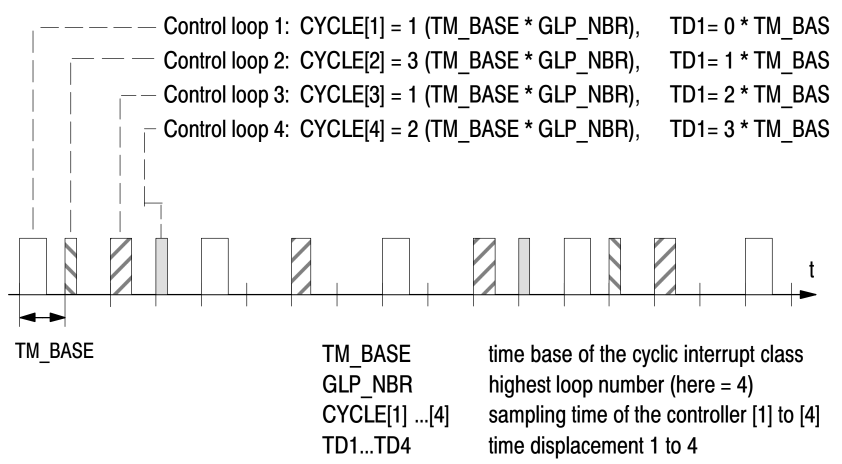

Control loop 1: CYCLE[1] = 1 (TM_BASE * GLP_NBR), TD1= 0 * TM_BAS
Control loop 2: CYCLE[2] = 3 (TM_BASE * GLP_NBR), TD1= 1 * TM_BAS
Control loop 3: CYCLE[3] = 1 (TM_BASE * GLP_NBR), TD1= 2 * TM_BAS
Control loop 4: CYCLE[4] = 2 (TM_BASE * GLP_NBR), TD1= 3 * TM_BAS
t
TM_BASE
TM_BASE time base of the cyclic interrupt class
GLP_NBR highest loop number (here = 4)
CYCLE[1] ...[4] sampling time of the controller [1] to [4]
TD1...TD4 time displacement 1 to 4

Figure 2-40 Call Sequence of Four Loops Called at Different Intervals

Block-Internal Limits

The values of the parameters are not limited in the block; the parameters are not checked.

2.1.16 NONLIN: Non-Linear Static Function

Application

With NONLIN, the input value, for example a measured value from a thermoelement, can be adapted using a selectable function.

Block Diagram

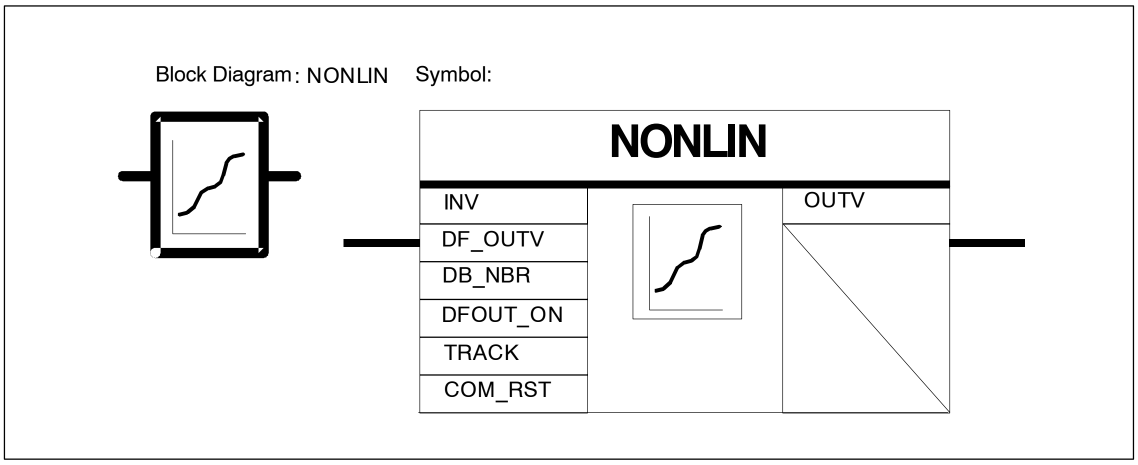


Figure 2-41 NONLIN, Block Diagram and Symbol

Functional Description

The points of the function are stored in a shared data block. NONLIN assigns the corresponding output value to the current input value from the function. If the input value is lower than the value at point 0, the value is extrapolated with the slope between points 0 and 1. If the value is higher than the value of the last point, *so wird mit der Steigerung des letzten Stützpunktpaares extrapoliert*. For the block to supply a feasible result, the values of the points must have a strictly monotonous rising slope.

Using control inputs, a selected value or the input variable itself can be output directly as the output variable.

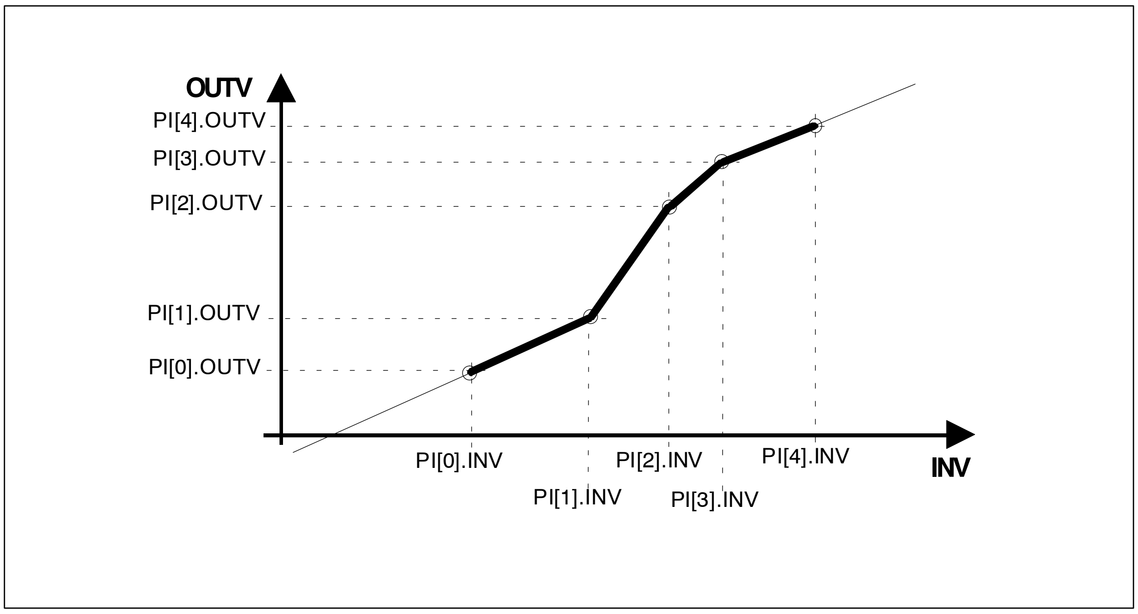


Figure 2-42 OUTV = f(INV)

Note

The block does not check whether a shared DB with the number DB_NBR really exists, or whether the parameter DB_NBR.NBR_PTS (number of points) matches the length of the data block. If the parameters are set incorrectly, the CPU changes to STOP with an internal system error.

Input Parameters

The following table shows the data type and structure of the input parameters of NONLIN.

Table 2-36 Input Parameters of NONLIN

Data Type	Parameter	Comment	Permitted Values	Default
REAL	INV	input variable	technical range of values	0.0
REAL	DF_OUTV	default output variable	technical range of values	0.0
BLOCK_DB	DB_NBR	data block number		DB 1
BOOL	DFOUT_ON	default output variable on		FALSE
BOOL	TRACK	tracking OUTV=INV		FALSE
BOOL	COM_RST	complete restart		FALSE

Output Parameters

The following table shows the data type and structure of the output parameters NONLIN.

Table 2-37 Output Parameters of NONLIN

Data Type	Parameter	Comment	Default
REAL	OUTV	output variable	0.0

Shared Data Block DB_NBR

The following table shows the data type and Parameter von **DB_NBR (**default with 5 curve points**)**.

Data Type	Parameter	Comment	Permitted Values	Default
INT	DB_NBR.NBR_PTS	highest coordinate	1 – 255	4
REAL	DB_NBR.PI[0].OUTV	output variable [0], point 0	technical range of values	0.0
REAL	DB_NBR.PI[0].INV	input variable [0], point 0	technical range of values	0.0
REAL	DB_NBR.PI[1].OUTV	output variable [1], point 1	technical range of values	0.0
REAL	DB_NBR.PI[1].INV	input variable [1], point 1	technical range of values	0.0
REAL	DB_NBR.PI[2].OUTV	output variable [2], point 2	technical range of values	0.0
REAL	DB_NBR.PI[2].INV	input variable [2], point 2	technical range of values	0.0
REAL	DB_NBR.PI[3].INV	input variable [3], point 3	technical range of values	0.0
REAL	DB_NBR.PI[3].OUTV	output variable [3], point 3	technical range of values	0.0
REAL	DB_NBR.PI[4].OUTV	output variable [4], point 4	technical range of values	0.0
REAL	DB_NBR.PI[4].INV	input variable [4], point 4	technical range of values	0.0

Complete Restart

During a complete restart OUTV = 0.0 is output.

If DFOUT_ON=TRUE then DF_OUTV is output.

Normal Operation

In normal operation, *wird durch Interpolation am aktuellen Stützpunktpaar der Ausgangswert berechnet.*

DFOUT_ON and TRACK have the following influence on OUTV:

Modes	DFOUT_ON	TRACK	OUTV
Default output variable	TRUE	any	DF_OUTV
Track	FALSE	TRUE	OUTV = INV
Normal operation	FALSE	FALSE	OUTV = f(INV)

- **Default Output Variable**

If DFOUT_ON = TRUE is set, DF_OUTV is output, the change at OUTV is a step change. At the changeover to DFOUT_ON = FALSE , the change at OUTV is also a step change.

- **Tracking**

If TRACK=TRUE is set, the input value is output directly (OUTV=INV). The change as with DFOUT_ON is a step change. TRACK has lower priority than DFOUT_ON.

Block-Internal Limits

The values of the parameters are not limited in the block; the parameters are not checked.

2.1.17 NORM: Physical Normalization

Application

The value of a process variable supplied by a sensor is often in a range that is not particularly suitable for the user (for example 0 to 10 V correspond to 0 to 1200 °C or 0 to 10 V correspond to 0 to 3000 rpm). By adapting the setpoint or process variable, both variables can have the same range.

Block Diagram

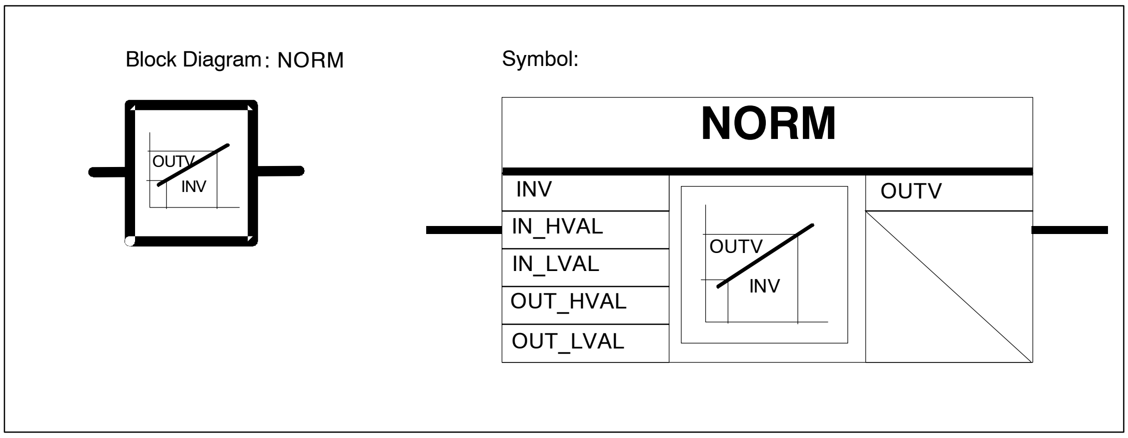


Figure 2-43 NORM, Block Diagram and Symbol

Functional Description

The block normalizes the input variable to form an output variable with a different range of values. The normalization curve is defined by two points.

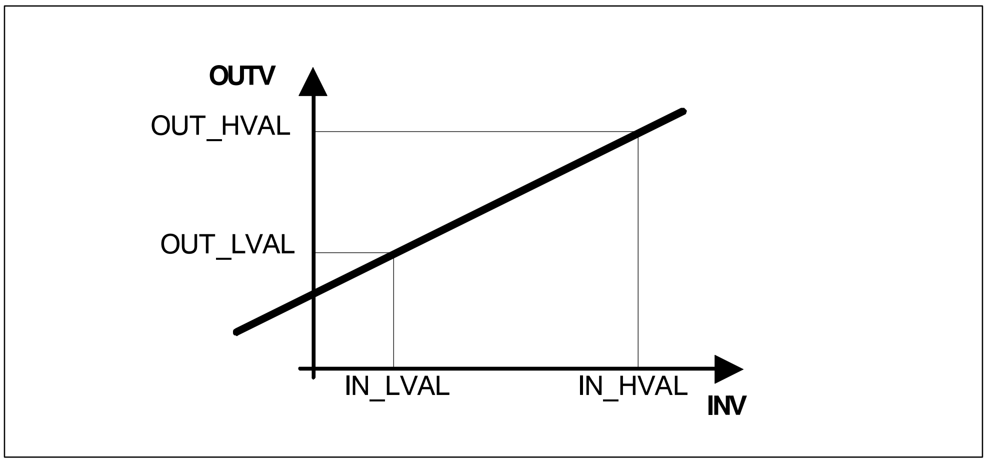


Figure 2-44 Normalization Curve

Input Parameters

The following table shows the data type and structure of the input parameters of NORM.

Table 2-38 Input Parameters of NORM

Data Type	Parameter	Comment	Permitted Values	Default
REAL	INV	input variable	technical range of values	0.0
REAL	IN_HVAL	physical input value high	tech. range > IN_LVAL	100.0
REAL	OUT_HVAL	physical output value high	tech. range > OUT_LVAL	100.0
REAL	IN_LVAL	physical input value low	tech. range < IN_HVAL	0.0
REAL	OUT_LVAL	physical output value low	tech. range < OUT_HVAL	0.0

Output Parameters

The following table shows the data type and structure of the output parameters NORM.

Table 2-39 Output Parameters of NORM

Data Type	Parameter	Comment	Default
REAL	OUTV	output variable	0.0

Complete Restart

The block has no complete restart routine.

Normal Operation

The input variable INV is converted to the output variable OUTV using the normalization curve. The normalization curve is determined by the points IN_HVAL, IN_LVAL, OUT_HVAL and OUT_LVAL.

Block-Internal Limits

The values of the parameters are not limited in the block; the parameters are not checked.

2.1.18 OVERRIDE: Override Control

Application

The block is required to implement an override control.

Block Diagram

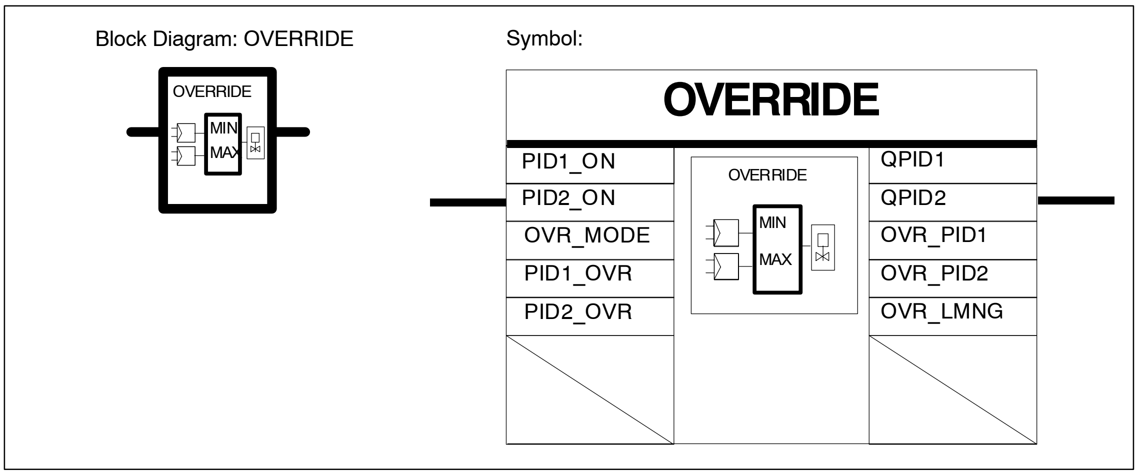


Figure 2-45 OVERRIDE, Block Diagram and Symbol

Functional Description

Two PID controllers are connected to one actuator. The maximum (OVR_MODE = FALSE) or minimum (OVR_MODE = TRUE) of the two PID controllers is applied to the actuator. To do this, 2 PID blocks are connected via the OVERRIDE block with one of the manipulated value blocks LMNGEN_C or LMNGEN_S.

The truth table of the switches PID1_ON, PID2_ON and OVR_MOD is shown below.

PID1_ON	PID2_ON	OVR_MOD	Function
1	0	▨	only PID1 algorithm is effective
0	1	▨	only PID2 algorithm is effective
0	0	0	the maximum of PID1 and PID2 is applied
1	1	0	the maximum of PID1 and PID2 is applied
0	0	1	the minimum of PID1 and PID2 is applied
1	1	1	the minimum of PID1 and PID2 is applied

▨ The function is processed regardless of this signal state.

Which PID algorithm is active is indicated at the outputs QPID1 and QPID2.

Input Parameters

The following table shows the data type and structure of the input parameters of OVERRIDE.

Table 2-40 Input Parameters of OVERRIDE

Data Type	Parameter	Comment	Permitted Values	Default
BOOL	PID1_ON	PID controller 1 on		FALSE
BOOL	PID2_ON	PID controller 2 on		FALSE
BOOL	OVR_MODE	override mode FALSE = maximum, TRUE = minimum		FALSE
STRUC	PID1_OVR	PID-LMNGEN interface		
STRUC	PID2_OVR	PID-LMNGEN interface		

Output Parameters

The following table shows the data type and structure of the output parameters OVERRIDE.

Table 2-41 Output Parameters of OVERRIDE

Data Type	Parameter	Comment	Default
BOOL	QPID1	PID controller 1active	
BOOL	QPID2	PID controller 2active	
STRUC	OVR_LMNG	PID-LMNGEN interface	

Complete Restart

The block has no complete restart routine.

Normal Operation

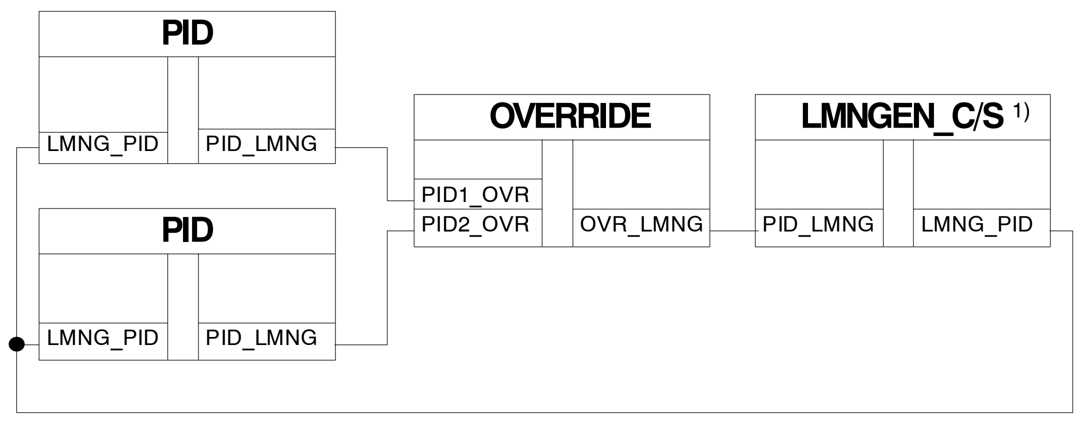
The block has no modes other than normal operation.

Block-Internal Limits

The values of the parameters are not limited in the block; the parameters are not checked.

Example

Figure 2-46 shows an example of connecting an override control.



1) Step controller only with position feedback LMNR_ON = TRUE

Figure 2-46 Example of Connecting an Override Control

2.1.19 PARA_CTL: Parameter Control

Application

The block is used in controller structures with parameter changeovers when optimum parameters are required for different operating ranges.

Block Diagram

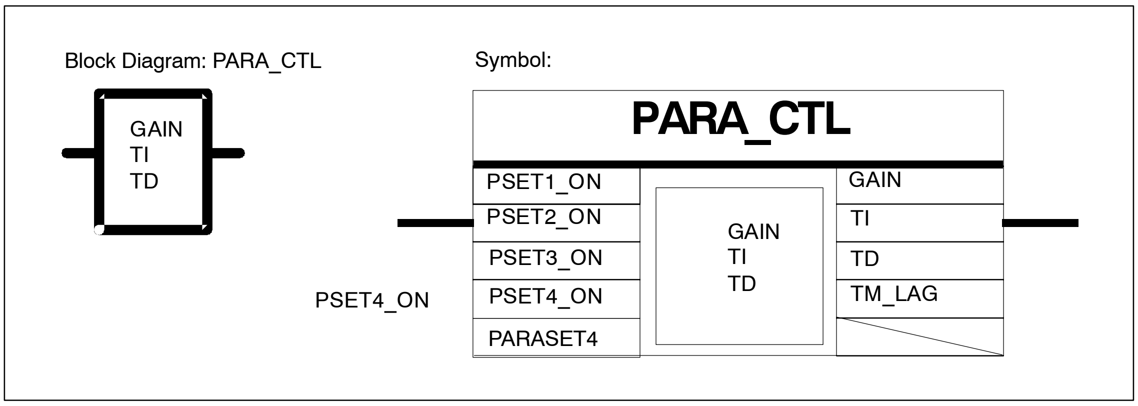


Figure 2-47 PARA_CTL, Block Diagram and Symbol

Functional Description

Several controller parameters sets (GAIN, TI, TD and TM_LAG) can be transferred to one PID controller.

With the switches PSET1_ON ... PSET4_ON, one of the four sets of parameters can be applied to the outputs GAIN, TI, TD and TM_LAG.

```
if   PSET1_ON = 1   then   GAIN = PARASET1.GAIN
                           TI = PARASET1.TR
                           TD = PARASET1.TD
                           TM_LAG = PARASET1.TM_LAG
if   PSET2_ON = 1   then   GAIN = PARASET2.GAIN
                           TI = PARASET2.TR
                           TD = PARASET2.TD
                           TM_LAG = PARASET2.TM_LAG
if   PSET3_ON = 1   then   GAIN = PARASET3.GAIN
                           TI = PARASET3.TR
                           TD = PARASET3.TD
                           TM_LAG = PARASET3.TM_LAG
```

```
if    PSET4_ON = 1    then    GAIN = PARASET4.GAIN
                              TI = PARASET4.TR
                              TD = PARASET4.TD
                              TM_LAG = PARASET4.TM_LAG
```

If 2 or more switches are set, the switch with the lowest parameter number has the highest priority. If no switch is set, the first set of parameters is transferred.

Input Parameters

The following table shows the data type and structure of the input parameters of PARA_CTL.

Table 2-42 Input Parameters of PARA_CTL

Data Type	Parameter	Comment	Permitted Values	Default
BOOL	PSET1_ON	set number 1		FALSE
BOOL	PSET2_ON	set number 2		FALSE
BOOL	PSET3_ON	set number 3		FALSE
BOOL	PSET4_ON	set number 4		FALSE
STRUCT	PARASET1	parameter set1		
STRUCT	PARASET4	parameterset4		

Output Parameters

The following table shows the data type and structure of the output parameters PARA_CTL.

Table 2-43 Output Parameters of PARA_CTL

Data Type	Parameter	Comment	Default
REAL	GAIN	proportional gain	1.0
TIME	TI	reset time	10s
TIME	TD	derivative time	5s
TIME	TM_LAG	time lag	2s

Controller Parameter Sets

The following table shows the data type and parameters.

Table 2-44

Data Type	Parameter	Comment	Default	Default
REAL	PARASET1.GAIN	proportional gain 1		1.0
TIME	PARASET1.TI	reset time 1		T#10s
TIME	PARASET1.TD	derivative time 1		T#5s
TIME	PARASET1.TM_LAG	time lag 1		T#2s
...	...		...	...
REAL	PARASET4.GAIN	proportional gain 4		1.0
TIME	PARASET4.TI	reset time 4		T#10s
TIME	PARASET4.TD	derivative time 4		T#5s
TIME	PARASET4.TM_LAG	time lag 4		T#2s

Complete Restart

The block has no complete restart routine.

Normal Operation

The block has no modes other than normal operation.

Block-Internal Limits

The values of the parameters are not limited in the block; the parameters are not checked.

Example

If a stepless changeover between parameter sets is required, a rate of change ramp must be included for the GAIN parameter (HROC_LIM) between the PI block and the PARA_CTL block.(see Figure 2-48).

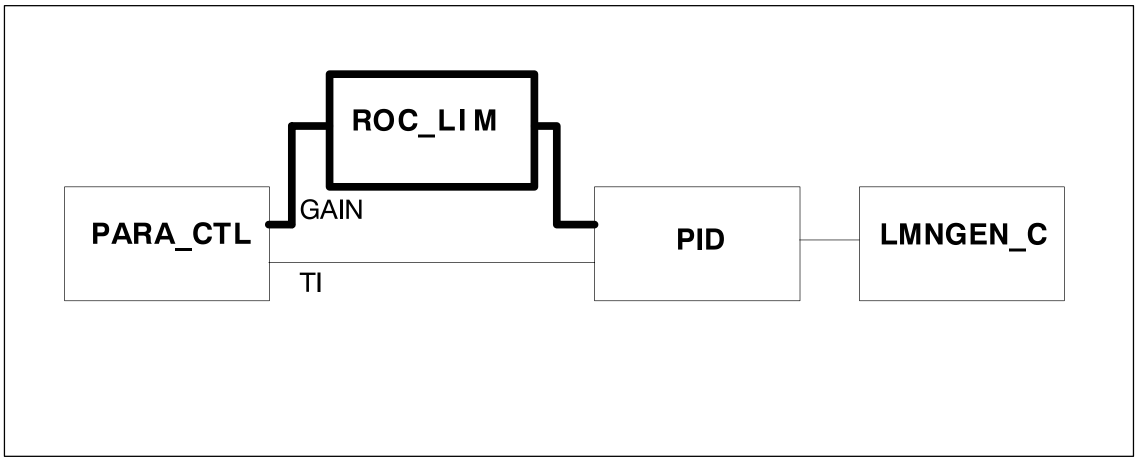


Figure 2-48 Stepless Changeover between Parameter Sets

2.1.20 PID: PID Algorithm

Application

The block contains the PID algorithm for creating the following controller types:

- Continuous PID controller:
  PID + LMNGEN_C
- PID pulse controller for proportional actuators:
  PID + LMNGEN_C + PULSEGEN
- PID step controller for integrating actuators:
  PID + LMNGEN_S

Block Diagram

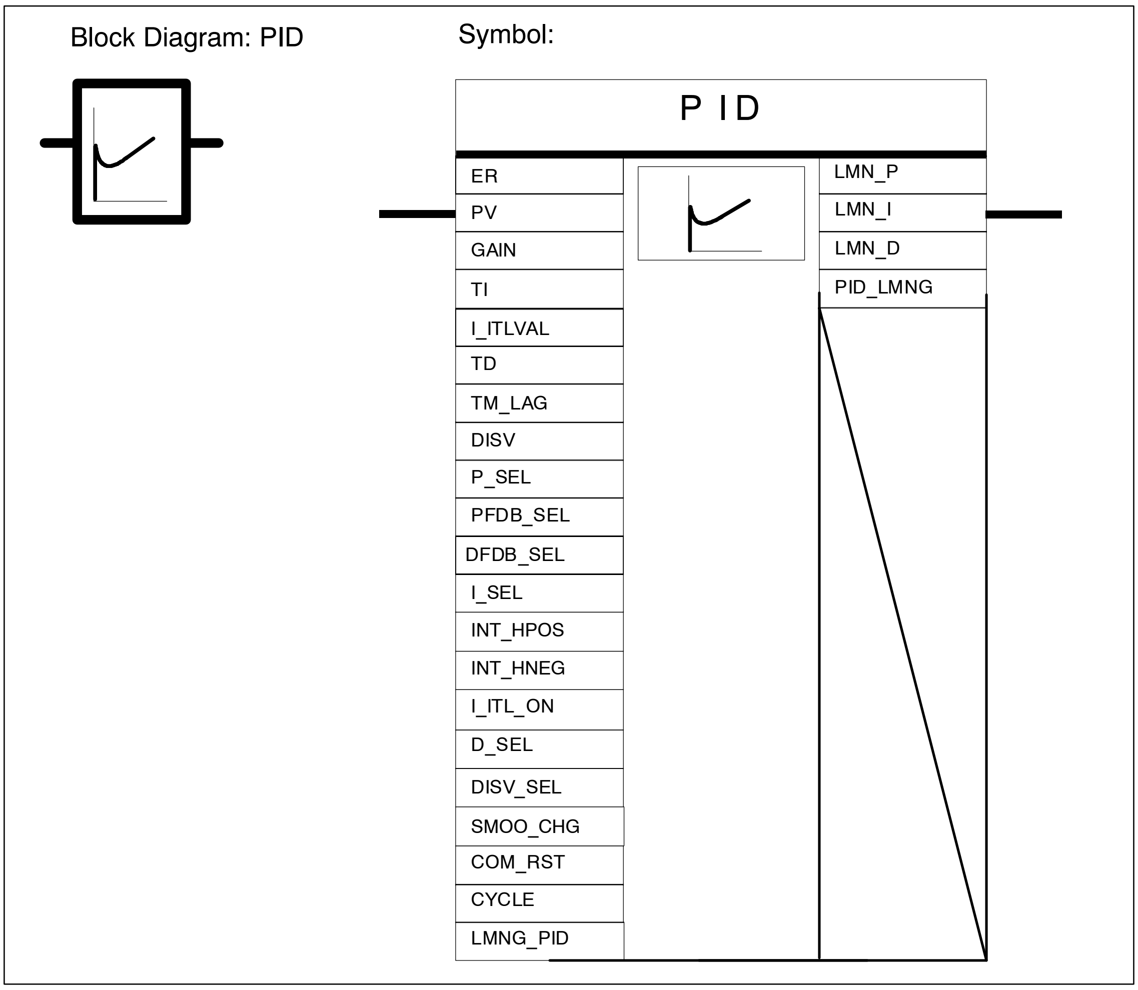


Figure 2-49 PID, Block Diagram and Symbol

Functional Description

The block implements the PID algorithm. It is designed as a purely parallel structure and functions solely as a positioning algorithm. The proportional, integral and derivative actions can be activated or deactivated individually. This allows P, PI, PD and PID controllers to be configured.

The calculation of the P and D actions can be located in the feedback path. By moving the P and D actions to the feedback path, the controller causes no sudden changes if disturbances are compensated at the same speed. It is not normally necessary to use a setpoint integrator to avoid step changes in the setpoint.

While the PID block is located in a cyclic interrupt priority class whose cycle time is adapted to the dominant system time constant, the blocks for processing the actuator signals (LMNGEN_C and LMNGEN_S) can be located in a faster cyclic interrupt priority class.

Input Parameters

The following table shows the data type and structure of the input parameters of PID.

Table 2-45 Input Parameters of PID

Data Type	Parameter	Comment	Permitted Values	Default
REAL	ER	error signal	technical range of values	0.0
REAL	PV	process variable (P- bzw. D-Anteil in Rückführung)	technical range of values	0.0
REAL	GAIN	proportional gain		1.0
TIME	TI	reset time		T#20s
REAL	I_ITLVAL	initialization value of the integral action	technical range of values	0.0
TIME	TD	derivative time		T#10s
TIME	TM_LAG	time lag of the derivative action		T#2s
REAL	DISV	disturbance variable	technical range of values	0.0
BOOL	P_SEL	proportional action on		TRUE
BOOL	PFDB_SEL	proportional action in feedback path on		FALSE
BOOL	DFDB_SEL	derivative action in feedback path on		FALSE
BOOL	I_SEL	integral action on		TRUE
BOOL	INT_HPOS	integral action hold in positive direction		FALSE
BOOL	INT_HNEG	integral action hold in negative direction		FALSE
BOOL	I_ITL_ON	initialization of the integral action		FALSE
BOOL	D_SEL	derivative action on		FALSE
BOOL	DISV_SEL	disturbance variable on		TRUE

Table 2-45 Input Parameters of PID, continued

Data Type	Parameter	Comment	Permitted Values	Default
BOOL	SMOO_CHG	smooth change over from the manual mode to the automatic mode		TRUE
BOOL	COM_RST	complete restart		FALSE
TIME	CYCLE	sampling time	≥ 1ms	T#1s
STRUC	PID_LMNG	PID-LMNGEN interface		

Output Parameters

The following table shows the data type and structure of the output parameters PID.

Table 2-46 Output Parameters of PID

Data Type	Parameter	Comment	Default
REAL	LMN_P	proportional component	0.0
REAL	LMN_I	integral component	0.0
REAL	LMN_D	derivative component	0.0
STRUC	PID_LMNG	PID-LMNGEN interface	

Complete Restart

During a complete restart, all output and in/out parameters are set to zero.

Normal Operation

Apart from normal operation, the block has the following modes:

Modes	P_SEL	I_SEL	D_SEL
P controller	TRUE	TRUE or FALSE	FALSE
PI controller	TRUE	TRUE	FALSE
PD controller	TRUE	FALSE	TRUE
PID controller	TRUE	TRUE	TRUE

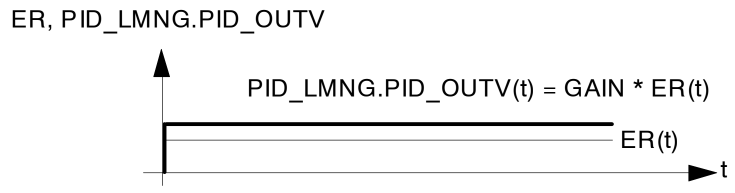
- **P Controller**

Here, the D action is turned off. With the I action, the initialization value I_ITLVAL can be used to specify an operating point (I_SEL = TRUE and I_ITL_ON = TRUE). If the operating point is to be kept to a constant zero, the I action can also be turned off.

Transfer Function

The error signal ER is multiplied by GAIN.

Step Response



where : PID_LMNG.

PID_OUTV(t) :	Manipulated variable automatic mode
ER :	Error signal
GAIN :	Controller gain
t :	Time

Figure 2-50 Step Response of a P Controller

- **PI Controller**

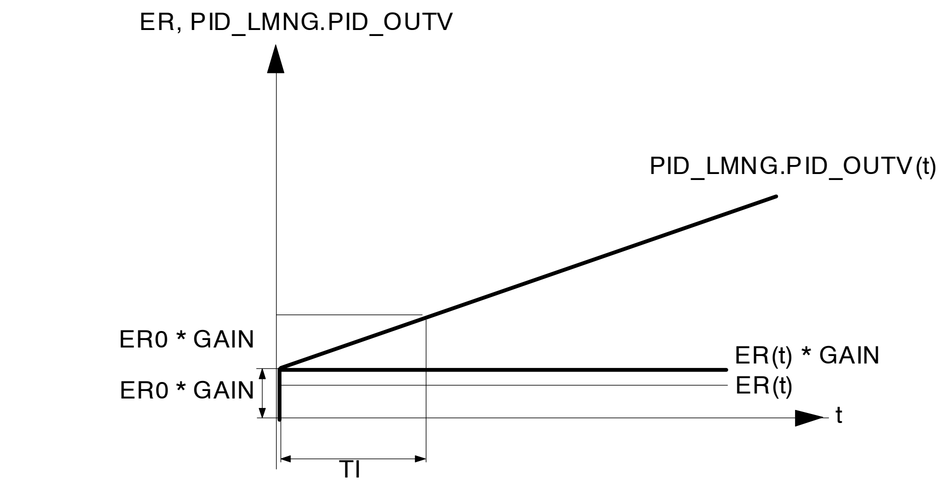
Here, the D action is turned off.

Transfer Function

The error signal ER is multiplied and integrated. The transfer function in the Laplace range is:

PID_LMNG.PID_OUTV(s) / ER(s) = GAIN∗ (1 + 1 / (TI∗s))

Step Response



$$\text{PID\_LMNG.PID\_OUTV}(t) = \text{GAIN} * \text{ER0}\ (1 + \frac{1}{\text{TI}} * t)$$

where: PID_LMNG.

PID_OUTV(t) :	Manipulated variable automatic mode
ER :	Error signal
ER0 :	Error signal step change
GAIN :	Gain
TI :	Reset time
t :	Time

Figure 2-51 Step Response of a PI Controller

- **PD Controller**

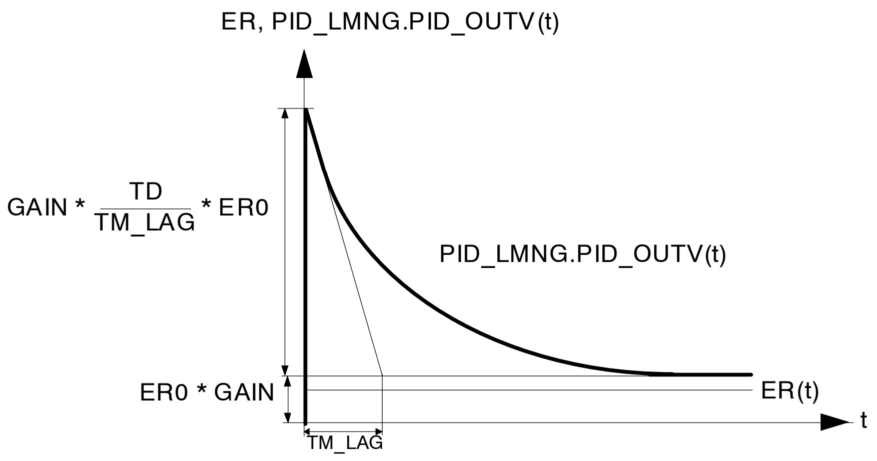
Here, the I action is turned off.

Transfer Function

The error signal ER is multiplied and differentiated. The transfer function in the Laplace range is:

PID_LMNG.PID_OUTV(s) / ER(s) = GAIN* (1 + TD*s / (1 + TM_LAG*s))

Step Response



$$\text{PID\_LMNG.PID\_OUTV}(t) = \text{GAIN} * \text{ER0}\ \left(1 + \frac{\text{TD}}{\text{TM\_LAG}} * e^{-\frac{t}{\text{TM\_LAG}}}\right)$$

where:

PID_LMNG. PID_OUTV (t) :	Manipulated variable automatic mode
ER :	Error signal
ER0 :	Error signal step change
GAIN :	Controller gain
TD :	Derivative time
TM_LAG :	Time lag
t :	Time

Figure 2-52 Step Response of a PD Controller

- **PID controller**

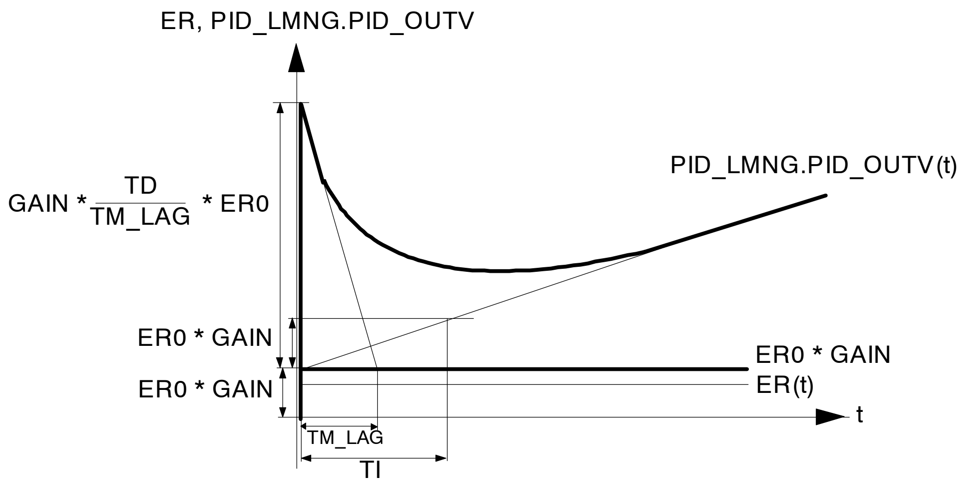
P, I and D actions are activated.

Transfer Function

The error signal ER is multiplied, integrated and differentiated. The transfer function in the Laplace range is:

PID_LMNG.PID_OUTV(s) / ER(s) = GAIN* (1 + 1 / (TI*s) + TD*s / (1 + TM_LAG*s))

Step Response



$$\text{PID\_LMNG.PID\_OUTV}(t) = \text{GAIN} * \text{ER0} \left(1 + \frac{1}{\text{TI}} * t + \frac{\text{TD}}{\text{TM\_LAG}} * e^{-\frac{t}{\text{TM\_LAG}}} \right)$$

where :	PID_LMNG.	
	PID_OUTV(t) :	Manipulated variable automatic mode
	ER :	Error signal
	ER0 :	Error signal step change
	GAIN :	Controller gain
	TI :	Reset time
	TD :	Derivative time
	TM_LAG :	Time lag
	t :	Time

Figure 2-53 Step Response of a PID Controller

Block Diagram

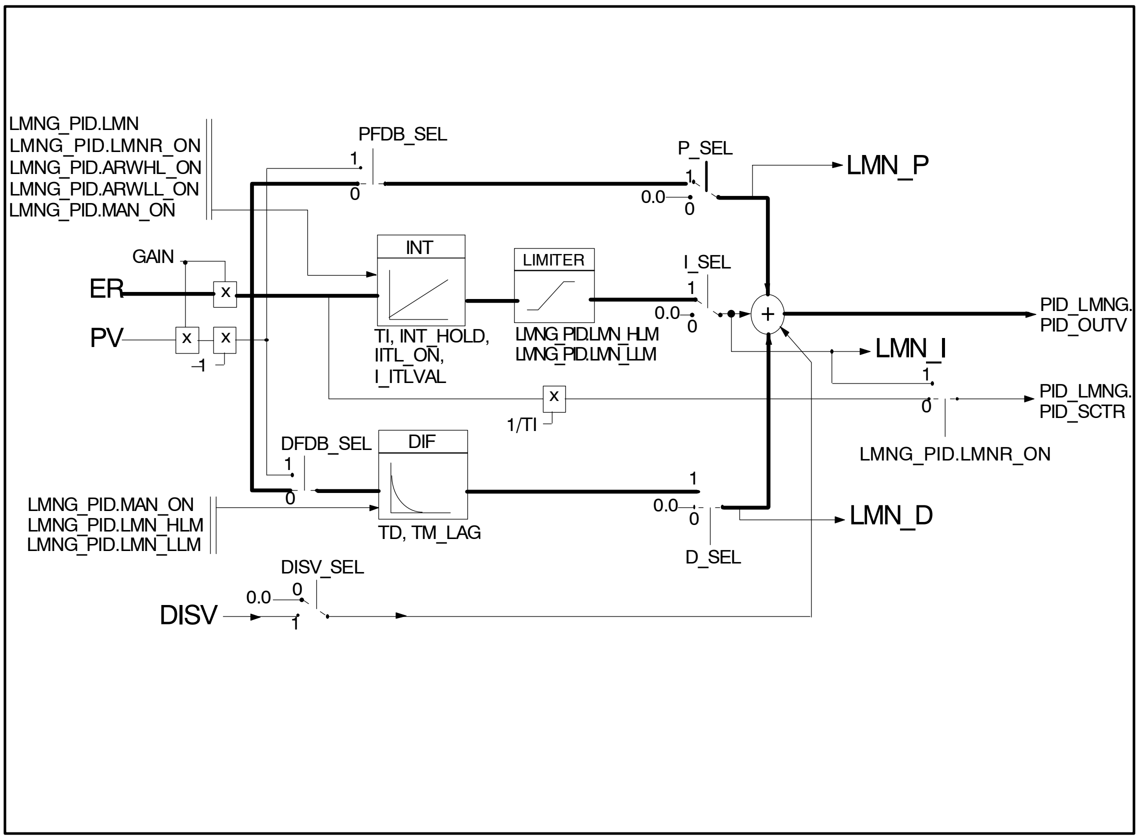


Figure 2-54   Block Diagram of a PID Algorithm

P Action

The proportional action can be activated and deactivated with the P_SEL switch. It can be included in the feedback path using the PFDB_SEL switch. In this case, the process variable PV is used as the input for the P action. The P action represents the product of the error signal ER (if the P action is in the feedback path it is the process variable PV) and the proportional gain GAIN.

I Action

The integral action can be activated and deactivated with the I_SEL switch. When it is deactivated, the I action and the internal memory of the integrator are set to zero. The I action can be frozen with INT_HOLD. The reset time is determined by the reset time constant TI. You can also assign your own value for the integrator. The value at input I_ITLVAL is transferred to the integrator via the I_ITL_ON switch. If the manipulated value is limited, the I action remains at the old value (anti reset wind-up).

D Action

The D action can be activated and deactivated with the D_SEL switch. It can be included in the feedback path using the DFDB_SEL switch. The input value of the D action is then the negative process variable PV. The time response is determined by the derivative action time TD. A first-order time lag is integrated in the derivative unit. The time lag is entered at TM_LAG.

Particularly in fast systems when the D action is active, unacceptable disturbance fluctuations can occur. In this case, the control quality can be improved by the time lag integrated in the D action. Usually a small TM_LAG is sufficient

Feedforward Control

A disturbance value DISV can be connected in addition to the manipulated value. This can be activated or deactivated using the DISV_SEL switch.

Block-Internal Limits

- The reset time is limited downwards to half the sampling time.
- The derivative time is limited downwards to the sampling time.
- The time lag is limited downwards to half the sampling time.

$TI_{\textbf{intern}}$ = CYCLE/2	when TI < CYCLE/2
$TD_{\textbf{intern}}$ = CYCLE	when TD < CYCLE
$TM\_LAG_{\textbf{intern}}$ = CYCLE/2	when TM_LAG < CYCLE/2

The values of the other input parameters are not limited in the block; the parameters are not checked.

2.1.21 PULSEGEN: Pulse Generator

Application

The block is used to structure a PID controller with pulse a output for proportional actuators. This allows three-step and two-step controllers with pulse duration modulation to be implemented.

Block Diagram

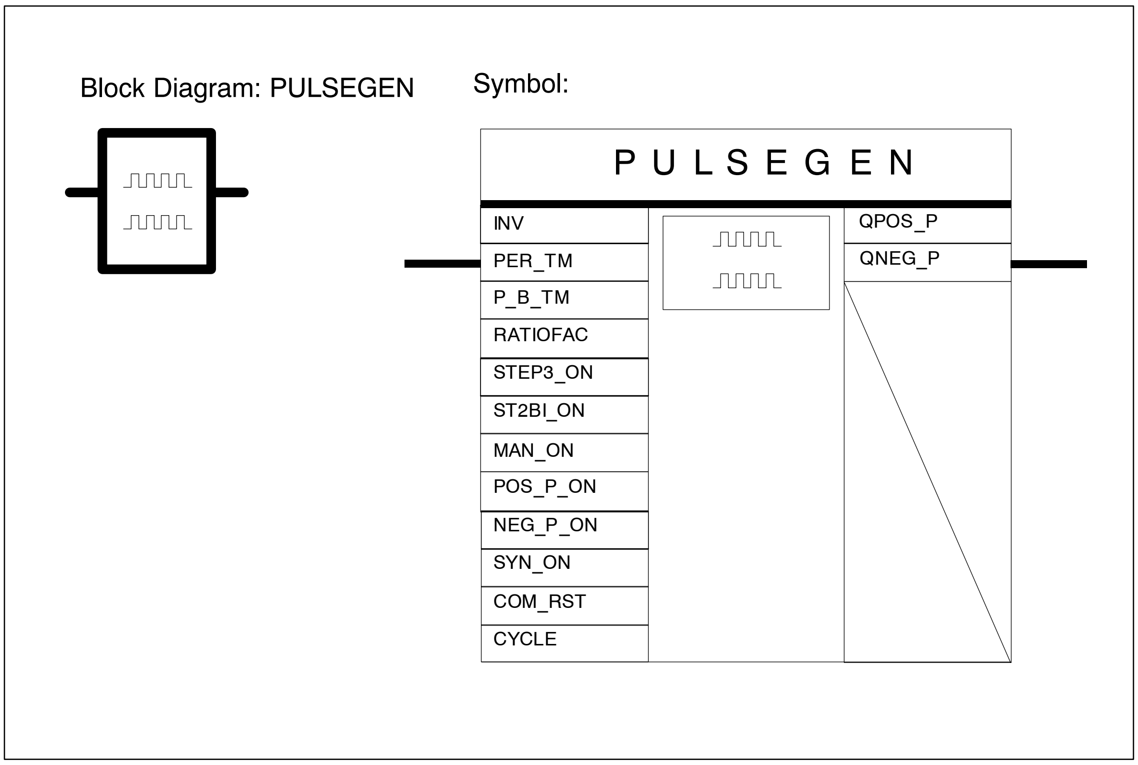


Figure 2-55 PULSEGEN, Block Diagram and Symbol

Functional Description

Using pulse duration modulation, the block transforms the input variable INV (= LMN of the PID controller) into a pulse train with a constant period PER_TM. The period corresponds to the cycle time interval at which the input variable is updated. The duration of a pulse per period is proportional to the input variable. An input variable of 30% therefore means: a positive pulse with a duration 0.3 * period, no pulse for 0.7 * period. The pulse duration is recalculated at the beginning of every period.

Figure 2-56 illustrates pulse duration modulation.

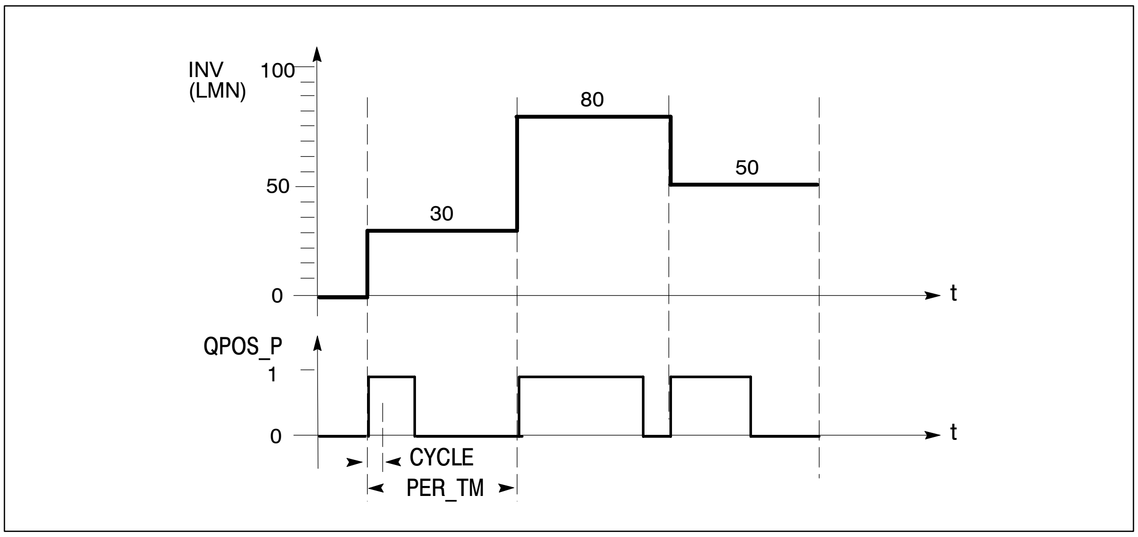


Figure 2-56 Pulse Duration Modulation

To reduce wear on the actuator, a minimum pulse/break time can be set.

In the three-step control mode, a ratio factor can be used to compensate different time constants for heating and cooling.

The block is usually used in conjunction with the continuous controller.

- **Two or three-step PID controller: PID+LMNGEN_C + PULSEGEN**

Figure 2-57 shows the connection of a PID Pulse Controller.

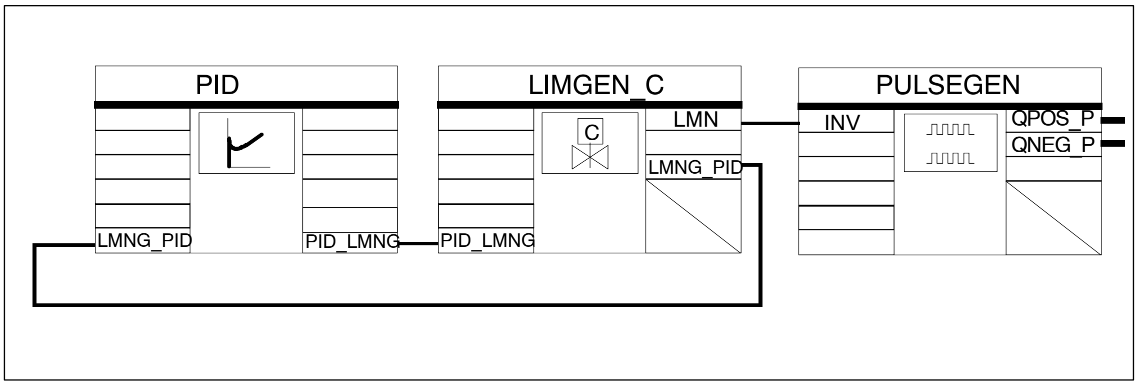


Figure 2-57 Connection of a PID Pulse Controller

Input Parameters

The following table shows the data type and structure of the input parameters of PULSEGEN.

Table 2-47 Input Parameters of PULSEGEN

Data Type	Parameter	Comment	Permitted Values	Default
REAL	INV	input variable	–100,0..100,0 (%)	0.0
TIME	PER_TM	period time	PER_TM >= 20∗CYCLE	T#1s
TIME	P_B_TM	minimum pulse/break time	P_B_TM >= CYCLE	T#50ms
REAL	RATIOFAC	ratio factor	0,1..10,0 (no dimension)	1.0
BOOL	STEP3_ON	three-step control on		TRUE
BOOL	ST2BI_ON	two-step control for bipolar manipulated value range on		FALSE
BOOL	MAN_ON	manual mode on		FALSE
BOOL	POS_P_ON	positive pulse on		FALSE
BOOL	NEG_P_ON	negative pulse on		FALSE
BOOL	SYN_ON	synchronization on		TRUE
BOOL	COM_RST	complete restart		FALSE
TIME	CYCLE	sampling time	CYCLE >= 1ms	T#10ms

Output Parameters

The following table shows the data type and structure of the output parameters PULSEGEN.

Table 2-48 Output Parameters of PULSEGEN

Data Type	Parameter	Comment	Default
BOOL	QPOS_P	output positive pulse	FALSE
BOOL	QNEG_P	output negative pulse	FALSE

Complete Restart

During a complete restart, all signal outputs are set to zero.

Accuracy of Pulse Generation

The implementation of this function in the CPU requires a decision about the current status of the binary signal n times at points in the cycle within a period of the controller output. The higher the value of n, the more accurate the pulse duration modulation.

While the continuous PID controller (PID–LMNGEN_C) is located in a slow cyclic interrupt priority class whose cycle time is adapted to the dominant system time constant, the PULSEGEN block must be located in a faster cyclic interrupt priority class. The faster the cyclic interrupt priority class, the greater the accuracy with which the manipulated value can be output. By using a cyclic interrupt priority class that is 100 times faster, you can achieve a resolution of 1% of the manipulated value range.

It is possible to synchronize the pulse output with the block that updates the input variable INV (for example PID_C) automatically. This ensures that a changing input variable is output as a pulse in the shortest possible time.

Automatic Synchronization

It is possible to synchronize the pulse output with the controller FB automatically. This ensures that a change in the value of the manipulated variable LMN(t) is output in the shortest possible time as a binary signal with a proportionally modified pulse duration.

The pulse generator evaluates the input variable INV at intervals corresponding to the period PER_TM. Since, however, INV is calculated in a slower cyclic interrupt priority class, the pulse generator should begin to convert the discrete value into a pulse signal as soon as possible after INV has been updated. To allow this, the block can synchronize the start of the period itself as explained below:

If INV has changed and if the block call is not in the first or in the last two call cycles of a period, synchronization is performed. The pulse duration is recalculated and in the next cycle, output starts with a new period.

The period PER_TM must correspond to the sampling time CYCLE of the controller.

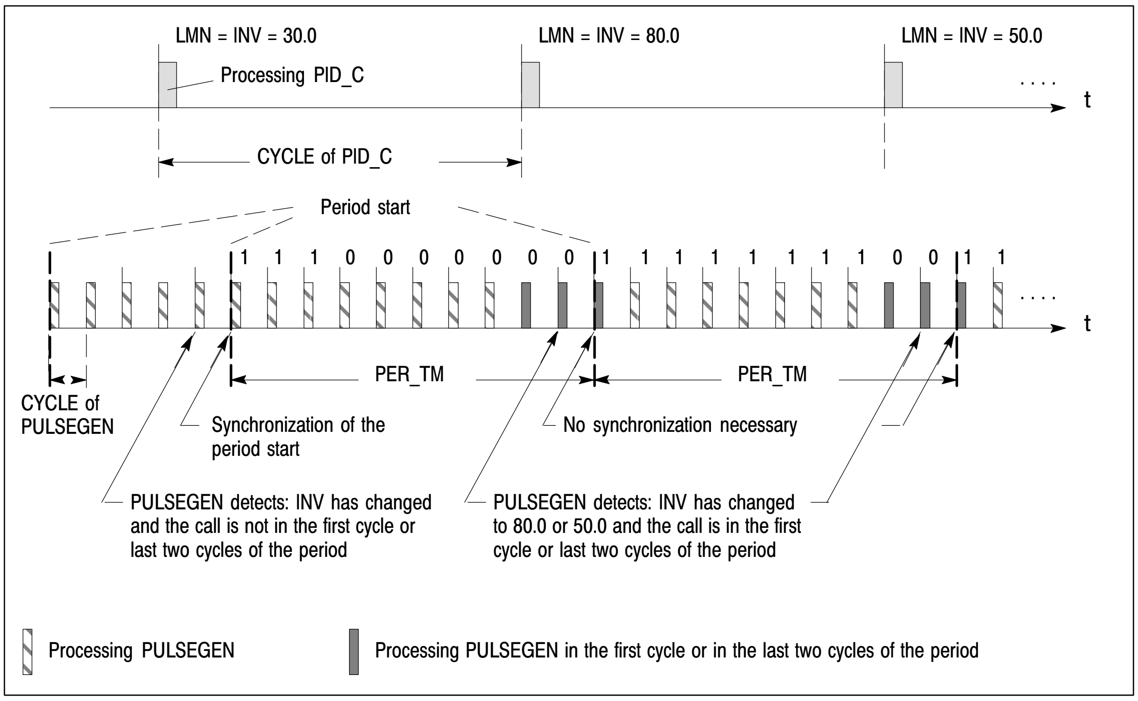

LMN = INV = 30.0
LMN = INV = 80.0
LMN = INV = 50.0
Processing PID_C
t
CYCLE of PID_C
Period start
1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 1 1
PER_TM
PER_TM
CYCLE of PULSEGEN
Synchronization of the period start
No synchronization necessary
PULSEGEN detects: INV has changed and the call is not in the first cycle or last two cycles of the period
PULSEGEN detects: INV has changed to 80.0 or 50.0 and the call is in the first cycle or last two cycles of the period
Processing PULSEGEN
Processing PULSEGEN in the first cycle or in the last two cycles of the period

Figure 2-58 Synchronization of the Start of the Period

Automatic synchronization can be turned off at input “SYN_ON” (= FALSE).

Note

With the start of the new period after synchronization, the old value of INV (in other words of LMN) is simulated more or less inaccurately by the pulse signal.

Modes of the Controller with Pulse Output

Depending on the parameter settings of the pulse generator, PID controllers with three-step output or with bipolar or monopolar two-step output are configured. The following table shows the switch combinations for the possible modes.

Switch / Mode	MAN_ON	STEP3_ON	ST2BI_ON
Three-step controller	FALSE	TRUE	any
Two-step controller with bipolar range (-100 % to 100 %)	FALSE	FALSE	TRUE
Two-step controller with monopolar range (0 % to 100 %)	FALSE	FALSE	FALSE
Manual mode	TRUE	any	any

Three-Step Control

In the three-step control mode, three states can be generated for the actuator signal, for example, depending on the actuator and process: more – off – less, forwards – stop – backwards, heat – off – cool etc. Depending on the requirements of the process to be controlled, the states of the binary output signals QPOS_P and QNEG_P are assigned to the corresponding operating states of the actuator. The table shows two examples.

	Heat Forwards	**Off Stop**	**Cool Backwards**
QPOS_P	TRUE	FALSE	FALSE
QNEG_P	FALSE	FALSE	TRUE

Dimensioning the minimum pulse or minimum break P_B_TM can prevent extremely short on and off times that can reduce the working life of switches and actuators. The proportional characteristic with which the pulse output is calculated has a response threshold applied to it.

Note

Small absolute values of the input variable LMN that would produce a pulse duration less than P_B_TM are suppressed. For large input values that would produce a pulse duration greater than PER_TM – P_B_TM, the pulse duration is set to 100% or –100%.

Values P_B_TM $\leq$ 0.1 * PER_TM are recommended.

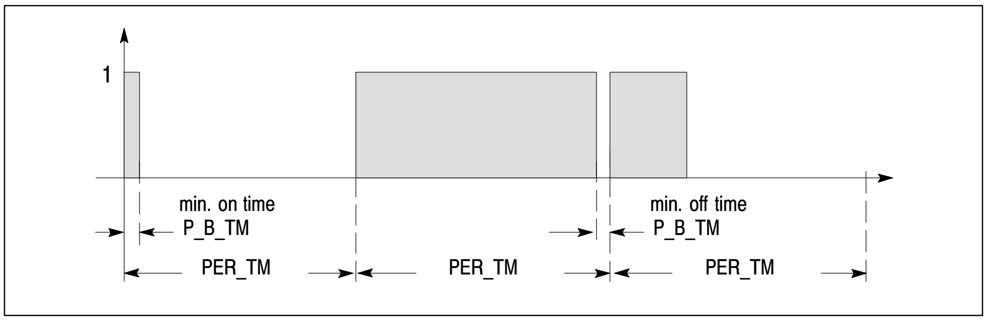


Figure 2-59 How the Pulse Output Switches On and Off

The duration of the positive or negative pulses is calculated from the input variable (in %) multiplied by the period:

$$\text{Pulse duration} = \frac{\text{INV}}{100} * \text{PER\_TM[s]}$$

Suppressing minimum pulse and minimum break times produces “doglegs” in the conversion characteristic at the start and end of the range. (Figure 2-60).

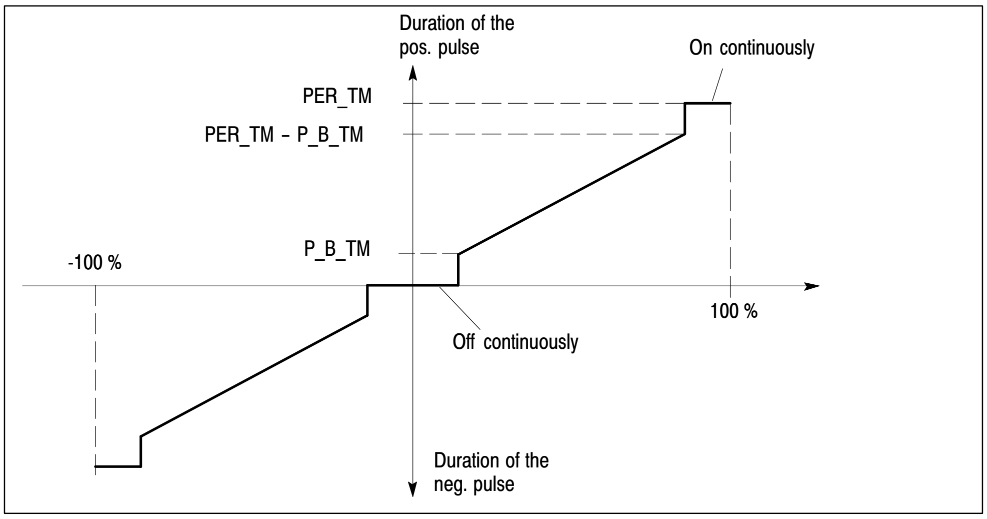


Figure 2-60 Symmetrical Curve of the Three-Step Controller (Ratio Factor = 1)

Asymmetrical Three-Step Control

With the ratio factor “RATIOFAC”, the ratio of the duration of negative pulses to positive pulses can be changed. In a thermal process, different time constants for heating and cooling can be compensated.

If, with the same absolute value for the input variable |INV|, the pulse duration at the negative pulse output must be shorter than the positive pulse, a ratio factor less than 1 must be set (Figure 2-61):

pos. pulse > neg. pulse: RATIOFAC < 1

Pulse duration negative: $\frac{\text{INV}}{100} * \text{PER\_TM} * \text{RATIOFAC}$

Pulse duration positive: $\frac{\text{INV}}{100} * \text{PER\_TM}$

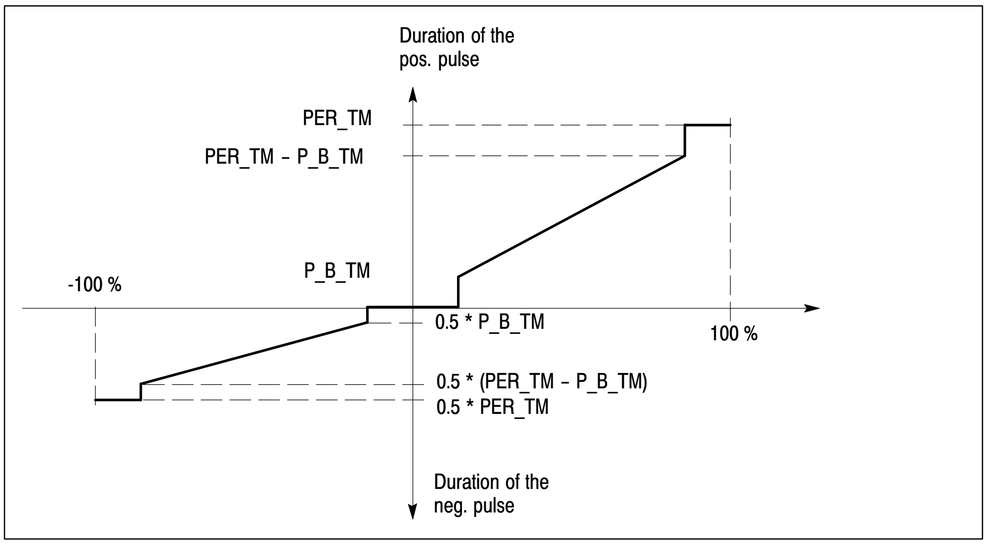


Figure 2-61 Asymmetrical Curve of the Three-step Controller (Ratio Factor = 0.5)

If the opposite is required, so that with the same absolute value for the input variable |INV|, the pulse duration at the positive pulse output is shorter than the negative pulse, a ratio factor greater than 1 must be set:

pos. pulse < neg. pulse: RATIOFAC > 1

Pulse duration negative: $\frac{INV}{100} * PER\_TM$

Pulse duration postitive: $\frac{INV * PER\_TM}{100 * RATIOFAC}$

The ratio factor also influences the minimum pulse and minimum break times. Mathematically, this means that with RATIOFAC < 1, the on threshold for negative pulses is multiplied by the ratio factor and with RATIOFAC > 1 the on threshold for positive pulses is divided by the ratio factor.

Two-Step Control

In a two-step control, only positive pulse output QPOS_P of PULSEGEN is connected to the on/off actuator. Depending on the actuating range, LMN = –100.0 ... 100.0 % or LMN = 0.0 % ... 100.0 %, the two-step controller has a bipolar or monopolar actuating range.

In the monopolar mode, the input variable INV can only have values between 0.0 and 100%.

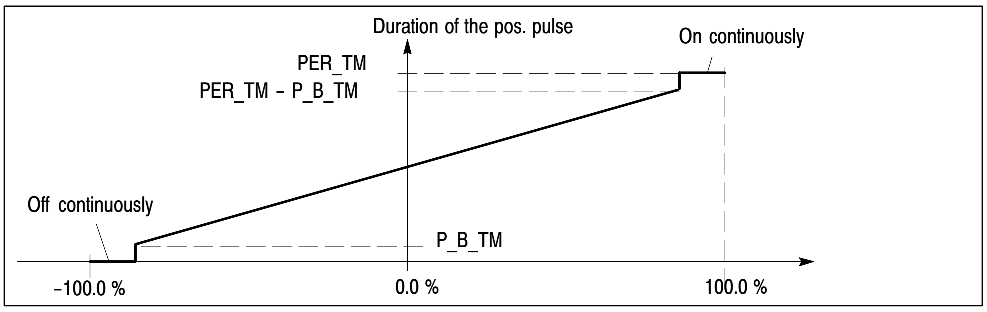


Figure 2-62 Two-step Controller With Bipolar Range (–100% to 100%)

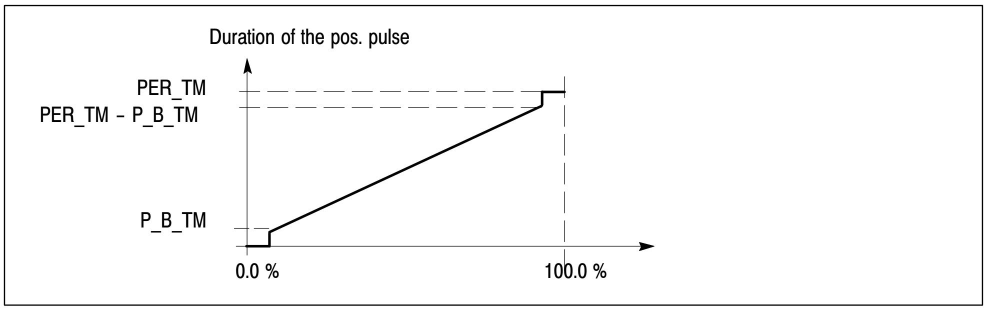


Figure 2-63 Two-step Controller With a Monopolar Range (0% to 100%)

The negated output signal is available at QNEG_P if the connection of the two-step controller in the control loop requires a logically inverted binary signal for the control pulses.

	On	Off
QPOS_P	TRUE	FALSE
QNEG_P	FALSE	TRUE

Manual Mode in Two or Three-Step Control

In the manual mode (MAN_ON = TRUE), the binary outputs of the three-step or two-step controllers can be set by the signals POS_P_ON and NEG_P_ON regardless of INV.

	POS_P_ON	**NEG_P_ON**	**QPOS_P**	**QNEG_P**
Three-step controller	FALSE	FALSE	FALSE	FALSE
	TRUE	FALSE	TRUE	FALSE
	FALSE	TRUE	FALSE	TRUE
	TRUE	TRUE	FALSE	FALSE
Two-step controller	FALSE	any	FALSE	TRUE
	TRUE	any	TRUE	FALSE

Block-Internal Limits

The values of the parameters are not limited in the block; the parameters are not checked.

2.1.22 RMP_SOAK: Ramp Soak

Application

The ramp soak is used mainly as a setpoint value generator for a controller that sets different setpoints at different times while the process is running.

Block Diagram

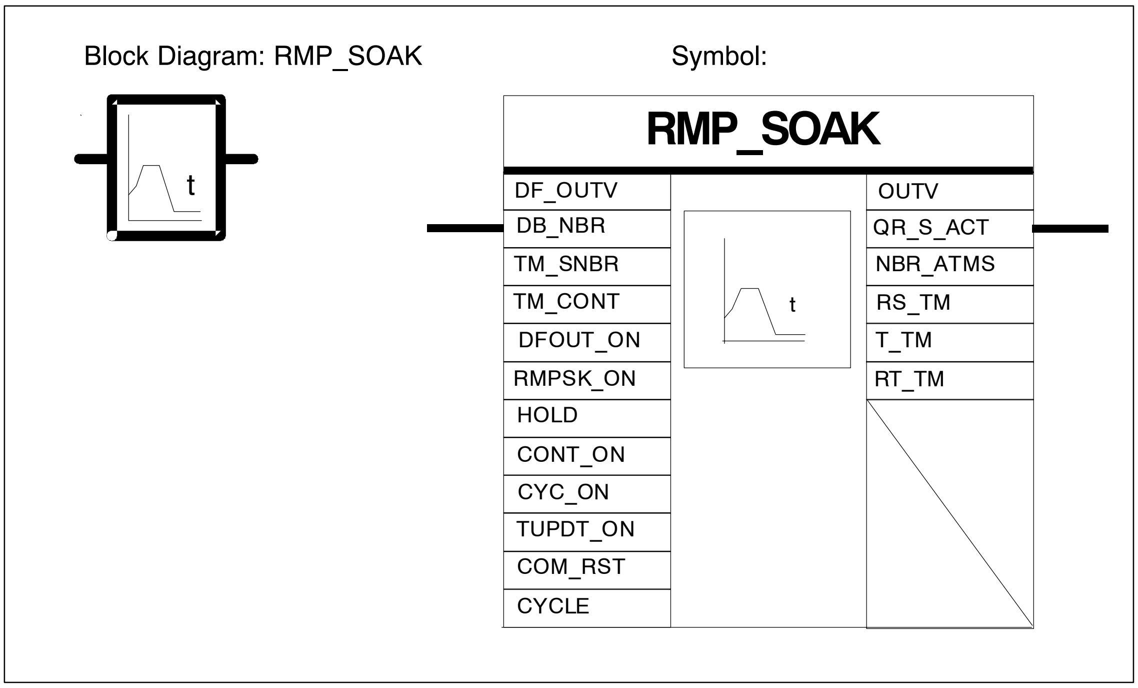


Figure 2-64 RMP_SOAK, Block Diagram and Symbol

Functional Description

This block allows curves whose coordinates are stored in a global data block to be executed. In each call cycle, values are output according to a time schedule. The value is interpolated in the time slices between the coordinates.

The following modes can be selected with control inputs:

- Ramp soak on
- Default output variable
- Hold processing
- Set time slice number and time to continue
- Repetition on (cyclic mode)
- Total time and total time remaining update on

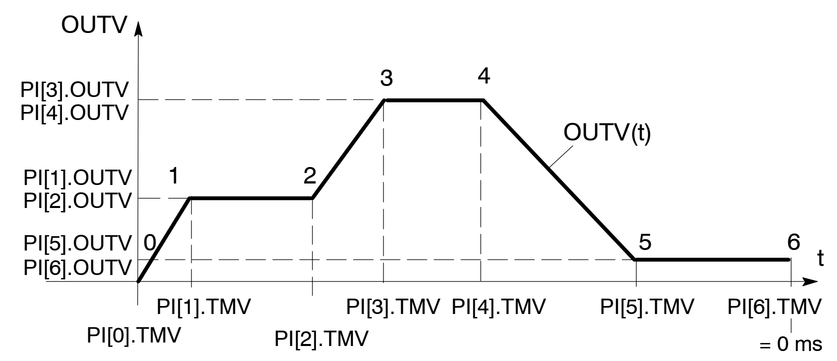


Figure 2-65 Example of a Ramp Soak with Starting Point and 6 Coordinates

With n coordinates, the time value for coordinate n = 0 ms (end of processing).

Note

The block does not check whether a shared DB with the number DB_NBR exists or not and whether the parameter DB_NBR.NBR_PTS (number of time slices) matches the DB length. If the parameter assignment is incorrect, the CPU changes to STOP due to an internal system error.

Input Parameters

The following table shows the data type and structure of the input parameters of RMP_SOAK.

Table 2-49 Input Parameters of RMP_SOAK

Data Type	Parameter	Comment	Permitted Values	Default
REAL	DF_OUTV	default output variable	technical range of values	0.0
BLOCK_DB	DB_NBR	data block number	depends on the CPU	DB 1
INT	TM_SNBR	time slice number	0 – 255	0
TIME	TM_CONT	time to continue (instant)	technical range of values	T#0s
BOOL	DFOUT_ON	default output variable on		FALSE
BOOL	RMPSK_ON	ramp soak on		FALSE
BOOL	HOLD	hold output variable		FALSE
BOOL	CONT_ON	continue		FALSE
BOOL	CYC_ON	cyclic repetition on		FALSE
BOOL	TUPDT_ON	total time update on		FALSE
BOOL	COM_RST	complete restart		FALSE
TIME	CYCLE	sampling time	technical range of values≥ 1ms	T#1s

Output Parameters

The following table shows the data type and structure of the output parameters of RMP_SOAK.

Table 2-50 Output Parameters of RMP_SOAK

Data Type	Parameter	Comment	Default
REAL	OUTV	output variable	0.0
BOOL	QR_S_ACT	ramp soak active	FALSE
INT	NBR_ATMS	number of acting time slice	0
TIME	RS_TM	remaining slice time	T#0s
TIME	T_TM	total time	T#0s
TIME	RT_TM	remaining total time	T#0s

The coordinates (points) and the number of points NBRPTS are stored in a shared data block (DB_NBR). Output begins at point 0 and ends at point NBR_PTS.

Shared Data

(Default with starting point and 4 points)

Table 2-51

Data Type	Parameter DB_NBR.	Comment	Permitted Values	Default
INT	NBR_PTS	number of points – 1	1 – 255	4
REAL	PI[0].OUTV	output variable [0]	technical range of values	0.0
TIME	PI[0].TMV	output time value [0]		T#1s
REAL	PI[1].OUTV	output variable [1]	technical range of values	0.0
TIME	PI[1].TMV	output time value [1]		T#1s
REAL	PI[2].OUTV	output variable [2]	technical range of values	0.0
TIME	PI[2].TMV	output time value [2]		T#1s
REAL	PI[3].OUTV	output variable [3]	technical range of values	0.0
TIME	PI[3].TMV	output time value [3]		T#1s
REAL	PI[4].OUTV	output variable [4]	technical range of values	0.0
TIME	PI[4].TMV	output time value [4]	0 ms	T#0s

Complete Restart

During a complete restart, output OUTV is set to 0.0. If DFOUT_ON=TRUE is set, DF_OUTV is output. The time slices (0 to NBRPTS–1) between points 0 to NBRPTS are totaled and are available at T_TM. The output QR_S_ACT is reset, the outputs NBR_ATMS and RS_TM are set to 0.

Normal Operation

- The coordinate parameters NBR_PTS, PI[i].TMV and PI[i].OUTV are stored in a shared data block.
- The parameter PI[i].TMV must be specified in the TIME format.
- The way in which the point values and time slice are counted is illustrated in the following schematic:

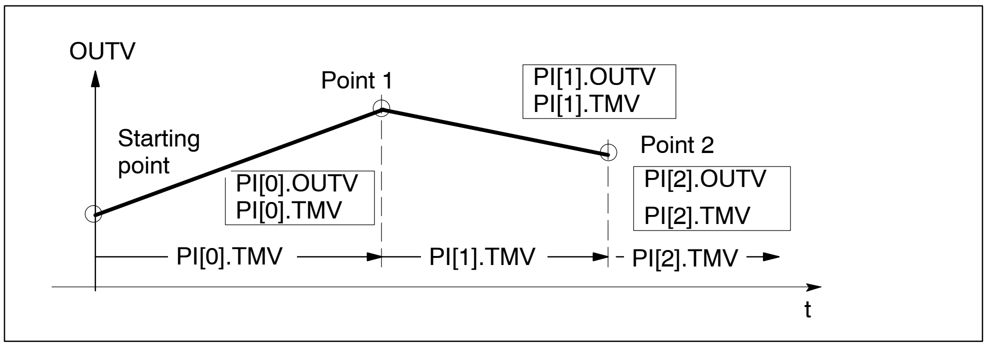


Figure 2-66 Counting the Coordinates and Time Slices

In normal operation, the ramp soak interpolates according to the following function where $0 \leq n < (\text{NBR\_PTS} - 1)$:

$$\text{OUTV}(t) = \text{PI}[n+1].\text{OUTV} - \frac{\text{RS\_TM}}{\text{PI}[n].\text{TMV}} (\text{PI}[n+1].\text{OUTV} - \text{PI}[n].\text{OUTV})$$

Modes of the Ramp Soak

To influence the control inputs, the following states and modes of the ramp soak can be implemented:

1. Ramp soak on for one run
2. Default value at output of the ramp soak
3. Cyclic ramp soak mode on
4. Hold ramp soak
5. Time slice number and time to continue (the remaining slice time RS_TM and the point number TM_SNBR are redefined)
6. Update total time and total time remaining

Modes

When setting one of the modes, the values of the control inputs as shown in the following table apply:

Table 2-52 Modes of the Ramp Soak (RMP_SOAK)

Mode	RMPSK _ON	DFOUT _ON	RMP _HOLD	CONT _ON	CYC _ON	TUPDT _ON	Output Signal OUTV
1. Ramp soak on	TRUE	FALSE	FALSE		FALSE		OUTV(t) Final value retained on completion of processing.
2. Default output variable	TRUE	TRUE					DF_OUTV
3. Cyclic ramp soak mode on	TRUE	FALSE	FALSE		TRUE		OUTV(t) Automatic start when completed
4. Hold ramp soak	TRUE	FALSE	TRUE	FALSE			Current value of OUTV(t) retained *)
5. Set time slice and time to continue	TRUE	FALSE	TRUE	TRUE			OUTV (old) *)
			FALSE				The ramp soak continues with new values.
6. Update total time						FALSE	Does not affect OUTV
						TRUE	Does not affect OUTV

*) As far as the next point, the curve does not have the slope set by the user.

▨ The selected mode is executed regardless of the value of the control signals in the shaded fields.

Ramp Soak On

The change in RMPSK_ON from FALSE to TRUE activates the ramp soak. After reaching the last time slice (last point in the curve), the ramp soak (curve) is completed. If you want to restart the function manually, RMPSK_ON must first be set to FALSE and then back to TRUE.

Default Output Variable, Starting the Ramp Soak

If you want the ramp soak to start at a specific output value, you must set DFOUT_ON = TRUE. In this case the signal value DF_OUTV is applied to the output.

Note

The signal for output of the constant setpoint DFOUT_ON has higher priority than the start signal for the ramp soak RMPSK_ON.

After the changeover from DFOUT_ON = FALSE, OUTV is adjusted with a linear rate of change starting from the setpoint (DF_OUTV) to the output value of the current point number PI[NBR_ATMS].OUTV.

Internal time processing is continued even when a fixed setpoint is applied to the output (RMPSK_ON = TRUE and DFOUT_ON = TRUE).

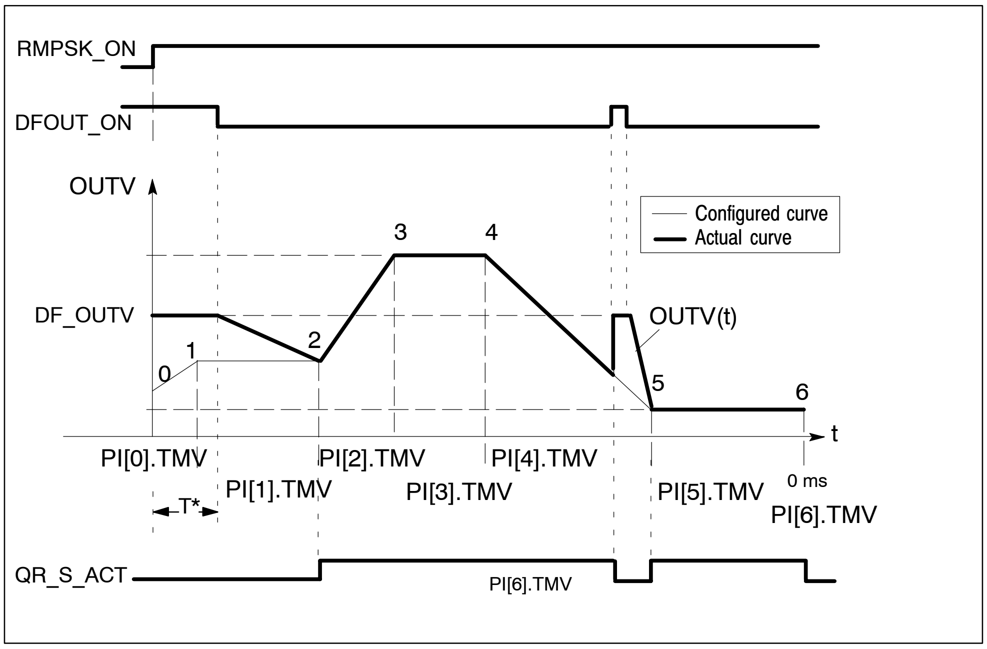


Figure 2-67 Influencing the Ramp Soak with the Default Signal DFOUT_ON

When the ramp soak is started with RMPSK_ON = TRUE, the fixed setpoint DF_OUTV is output until DFOUT_ON changes from TRUE to FALSE after the time T* (Figure 2-67). At this point, the time PI[0].TMV and part of the time PI[1].TMV has expired. OUTV changes from DF_OUTV to PI[2].OUTV other words to point 2.

The configured curve is only output starting at point 2, in other words the output signal QR_S_ACT has the value TRUE. If the default signal DFOUT_ON changes while the ramp soak is being processed, the output value OUTV jumps to DF_OUTV without delay.

Cyclic Ramp Soak Mode On

If the 'cyclic repetition' mode is active (CYC_ON = TRUE), the ramp soak automatically returns to the starting point and runs through again when it has output the last value.

There is no interpolation between the last point and the starting point. To achieve a smooth transition, the following must apply: PI[NBR_PTS].OUTV = PI[0].OUTV.

Hold Ramp Soak

If RMP_HOLD = TRUE is set, the value of the output variable (including time processing) is put on hold. When this is reset (RMP_HOLD = FALSE), the ramp soak continues from the point at which it was stopped PI[x].TMV.

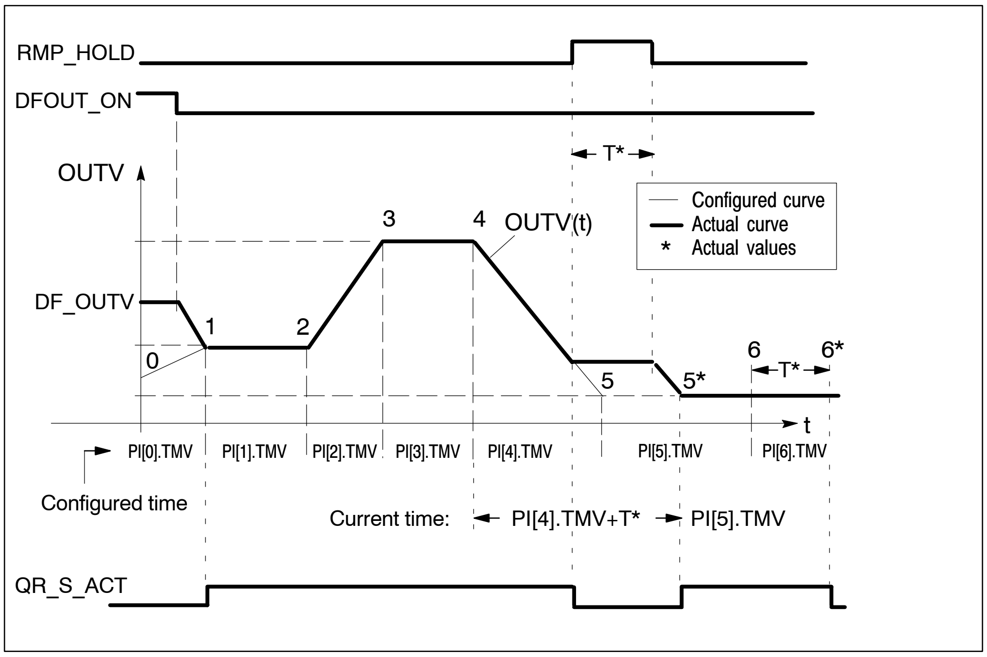


Figure 2-68 Influencing the Ramp Soak with the Hold Signal RMP_HOLD

The execution time of the ramp soak is extended by the hold time T*. The ramp soak takes the configured course from point 1 until the signal change at RMP_HOLD (FALSE → TRUE) and from point 5* to point 6*, in other words the output signal QR_S_ACT has the value TRUE. (Figure 2-68).

If the bit CONT_ON is set, the stopped ramp soak continues operation at the specified point TM_CONT.

Time Slice and Time to Continue

If the control input CONT_ON for the continuation of the ramp soak is set to TRUE, the ramp soak continues at TM_CONT (time to continue) with point TM_SNBR (destination point). The time parameter TM_CONT determines the remaining time that the ramp soak requires to the destination point TM_SNBR.

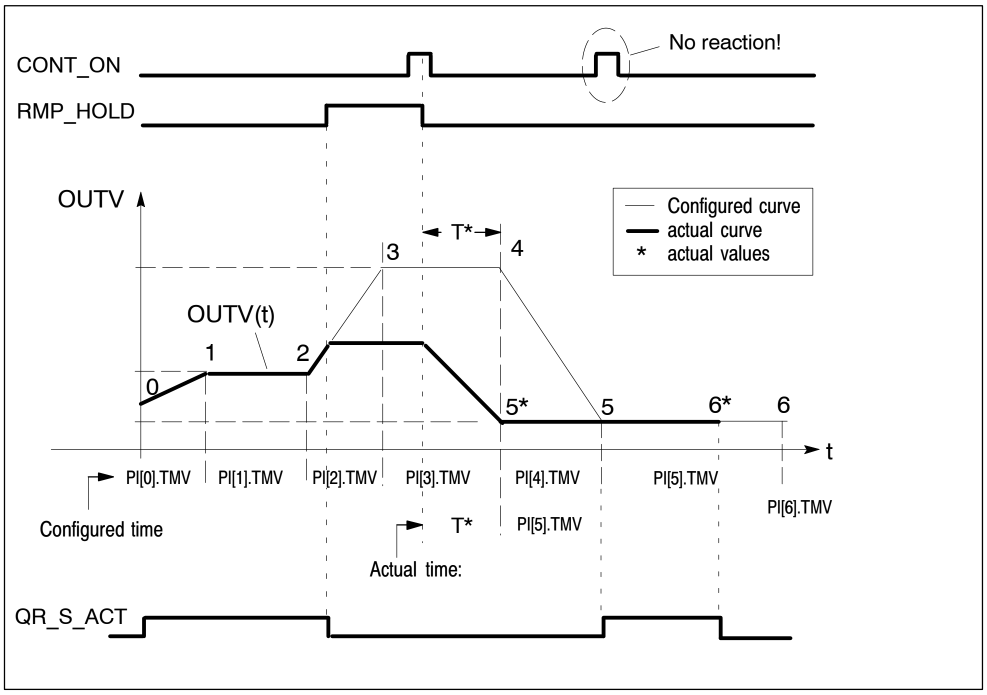


Figure 2-69 Influencing the Ramp Soak of the Hold Signal RMP_HOLD and the Continue Signal CONT_ON

In the example (Figure 2-69), if RMP_HOLD = TRUE and CONT_ON = TRUE and if the following are set:

time slice to continue	TM_SNBR = 5
and remaining time to required point	TM_CONT = T*

for the processing cycle of the ramp soak, configured points 3 and 4 are omitted. After a signal change at RMP_HOLD from TRUE to FALSE, the configured curve is only achieved again starting at point 5.

The output QR_S_ACT is only set , when the ramp soak has worked through the curve selected by the user.

Update Total Time and Total Time Remaining

In every cycle, the current point number NBR_ATMS, the current actual remaining time until the curve point is reached RS_TM, the total time T_TM and the total time remaining to the end of the curve RT_TM are updated.

If PI[n].TMV is changed online, the total time and the total time remaining to the end of the curve are also changed. Since the calculation of T_TM and RT_TM greatly increases the run time of the function block when there is a large number of time slices, the calculation is only made after a complete restart or when TUPDT_ON = TRUE is set. The time slices PI[0 ... NBR_PTS].TMV between the individual curve points are totaled and indicated at the outputs total time T_TM and total time remaining RT_TM.

Please note that determining the total times means a relatively long CPU run time!

Block-Internal Limits

The values of the parameters are not limited in the block; the parameters are not checked.

2.1.23 ROC_LIM: Rate of Change Limiter

Application

Ramp functions are used when the process must not be subjected to a step change at the input. This is, for example, the case when gearing is included between the motor and the load and when increasing the motor speed too fast would overload the gearing.

Block Diagram

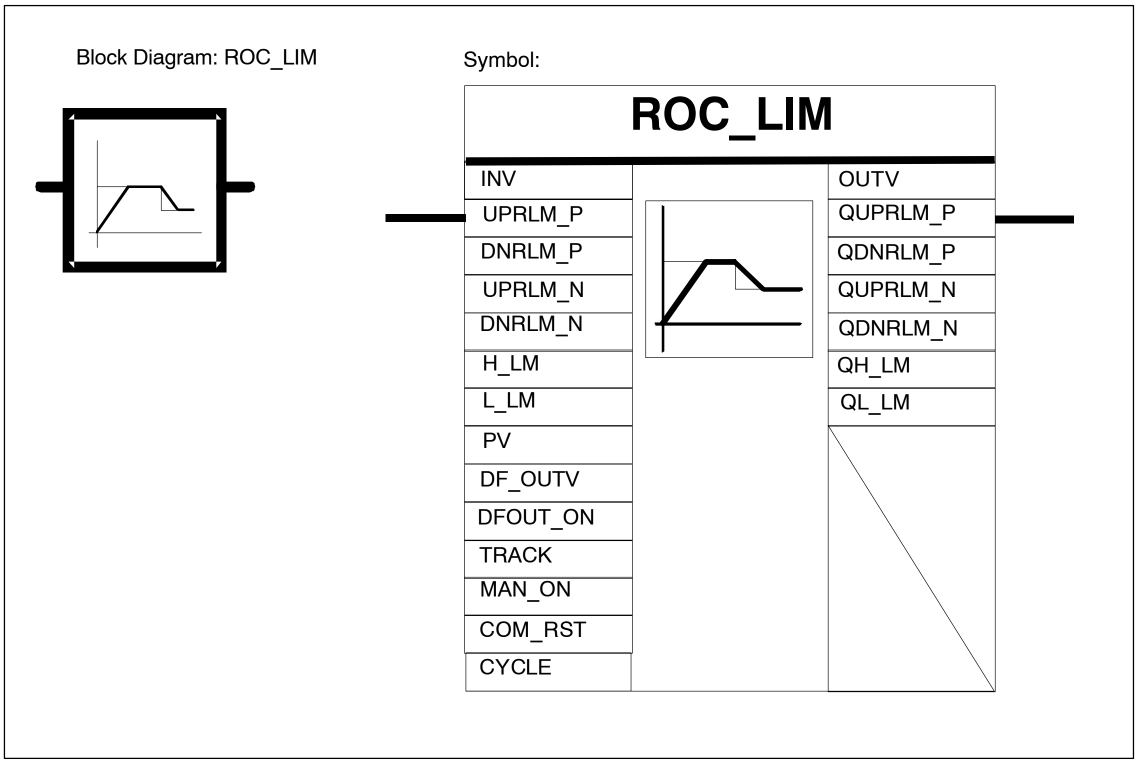


Figure 2-70 ROC_LIM, Block Diagram and Symbol

Functional Description

The block limits the rate of change of an output value. A step change becomes a ramp function. Two ramps (rising and falling values) in the positive and negative range can be selected for input variable and output variable. Control inputs set the following modes:

- Default output variable
- Tracking
- Stepless automatic-manual switchover

The value of the output variable can be limited by two selectable limits. If the rate of change limit rising or falling is reached or the high/low limit is reached, this is indicated at outputs.

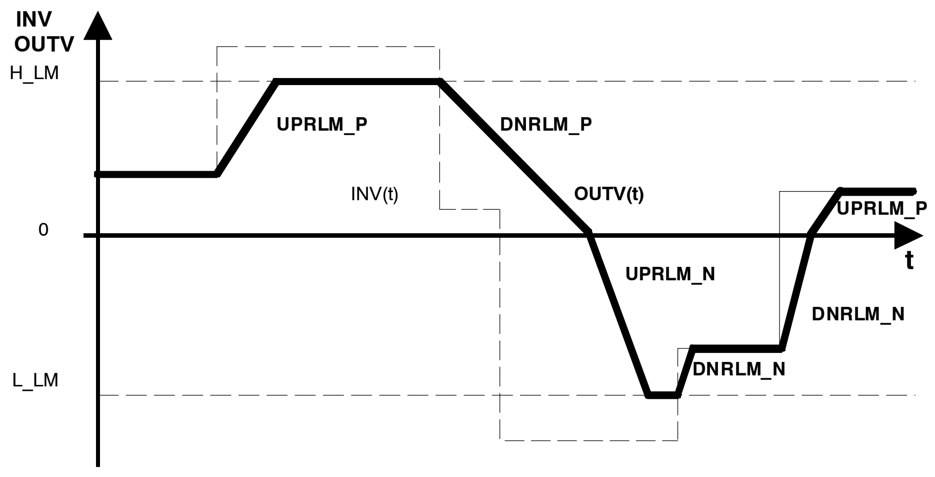


Figure 2-71 Ramp Function

The ramps are identified as follows:

OUTV > 0 and |OUTV| rising	UPRLM_P
OUTV > 0 and |OUTV| rising	UPRLM_P
OUTV > 0 and |OUTV| falling	DNRLM_P
OUTV < 0 and |OUTV| rising	UPRLM_N
OUTV < 0 and |OUTV| falling	DNRLM_N

Input Parameters

The following table shows the data type and structure of the input parameters of ROC_LIM.

Table 2-53

Data Type	Parameter	Comment	Permitted Values	Default
REAL	INV	input variable	technical range of values	0.0
REAL	UPRLM_P	up rate limit in positive range	> 0.0	10.0
REAL	DNRLM_P	down rate limit in positive range	> 0.0	10.0
REAL	UPRLM_N	up rate limit in negative range	> 0.0	10.0
REAL	DNRLM_N	down rate limit in negative range	> 0.0	10.0
REAL	H_LM	high limit	tech. range > L_LM	100.0
REAL	L_LM	low limit	tech. range < H_LM	0.0
REAL	PV	process variable	technical range of values	0.0
REAL	DF_OUTV	default output variable	technical range of values	0.0
BOOL	DFOUT_ON	default output variable on		FALSE
BOOL	TRACK	tracking OUTV = INV		FALSE
BOOL	MAN_ON	manual mode on		FALSE
BOOL	COM_RST	complete restart		FALSE
TIME	CYCLE	sampling time		T#1s

Output Parameters

The following table shows the data type and structure of the output parameters ROC_LIM.

Table 2-54

Data Type	Parameter	Comment	Default
REAL	OUTV	output variable	0.0
BOOL	QUPRLM_P	up rate limit in positive range reached	FALSE
BOOL	QDNRLM_P	down rate limit in positive range reached	FALSE
BOOL	QUPRLM_N	up rate limit in negative range reached	FALSE
BOOL	QDNRLM_N	down rate limit in negative range reached	FALSE
BOOL	QH_LM	high limit reached	FALSE
BOOL	QL_LM	low limit reached	FALSE

Complete Restart

During a complete restart, output OUTV is reset to 0.0. If DFOUT_ON = TRUE is set, DF_OUTV is output. All signal outputs are set to FALSE.

Normal Operation

The slopes are straight line limit curves and relate to a rise/fall per second. If, for example, the value 10.0 is set for UPRLM_P at a sampling time of 1s/100ms/10ms, then when the block is called, 10.0/1.0/0.1 is added to OUTV if INV > OUTV, until INV is reached. Limiting the output variable upwards and downwards is possible if the input variable exceeds H_LM or falls below L_LM. (Exception: manual mode MAN_ON=TRUE; see Examples Figure 2-72)

If one of the limits is exceeded, this is indicated at the outputs QUPRLM_P, QDNRLM_P, QUPRLM_N, QDNRLM_N , QH_LM and QL_LM.

- **Default Output Variable**

If DFOUT_ON = TRUE is set, DF_OUTV is output. If TRUE changes to FALSE, OUTV changes from DF_OUTV to INV and if FALSE changes to TRUE, OUTV changes from INV to DF_OUTV.

- **Tracking**

To track (OUTV = INV), the bit TRACK = TRUE is set. Since the input variable is switched directly to the output variable, any step changes in the input variable are output.

- **Stepless Manual-Automatic Switchover**

For this mode, the ramp block must be included in the setpoint branch directly before the error signal. The process variable is connected to the input PV and the manual-automatic bit to input MAN_ON. When you change to the manual mode MAN_ON = TRUE, the value at input PV is switched immediately to the output OUTV. Since the setpoint and process variable are the same, the error signal becomes zero and the controller is in a stationary settled state. When you return to the automatic mode MAN_ON = FALSE, the ramp function ensures a gradual change in the output OUTV from the current value PV to the input value INV. This results in a stepless switchover from the manual to the automatic mode (see Figure 2-72)

The default output variable mode has lower priority if MAN_ON=TRUE is set and is ignored.

Block-Internal Limits

The values of the parameters are not limited in the block; the parameters are not checked.

Example

If MAN_ON, TRACK, DFOUT_ON are set to FALSE, the signal outputs have values as shown below:

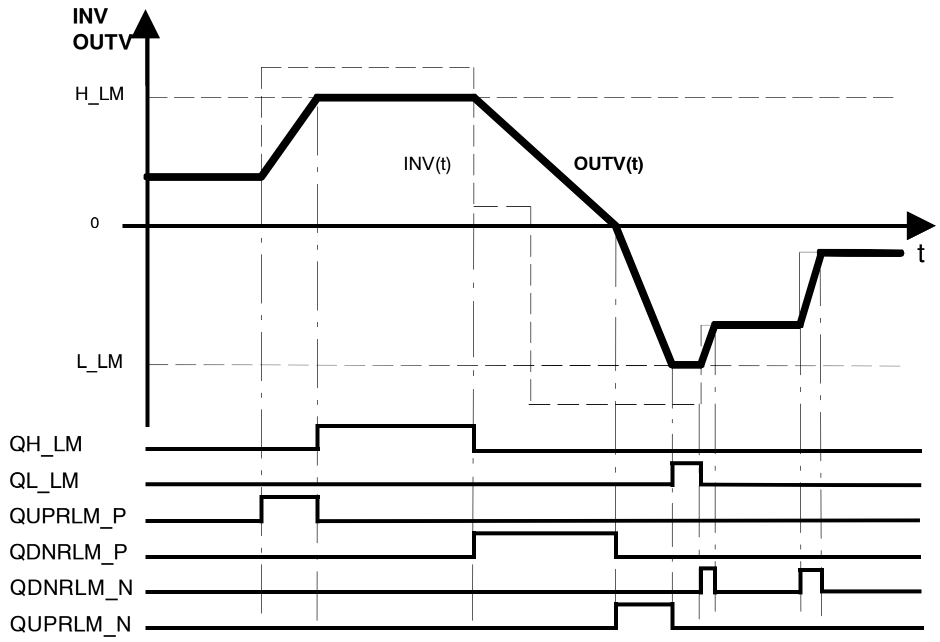


Figure 2-72 Example (MAN_ON, TRACK, DFOUT_ON = FALSE; L_LM < 0.0 < H_LM)

If MAN_ON is set, the limits H_LM and L_LM are not effective. If TRACK is set, the input value is output without being changed. If DFOUT_ON = TRUE, DF_OUTV is always output.

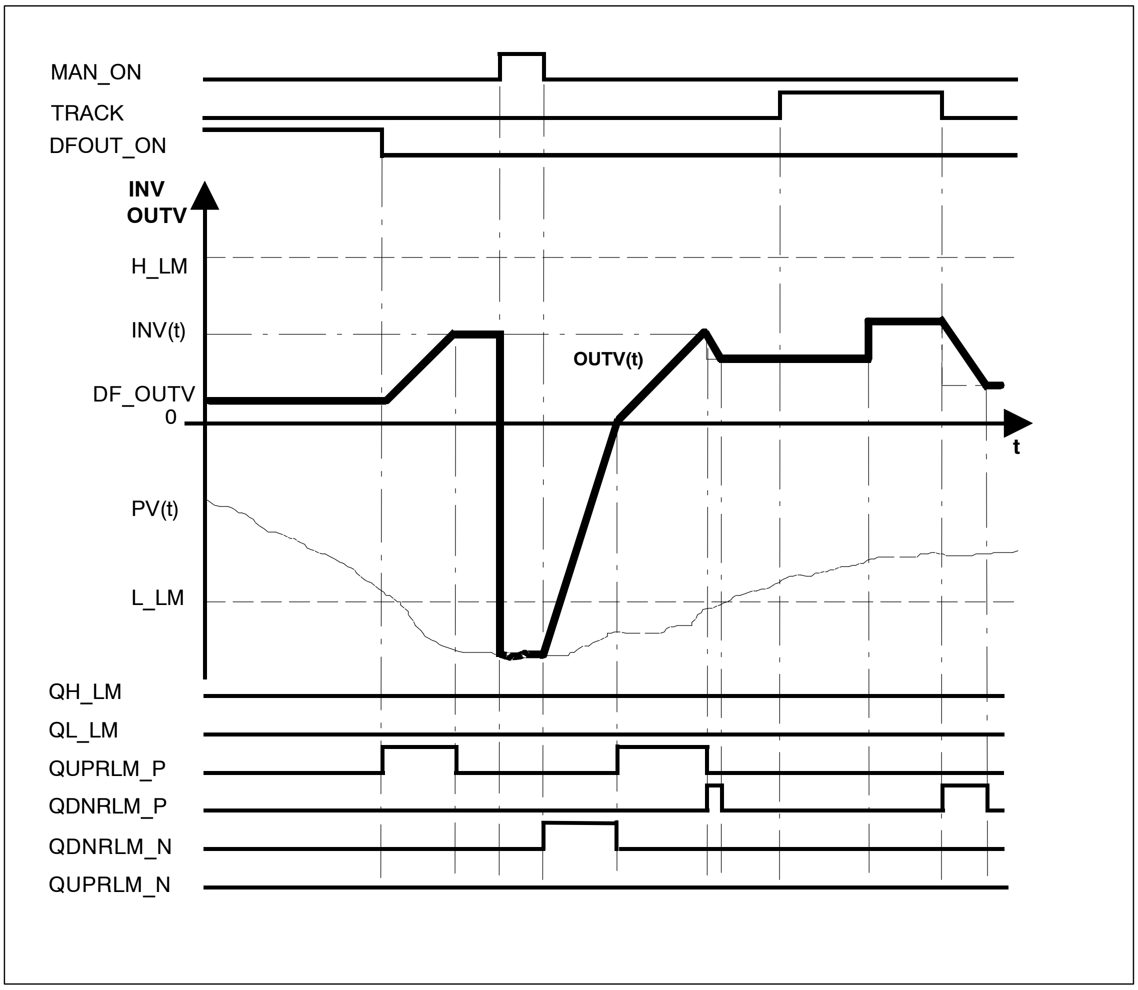


Figure 2-73 Example (L_LM < 0.0 < H_LM)

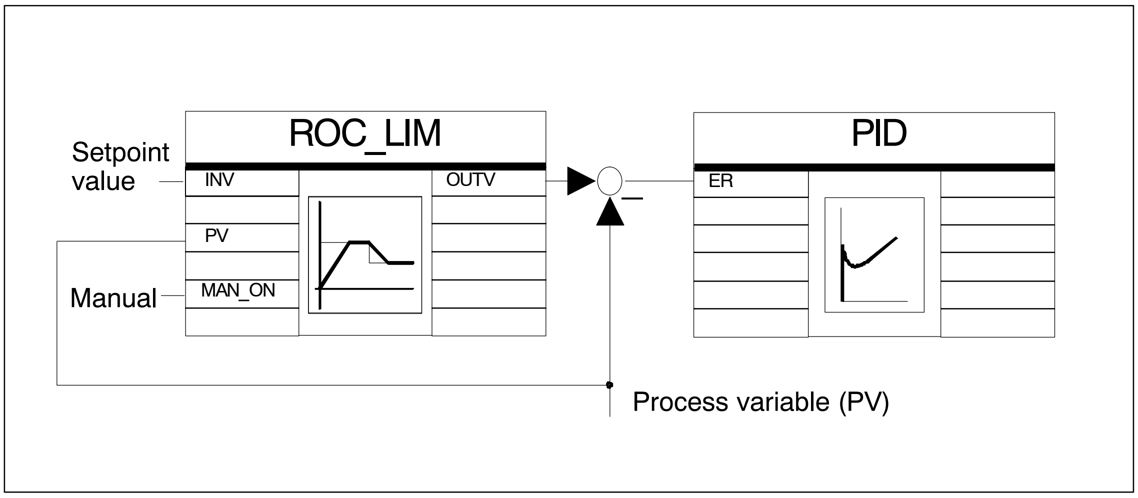


Figure 2-74 Example Stepless Changeover from Manual to Automatic

Example of a ramp used only in the positive range

Input Parameters

The following table shows the data type and structure of the input parameters.

Table 2-55 Input Parameters

Data Type	Parameter	Comment	Parameter Assignment
REAL	INV	input variable	0.0
REAL	UPRLM_P	up rate limit in positive range	10.0
REAL	DNRLM_P	down rate limit in positive range	5.0
REAL	UPRLM_N	up rate limit in negative range	0.0
REAL	DNRLM_N	down rate limit in negative range	0.0
REAL	H_LM	high limit	85.5
REAL	L_LM	low limit	27.0
REAL	PV	process variable	0.0
REAL	DF_OUTV	default output variable	46.15
BOOL	DFOUT_ON	default output variable on	FALSE
BOOL	TRACK	tracking OUTV = INV	FALSE
BOOL	MAN_ON	manual mode on	FALSE
BOOL	COM_RST	complete restart	FALSE
TIME	CYCLE	sampling time	T#1s

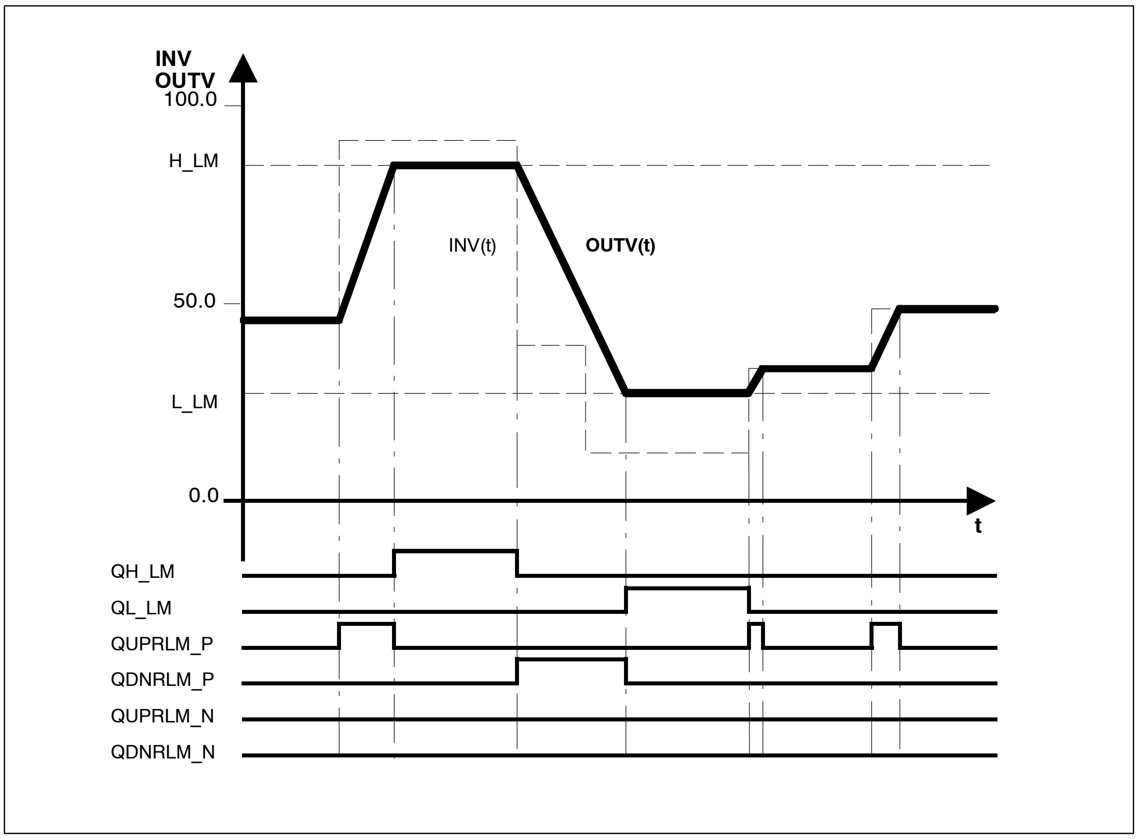


Figure 2-75 Example of an Operating Range with Values only ≥ 0.0

2.1.24 SCALE: Linear Scaling

Application

The value of a process variable supplied by a sensor is often in a range that is not particularly suitable for the user (for example 0 to 10 V correspond to 0 to 1200 °C or 0 to 10 V correspond to 0 to 3000 rpm). By adapting the setpoint or process variable, both variables can have the same range.

Block Diagram

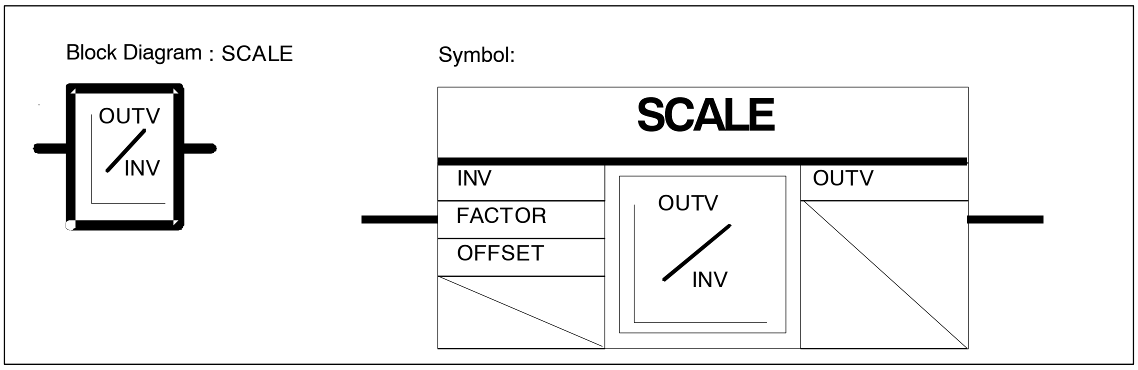


Figure 2-76 SCALE, Block Diagram and Symbol

Functional Description

The block normalizes an analog variable. The normalization curve is defined by the slope (FACTOR) and distance between OUTV when INV = 0 and the coordinate axis OUTV = 0.

Algorithm

OUTV = INV * FACTOR + OFFSET

An analog variable INV is applied to the output variable OUTV via the normalization curve. The normalization curve is defined by the variables FACTOR and OFFSET.

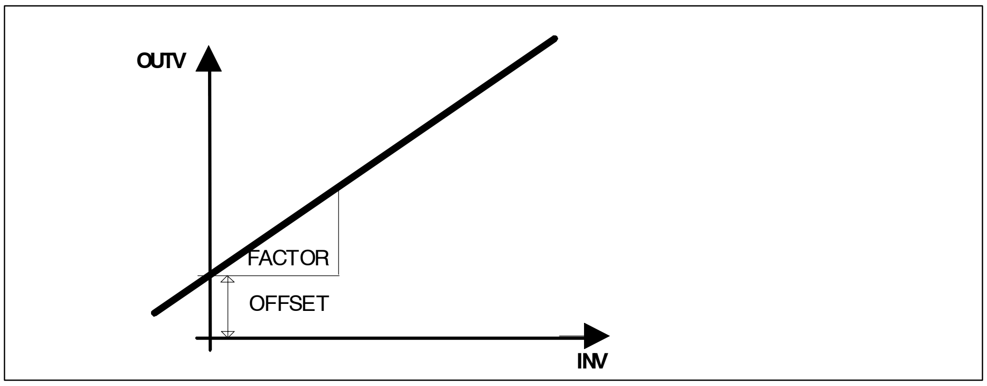


Figure 2-77 Normalization Curve with Limitation

Input Parameters

The following table shows the data type and structure of the input parameters of SCALE.

Table 2-56 Input Parameters of SCALE

Data Type	Parameter	Comment	Permitted Values	Default
REAL	INV	input variable	technical range of values	0.0
REAL	FACTOR	scaling factor		1.0
REAL	OFFSET	offset	technical range of values	0.0

Output Parameters

The following table shows the data type and structure of the output parameters SCALE.

Table 2-57 Output Parameters of SCALE

Data Type	Parameter	Comment	Default
REAL	OUTV	output variable	0.0

Complete Restart

The block has no complete restart routine.

Normal Operation

The block has no modes other than normal operation.

Block-Internal Limits

The values of the parameters are not limited in the block; the parameters are not checked.

2.1.25 SP_GEN: Setpoint Value Generator

Application

To enter a setpoint value manually, the output value can be modified with the SP_GEN block using two inputs. To allow small changes, the block should have a sampling time of $\leq$100 ms.

Block Diagram

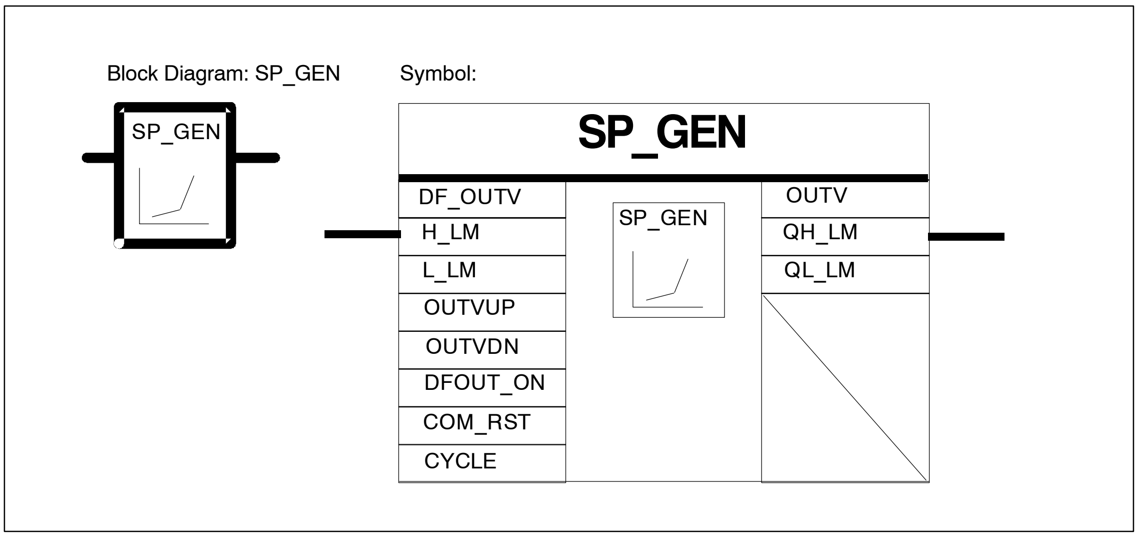


Figure 2-78 SP_GEN, Block Diagram and Symbol

Functional Description

At the inputs OUTVUP and OUTVDN, the output variable OUTV can be continuously increased or decreased within the limits H_LM and L_LM. The rate of change depends on the length of time that OUTVUP and OUTVDN are activated.

During the first three seconds after setting OUTVUP or OUTVDN the rate of change is

dV/dt = (H_LM – L_LM) / 100 seconds; then it is

dV/dt = (H_LM – L_LM) / 10 seconds.

The value of OUTV is L_LM OUTV H_LM; if OUTV is limited, a message is displayed.

With DFOUT_ON, OUTV can be assigned DF_OUTV.

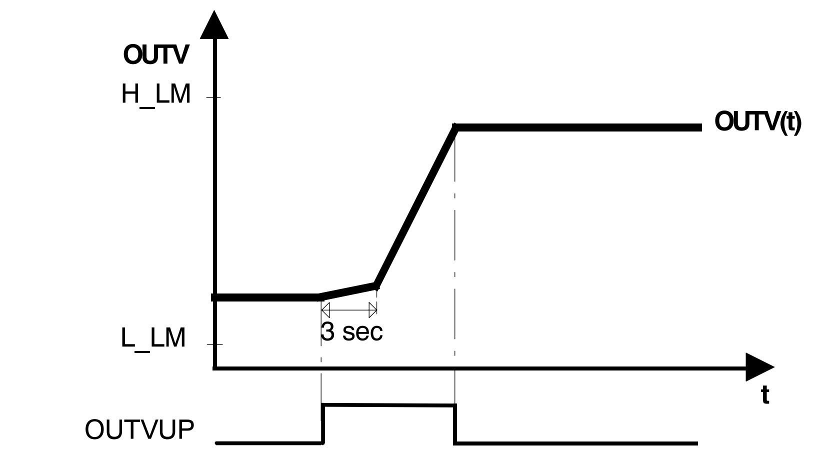


Figure 2-79 Changing the Output Value by Setting OUTVUP

Input Parameters

The following table shows the data type and structure of the input parameters of SP_GEN.

Table 2-58 Input Parameters of SP_GEN

Data Type	Parameter	Comment	Block-Internal Limits	Default
REAL	DF_OUTV	default output variable	technical range of values	0.0
REAL	H_LM	high limit	tech. range > L_LM	100.0
REAL	L_LM	low limit	tech. range < H_LM	0.0
BOOL	OUTVUP	output variable up		FALSE
BOOL	OUTVDN	output variable down		FALSE
BOOL	DFOUT_ON	default output variable on		FALSE
BOOL	COM_RST	complete restart		FALSE
TIME	CYCLE	sampling time		T#100ms

Output Parameters

The following table shows the data type and structure of the output parameters SP_GEN.

Table 2-59 Output Parameters of SP_GEN

Data Type	Parameter	Comment	Default after Complete Restart
REAL	OUTV	output variable	0.0
BOOL	QH_LM	high limit reached	FALSE
BOOL	QL_LM	low limit reached	FALSE

Complete Restart

During a complete restart, the OUTV is set to 0.0. If DFOUT_ON = TRUE is set, DF_OUTV is output. The limits are also effective during a complete restart.

Normal Operation

DFOUT_ON, OUTVUP and OUTVDN have the following influence on OUTV:

DFOUT_ON	OUTVDN	OUTVUP	OUTV
TRUE	any	any	DF_OUTV
FALSE	TRUE	TRUE	OUTV unchanged
	FALSE		OUTV rising
	TRUE	FALSE	OUTV falling
	FALSE		OUTV unchanged

- **Default Output Variable (DFOUT_ON = TRUE)**

If DFOUT_ON = TRUE is set, DF_OUTV is output. If the value of DF_OUTV is higher/lower than H_LM/L_LM, it is limited to H_LM/L_LM and QH_LM/QL_LM=TRUE is output. The change in OUTV is a step change. The changeover to DFOUT_ON = FALSE is without any sudden change.

- **Reduce Output Value (OUTVDN=TRUE)**

if OUTVDN=TRUE, OUTV is reduced by the following for 3 seconds

$$\boxed{(H\_LM - L\_LM) * \frac{CYCLE}{100s}}$$

After 3 seconds, OUTV is reduced by the following per cycle

$$(H\_LM - L\_LM) * \frac{CYCLE}{10s}$$

If the value of OUTV is less than L_LM, it is limited to L_LM and QL_LM=TRUE is output; if OUTVDN=FALSE is set, QL_LM=FALSE is also set. OUTVDN has lower priority than DFOUT_ON.

- **Increase Output Value (OUTVUP=TRUE)**

If OUTVUP=TRUE is set, the same rate of change applies as for OUTVDN.

If the value of OUTV is greater than H_LM, it is limited to H_LM and QH_LM=TRUE is output; if OUTVUP=FALSE is set, QH_LM=FALSE is also set. OUTVUP has lower priority than OUTVDN.

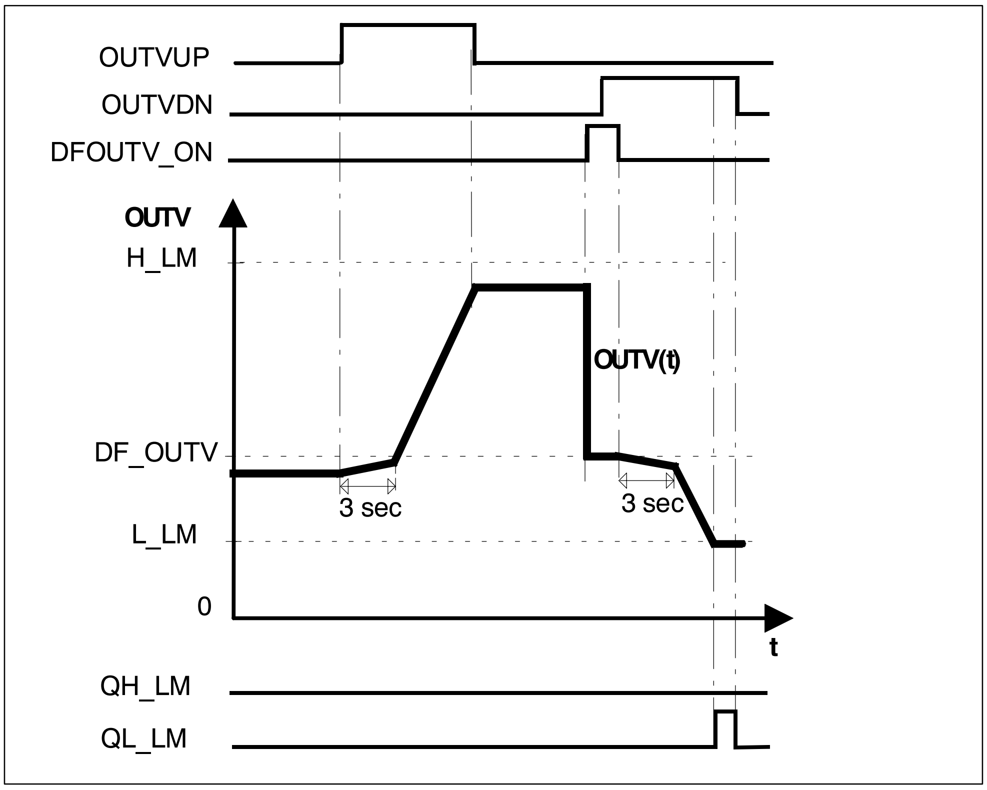


Figure 2-80 Influencing OUTV with OUTVUP, OUTVDN and DFOUTV_ON

Block-Internal Limits

No values are limited internally in the block; the parameters are not checked.

2.1.26 SPLT_RAN: Split Ranging

Application

The block is required to implement a split-range controller.

The manipulated value range of a PID controller is split into several subranges. The block must be called once per subrange and connected to one of the manipulated value processing blocks LMNGEN_C or LMNGEN_S.

Block Diagram

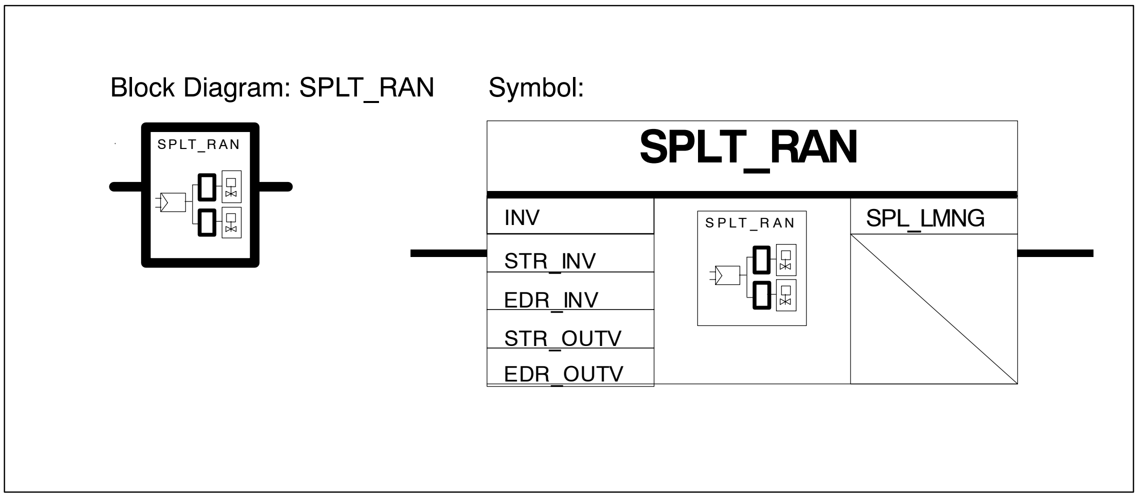


Figure 2-81 SPLT_RAN, Block Diagram and Symbol

Functional Description

An input value within a range limited by STR_INV and EDR_INV is converted to an output value within a range limited by STR_OUTV and EDR_OUTV (see Figure 2-82).

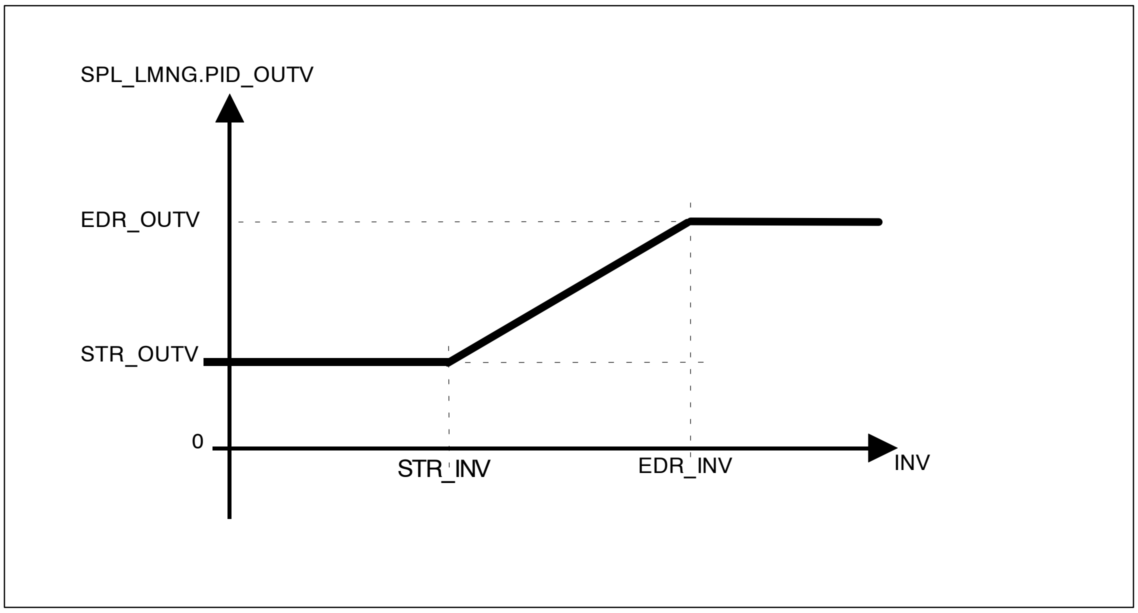


Figure 2-82

Input Parameters

The following table shows the data type and structure of the input parameters of SPLT_RAN.

Table 2-60 Input Parameters of SPLT_RAN

Data Type	Parameter	Comment	Permitted Values	Default
REAL	INV	input variable	technical range of values	0.0
REAL	STR_INV	start of range INV	technical range of values	0.0
REAL	EDR_INV	end of range INV	technical range of values	50.0
REAL	STR_OUTV	start of range OUTV	technical range of values	0.0
REAL	EDR_OUTV	end of range OUTV	technical range of values	100.0

Output Parameters

The following table shows the data type and structure of the output parameters SPLT_RAN.

Table 2-61 Output Parameters of SPLT_RAN

Data Type	Parameter	Comment	Default
STRUC	SPL_LMNG	PID-LMNGEN interface	

Complete Restart

The block has no complete restart routine.

Normal Operation

The block has no modes other than normal operation.

Block-Internal Limits

The values of the parameters are not limited in the block; the parameters are not checked.

Example

The output value of the PID block is distributed on two manipulated value processing blocks LMNGEN_C and LMNGEN_S by SPLT_RAN.

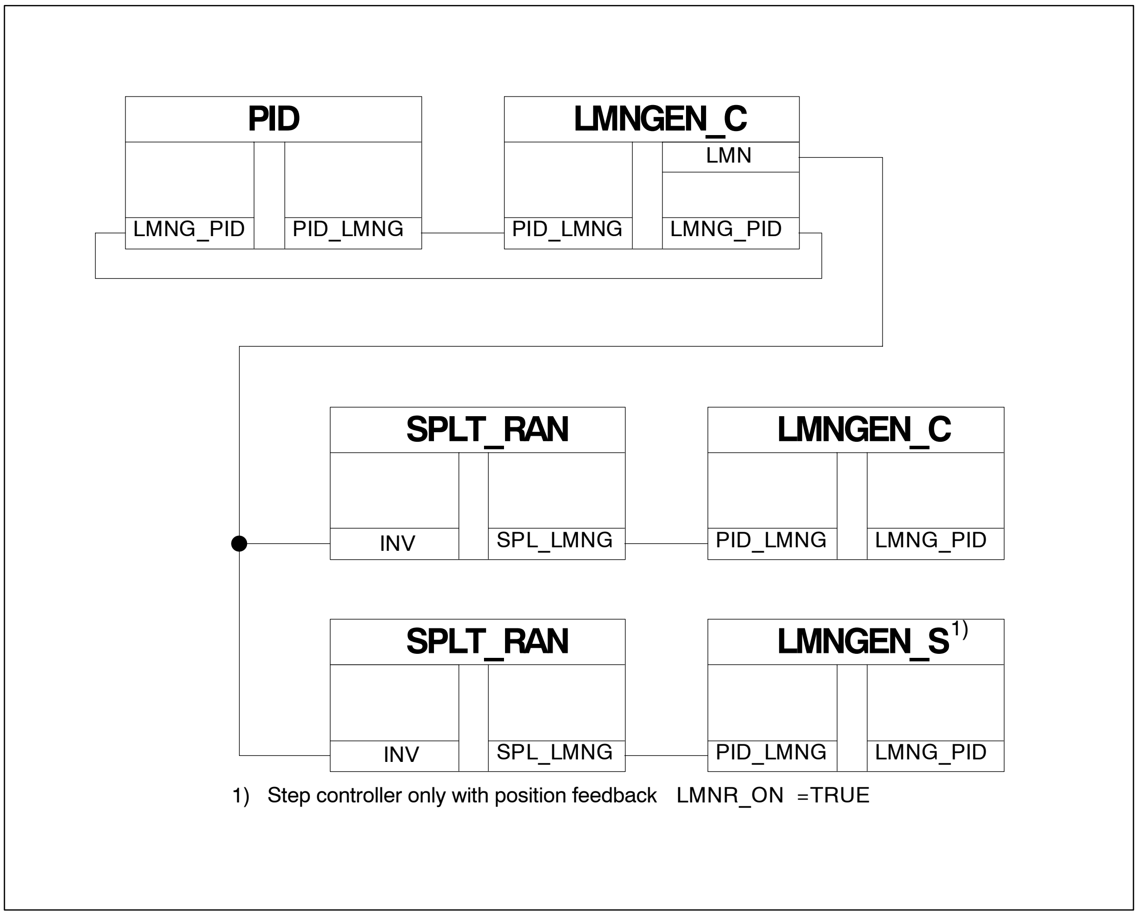


Figure 2-83 Connection of SPLT_RAN with PID and LMNGEN_S

2.1.27 SWITCH: Switch

Application

The block is used as an input and/or output multiplexer of two input/output variables.

Block Diagram

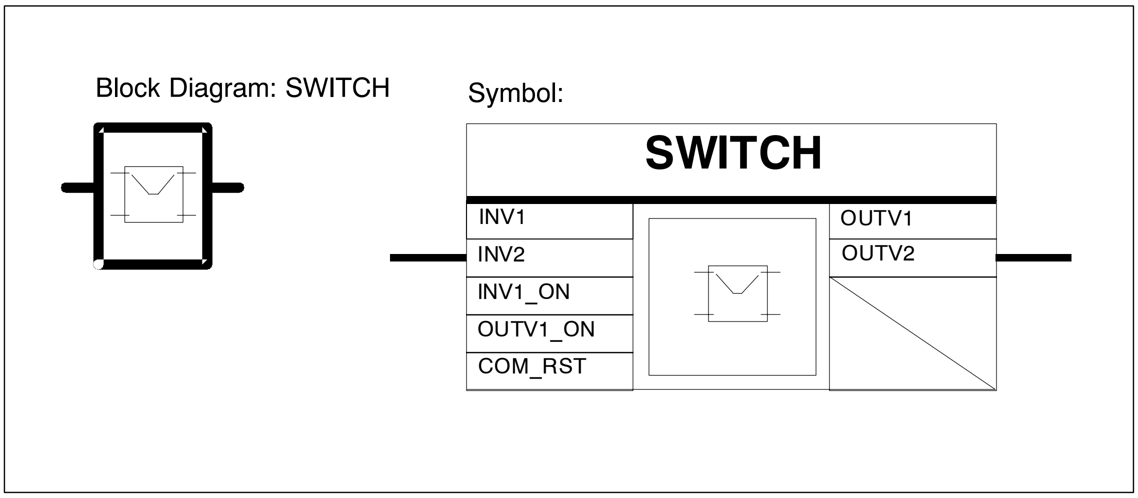


Figure 2-84 SWITCH, Block Diagram and Symbol

Functional Description

The block switches one of two analog input values to one of two output values according to the following table:

INV1_ON	OUTV1_ON	OUTV1	OUTV2
0	0	unchanged	INV2
1	0	unchanged	INV1
0	1	INV2	unchanged
1	1	INV1	unchanged

Input Parameters

The following table shows the data type and structure of the input parameters of SWITCH.

Table 2-62 Input Parameters of SWITCH

Data Type	Parameter	Comment	Permitted Values	Default
REAL	INV1	input variable 1	technical range of values	0.0
REAL	INV2	input variable 2	technical range of values	0.0
BOOL	INV1_ON	connect through INV1		FALSE
BOOL	OUTV1_ON	connect through OUTV1		FALSE
BOOL	COM_RST	complete restart		FALSE

Output Parameters

The following table shows the data type and structure of the output parameters SWITCH.

Table 2-63 Output Parameters of SWITCH

Data Type	Parameter	Comment	Default
REAL	OUTV1	output variable 1	0.0
REAL	OUTV2	output variable 2	0.0

Complete Restart

During a complete restart, OUTV1=0.0 and OUTV2=0.0 are set.

Normal Operation

The block has no modes other than that in normal operation.

Block-Internal Limits

The values of the parameters are not limited in the block; the parameters are not checked.

Examples

3

3.1 Using Modular PID Control

Overview

Using blocks from the ModPID library of Modular PID Control, you can create your own specific controller.

The project zEn28_4_ModCon contains 12 examples of controller structures (EXAMPLE01 to EXAMPLE12). Sections 3.2 to 3.13 describe these 12 examples that are made up of the blocks of the ModPID library as described in Chapter 2.

Examples and Their Uses

Table 3-1 lists the examples supplied in the zEn28_4_ModCon project.

Table 3-1 List of Examples

Example	Function
EXAMPLE01	Fixed Setpoint Controller with Switching Output for Integrating Actuators
EXAMPLE02	Fixed Setpoint Controller with Continuous Output
EXAMPLE03	Fixed Setpoint Controller with Switching Output for Proportional Actuators
EXAMPLE04	Single-Loop Ratio Controller
EXAMPLE05	Multiple-Loop Ratio Controller
EXAMPLE06	Blending Controller
EXAMPLE07	Cascade Controller
EXAMPLE08	Controller with Precontroller
EXAMPLE09	Controller with Feedforward Control
EXAMPLE10	Range Splitting Controller
EXAMPLE11	Override Controller
EXAMPLE12	Multiple Variable Controller

Based on the examples in Table 3-1, you can see the calls and interconnection of the most important blocks.

You can copy the example that comes closest to the controller structure you require as a template and then modify the template by removing or adding block calls and interconnections.

Note

Only examples 1 to 3 include process simulation and can be run without a connection to a process.

Examples 4 to 12 require a connection to a process. Before you can use these examples, the blocks (CRP_IN, LMNGEN_C, LMNGEN_S, SP_GEN ...) must be assigned new parameter values to that the process values are connected through.

You implement your controller structure (user FB) as an FB with local instances of FBs from the ModPID library. Your user FB contains the block call and the interconnection of input and output parameters. You can create your user FBs both with STL and SCL.

You can call these controllers (user FBs) in an organization block suitable for your application.

STL Programming Example

The following example shows how to call the blocks from the ModPID library and interconnect them using STL.

Address	Declaration	Name	Type
0.0	in	SP_UP	BOOL
0.1	in	SP_DOWN	BOOL
2.0	out	OUT	REAL
6.0	stat	DI_SP_GEN	FB 25
46.0	stat	DI_ROC_LIM	FB 22

```
STL                           Explanation
Network 1:
CALL    #DI_SP_GEN            //Block call
      OUTVUP  := #SP_UP
      OUTVDN  :=#SP_DOWN
L       #DI_SP_GEN.OUTV       //Interconnection
T       #DI_ROC_LIM.INV
CALL    #DI_ROC_LIM           //Block call
      OUTV    :=#OUT
BE
```

SCL Programming Example

The following example shows how to call the blocks from the ModPID library and interconnect them using SCL.

```
FUNCTION_BLOCK  User FB
VAR_INPUT
     SP_UP:                         bool := FALSE;
     SP_DOWN:                       bool := FALSE;
END_VAR
VAR_OUTPUT
     OUT:                           real := 0.0;
END_VAR
VAR
     DI_SP_GEN:    SP_GEN;
     DI_ROC_LIM:   ROC_LIM;
END_VAR
BEGIN
     DI_SP_GEN(                     //Block call + interconnection
     OUTVUP  := SP_UP,
     OUTVDN  :=SP_DOWN);
     DI_ROC_LIM(                    //Block call + interconnection
     INV := DI_SP_GEN.OUTV);
     OUT := DI_ROC_LIM.OUTV;    //Interconnection
END_FUNCTION_BLOCK
```

Practising with the Examples

The examples 1 to 3 contain a complete control loop. They are particularly suitable for practising.

Using the standard tool “Monitoring and Modifying Variables”, it is simple to modify control parameters and you can then watch the results of the changes in the reaction of the simulated control loop.

The configuration tool provides a graphic interface with the loop monitor and curve recorder and allows process identification.

3.2 Example 1: Fixed Setpoint Controller with Switching Output for Integrating Actuators with Process Simulation

Overview

Example 1 is called EXAMPLE01 and consists of a PID step controller (fixed setpoint controller with switching output for integrating actuators) and a simulated process.

Control Loop

Figure 3-1 shows the complete control loop of example 1.

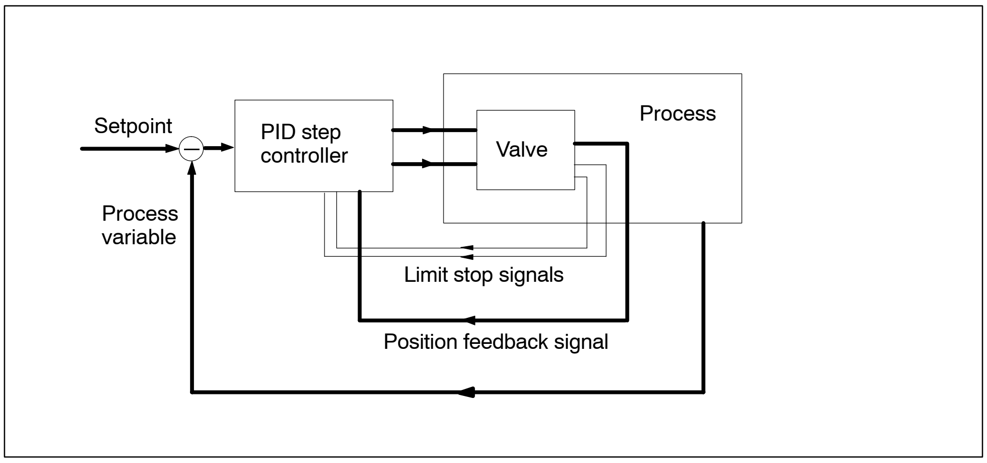


Figure 3-1 Control Loop of Example 1

Note

You have to set the parameter DB50.DI_LMNGEN_S.MAN_ON to FALSE to be able to work with the loop monitor function of the configuration tool.

Block Call and Interconnection

Figure 3-2 shows the block call and the interconnection of example 1.

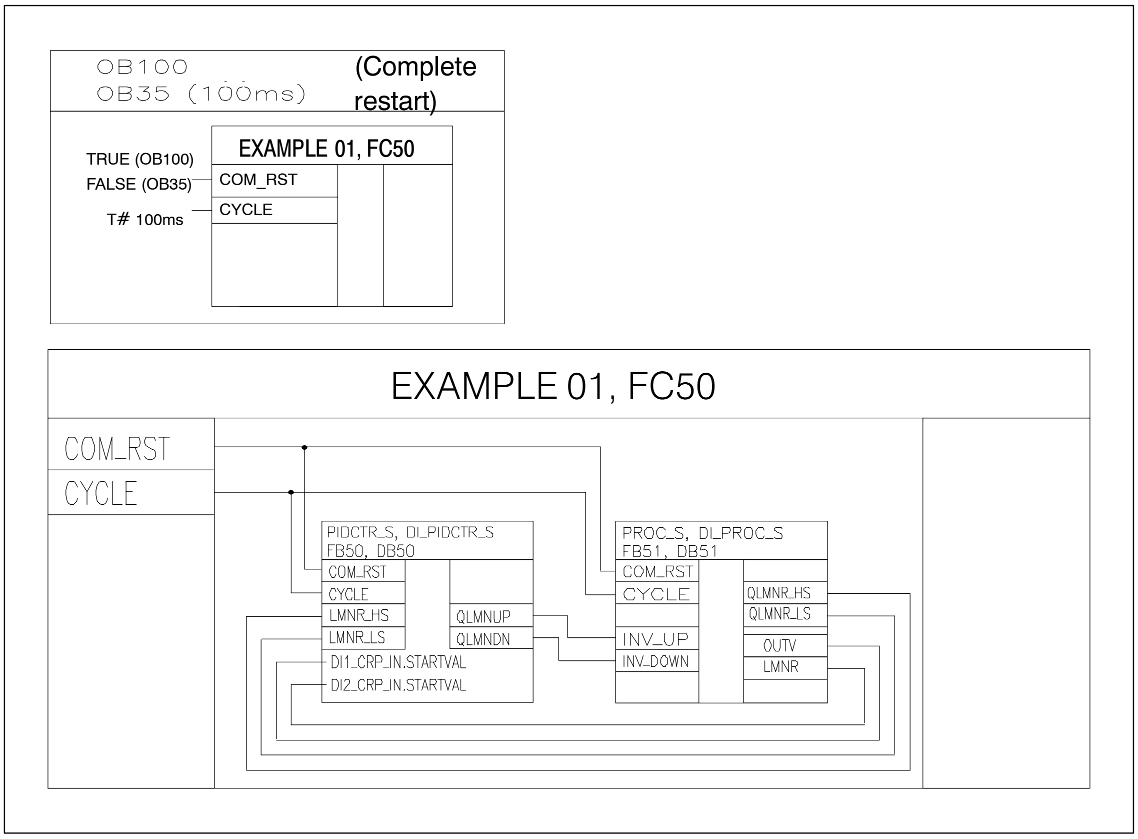


Figure 3-2 Block call and Interconnection of Example 1

3.2.1 PIDCTR_S: Fixed Setpoint Controller with Switching Output for Integrating Actuators

Application

The PIDCTR_C block implements a PID step controller for integrating actuators (for example motor-driven valves in industrial processes). Figure 3-3 shows the block interconnections of PIDCTR_S.

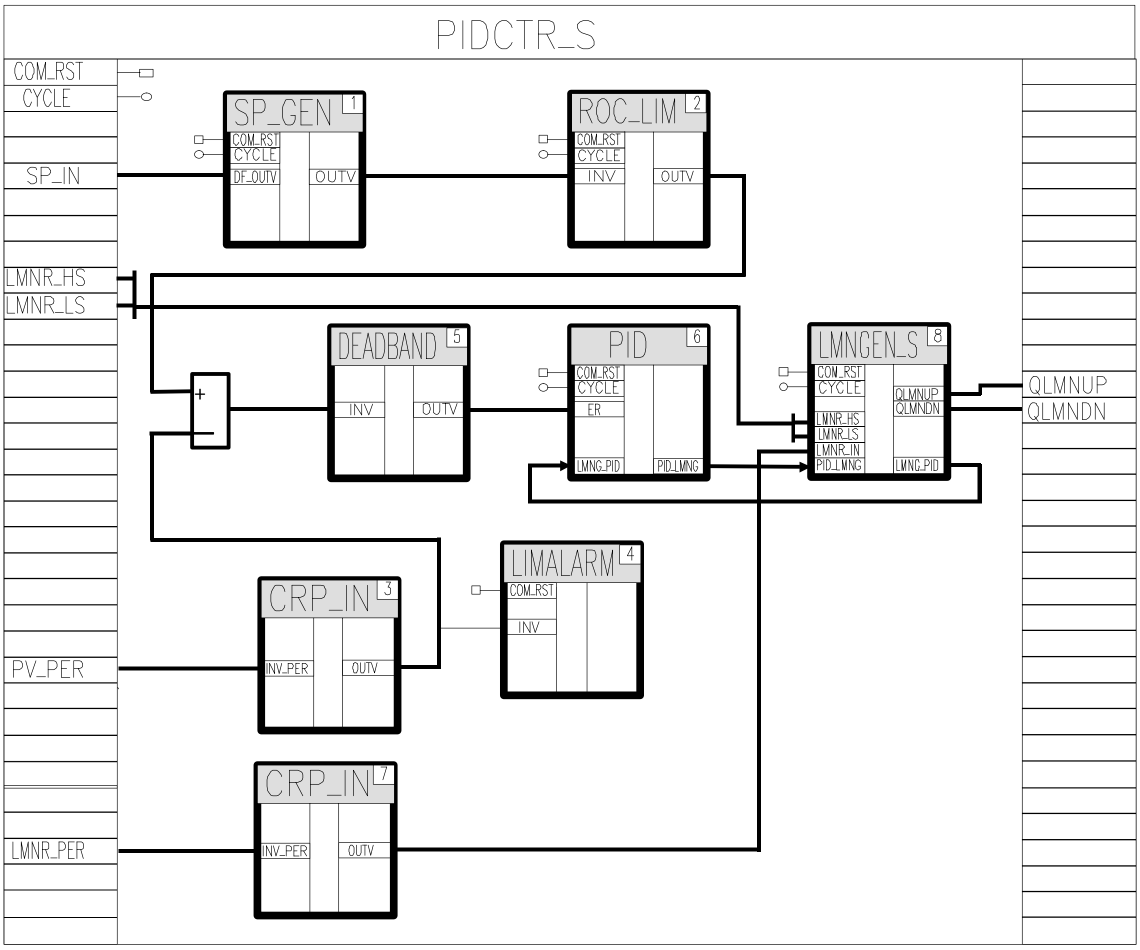


Figure 3-3 Block Interconnections of PIDCTR_S

Functional Description

The setpoint generator SP_GEN sets the setpoint whose rate of change is limited by the limiter ROC_LIM. The peripheral process variable is converted to a floating-point value by CRP_IN and monitored by the limit value monitor LIMALARM to check that it does not exceed selected limit values. The error signal is routed to the PID algorithm via a DEADBAND. The position feedback signal is read in via a second CRP_IN block. The manipulated value processing block LMNGEN_S sets the output signals QLMNUP and QLMNDN.

Complete Restart

During a complete restart, each block is called individually. Blocks with a complete restart routine run through this routine.

3.2.2 PROC_S: Process for Step Controllers

Application

The PROC_S block simulates an integrating actuator with a 3rd-order time lag.

Figure 3-4 shows the block diagram of PROC_S.

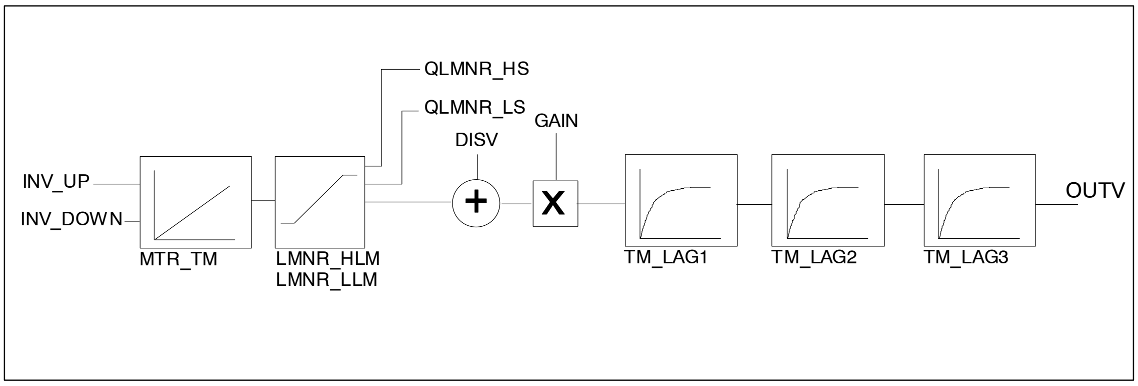


Figure 3-4 Block Diagram of PROC_S

Functional Description

The block simulates an integrating actuator and three 1st-order time lags in series. The disturbance variable **DISV** is always added to the output of the actuating valve. The motor actuating time **MTR_TM** is the time required by the valve from limit stop to limit stop.

Complete Restart

During a complete restart, the output variable **OUTV** and the internally stored values are all set to 0.

3.3 Example 2: Fixed Setpoint Controller with Continuous Output with Process Simulation

Overview

Example 2 is called EXAMPLE02 and consists of a continuous PID controller and a simulated process.

Control Loop

Figure 3-5 shows the complete control loop of example 2.

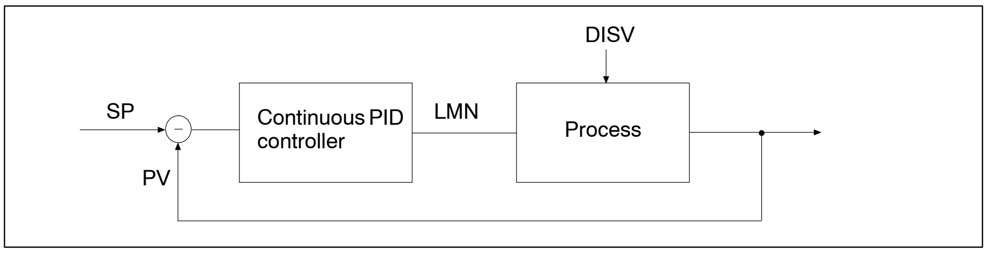


Figure 3-5 Control Loop of Example 2

Block Call and Interconnection

Figure 3-6 shows the block call and the interconnection of Example 2.

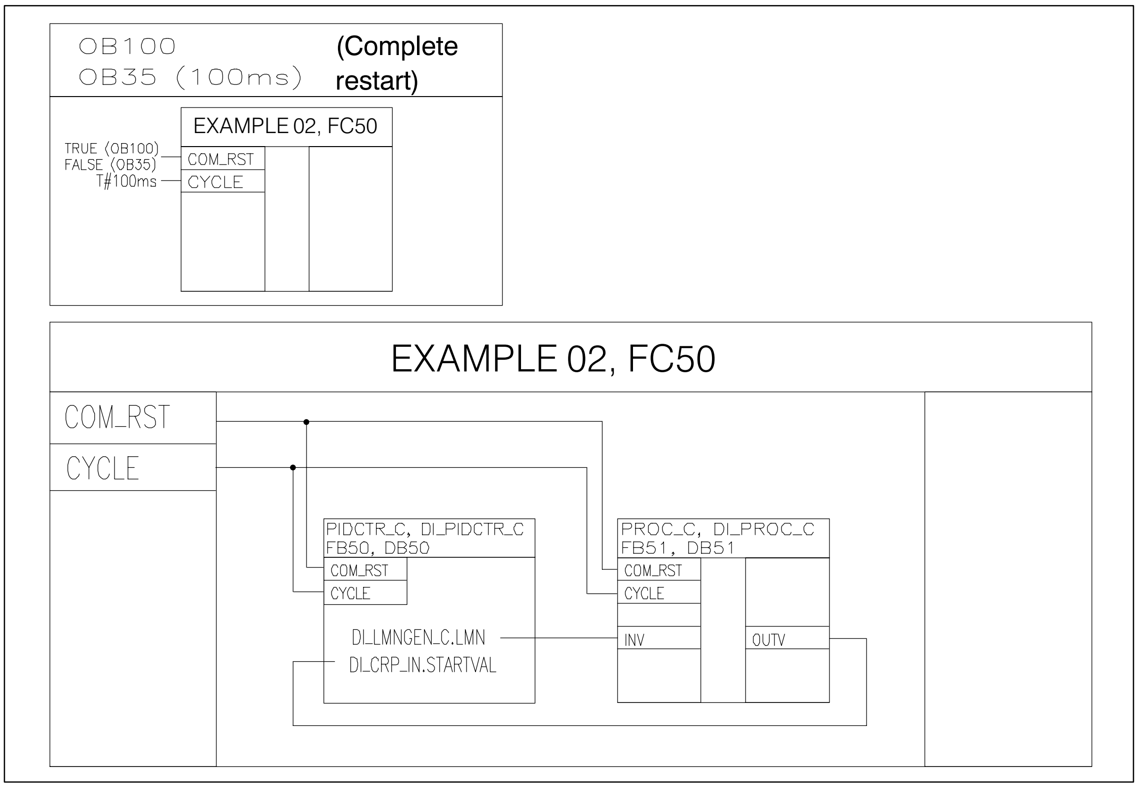


Figure 3-6 Block call and Interconnection of Example 2

3.3.1 PIDCTR_C: Fixed Setpoint Controller with Continuous Output for Integrating Actuators

Application

The PIDCTR_C block is used as a fixed setpoint controller with continuous output. Figure 3-7 shows the block interconnections of PIDCTR_C.

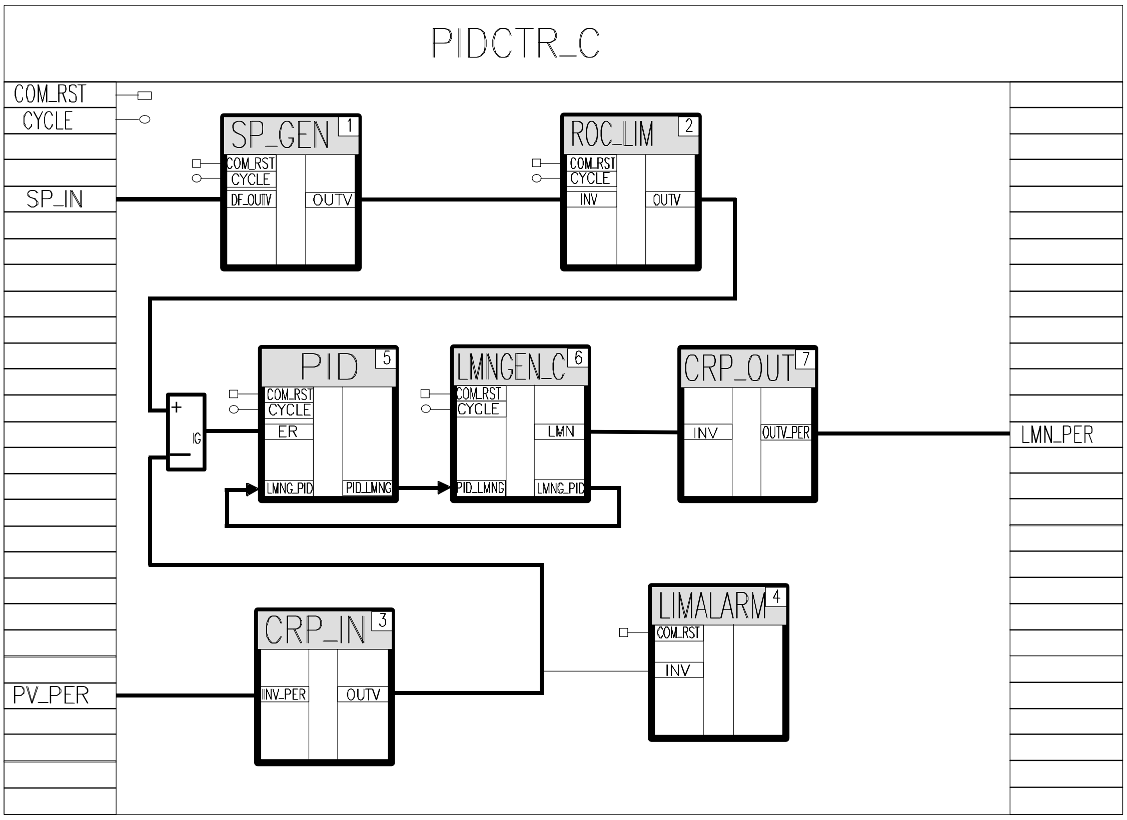


Figure 3-7 Block Interconnections of PIDCTR_C

Functional Description

The setpoint generator SP_GEN sets the setpoint whose rate of change is limited by the limiter ROC_LIM. The peripheral process variable is converted to a floating-point value by CRP_IN and monitored by the limit value monitor LIMALARM to check that it does not exceed selected limit values. The error signal is routed to the PID algorithm. The manipulated value processing block LMNGEN_C generates the analog manipulated variable LMN that is converted to the peripheral format by CRP_OUT.

Complete Restart

During a complete restart, each block is called individually. Blocks with a complete restart routine run through this routine.

3.3.2 PROC_C: Process for Continuous Controller

Application

The block PROC_C simulates a 3rd-order time lag.

Figure 3-8 shows the block diagram of PROC_C.

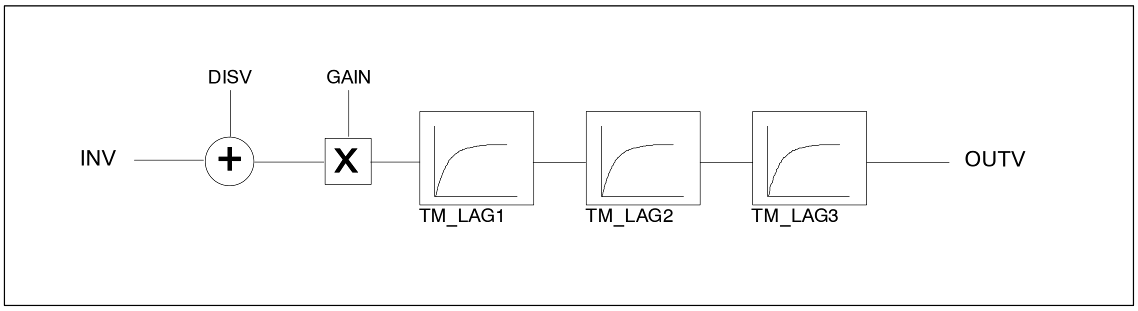


Figure 3-8 Block Diagram of PROC_C

Functional Description

The block simulates three 1st-order time lags in series. The disturbance variable **DISV** is always added to the input INV.

Complete Restart

During a complete restart, the output variable **OUTV** and the internally stored values are all set to 0.

3.4 Example 3: Fixed Setpoint Controller with Switching Output for Proportional Actuators with Process Simulation

Overview

Example 3 is called EXAMPLE03 and consists of a continuous PID controller with pulse duration modulation and a simulated process.

Control Loop

Figure 3-9 shows the complete control loop of example 3.

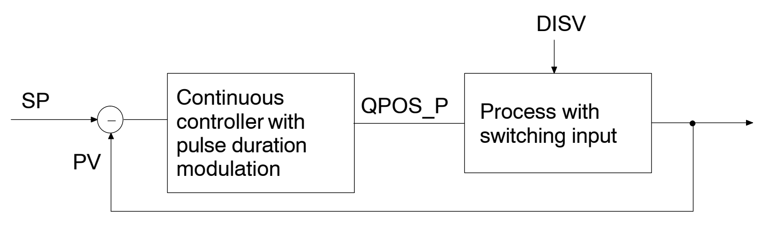

DISV
SP
−
PV
Continuous controller with pulse duration modulation
QPOS_P
Process with switching input

Figure 3-9 Control Loop of Example 3

Block Call and Interconnection

Figure 3-10 shows the block call and the interconnection of Example 3.

Note

Die Zykluszeit des OB35 müssen Sie mittels “HW Konfig: Hardware konfigurieren” auf 10 ms einstellen.

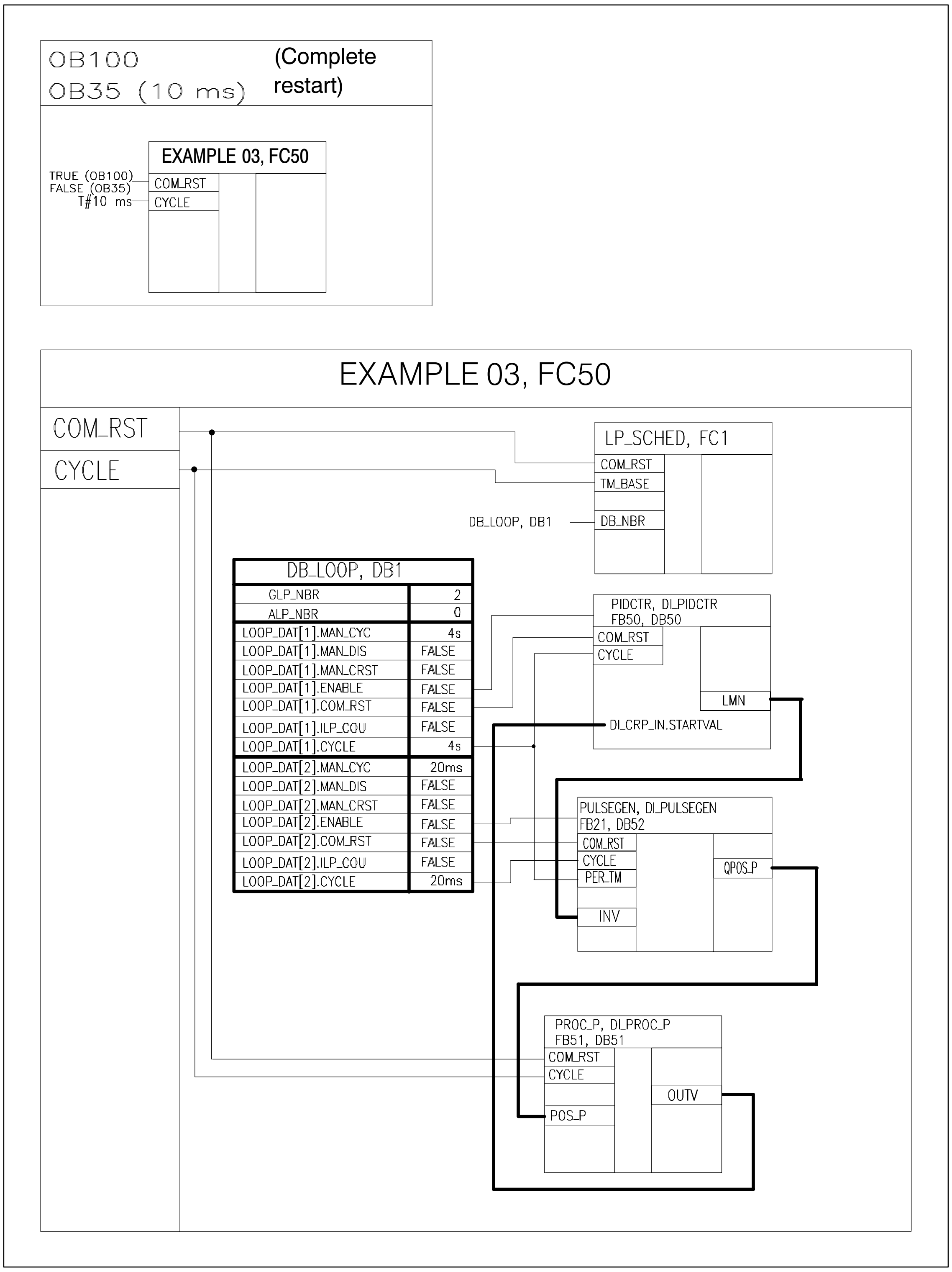


Figure 3-10 Block Call and Interconnection of Example 3

3.4.1 PIDCTR: Primary Controller for a Continuous Controller with Pulse Generator

Application

The block PIDCTR implements a PID controller with continuous output. It is used to calculate the analog manipulated value within a pulse-break controller. It is also used as a primary controller in ratio controls, blending controls, and cascade controls. Figure 3-11 shows the block interconnections of PIDCTR.

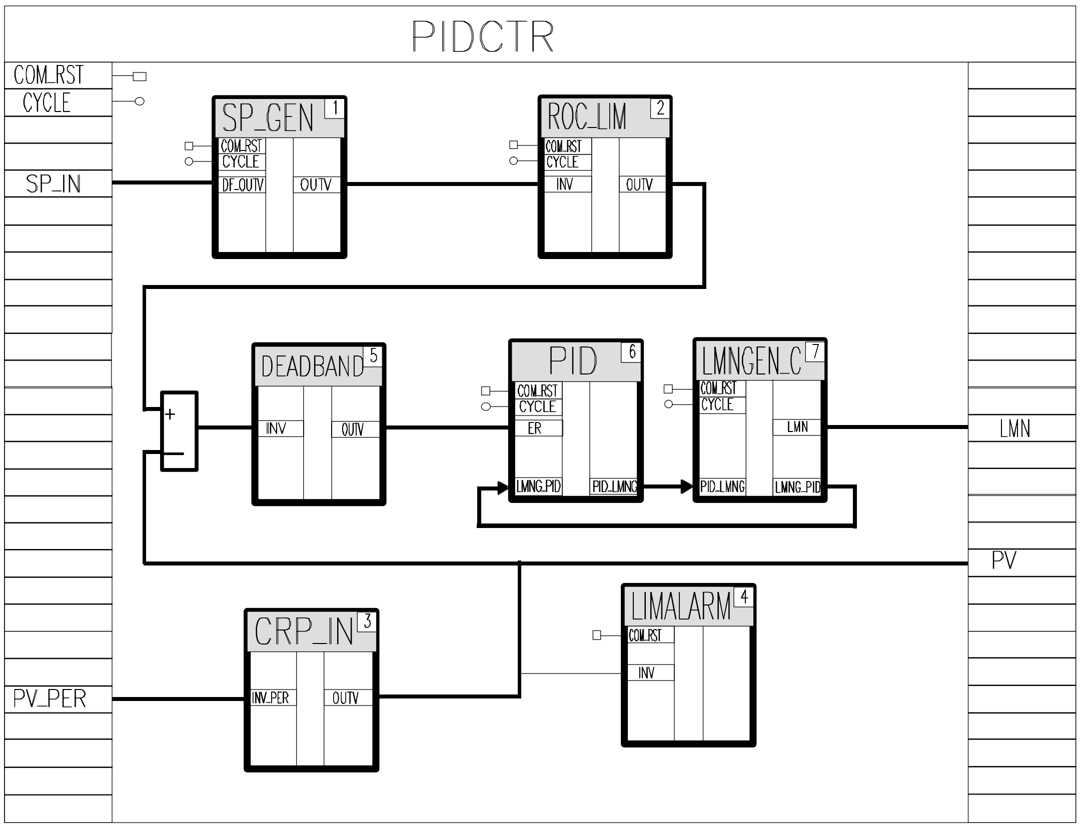


Figure 3-11 Block Interconnections of PIDCTR

Functional Description

The setpoint generator SP_GEN sets the setpoint whose rate of change is limited by the limiter ROC_LIM. The peripheral process variable is converted to a floating-point value by CRP_IN and monitored by the limit value monitor LIMALARM to check that it does not exceed selected limit values. The error signal is routed to the PID algorithm. The manipulated value processing block LMNGEN_C generates the analog manipulated variable LMN.

Complete Restart

During a complete restart, each block is called individually. Blocks with a complete restart routine run through this routine.

3.4.2 PROC_P: Process for a Continuous Controller with Pulse Generator

Application

The PROC_P block simulates a continuous actuator valve with a digital input and a 3rd-order time lag.

Figure 3-12 shows the block diagram of PROC_P.

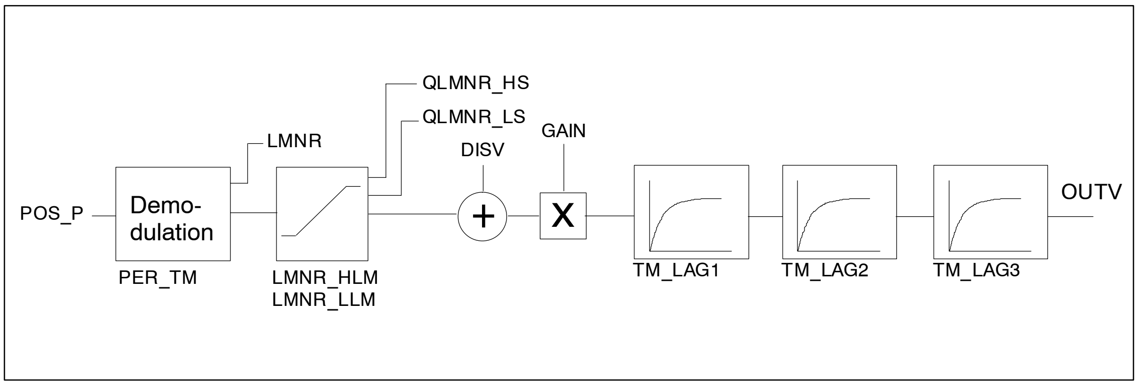


Figure 3-12 Block Diagram of PROC_P

Functional Description

The block converts the binary input values of the pulse duration modulation into continuous analog values and, after the disturbance variable has been added, delays the output signal with three 1st-order time lags.

Complete Restart

During a complete restart, the output variable **OUTV** and the internally stored values are all set to 0.

3.5 Example 4: Single-Loop Ratio Controller (RATIOCTR)

Overview

Example 4 is called EXAMPLE04 and is a single-loop ratio controller. There is no simulated process with this example.

Control Loop

Figure 3-13 shows the application of Example 4 in a complete control loop.

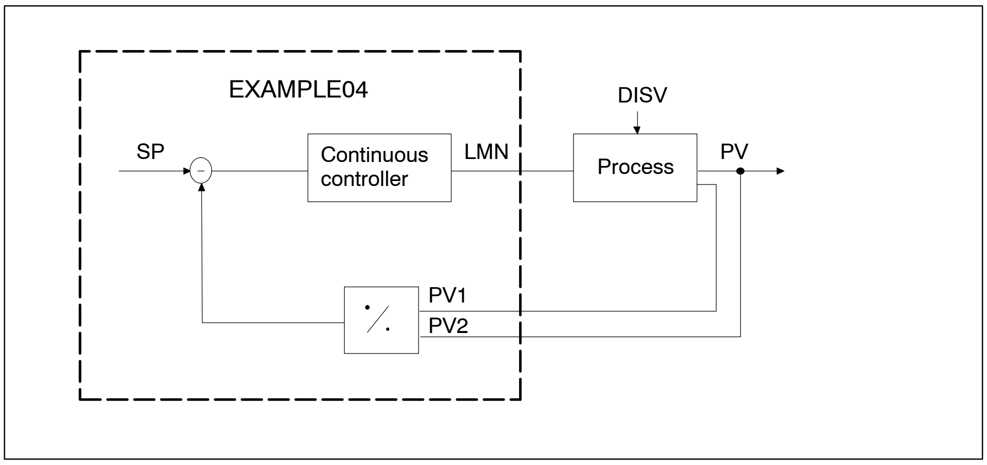


Figure 3-13 Control Loop with Example 4

Block Call

Figure 3-14 shows the block call of Example 4.

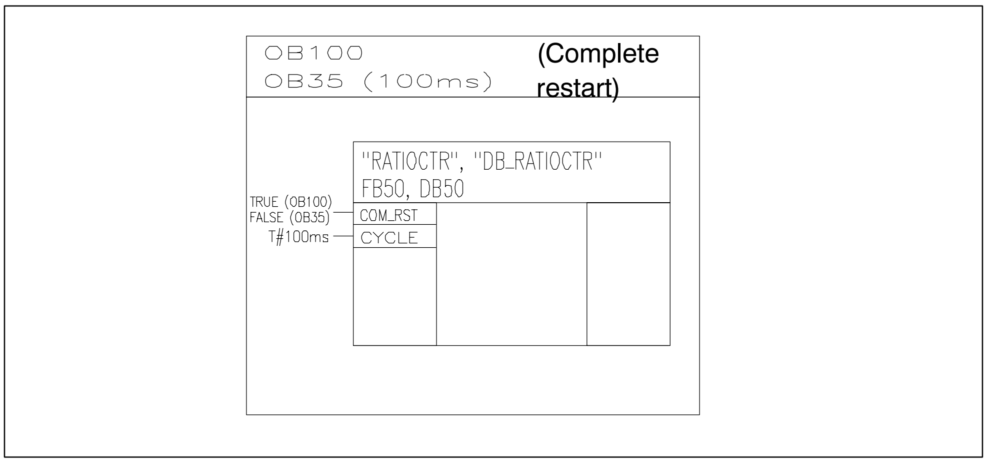


Figure 3-14 Block Call of Example 4

Application

The block implements a single loop ratio controller for continuous actuators. Figure 3-15 shows the block interconnections of RATIOCTR.

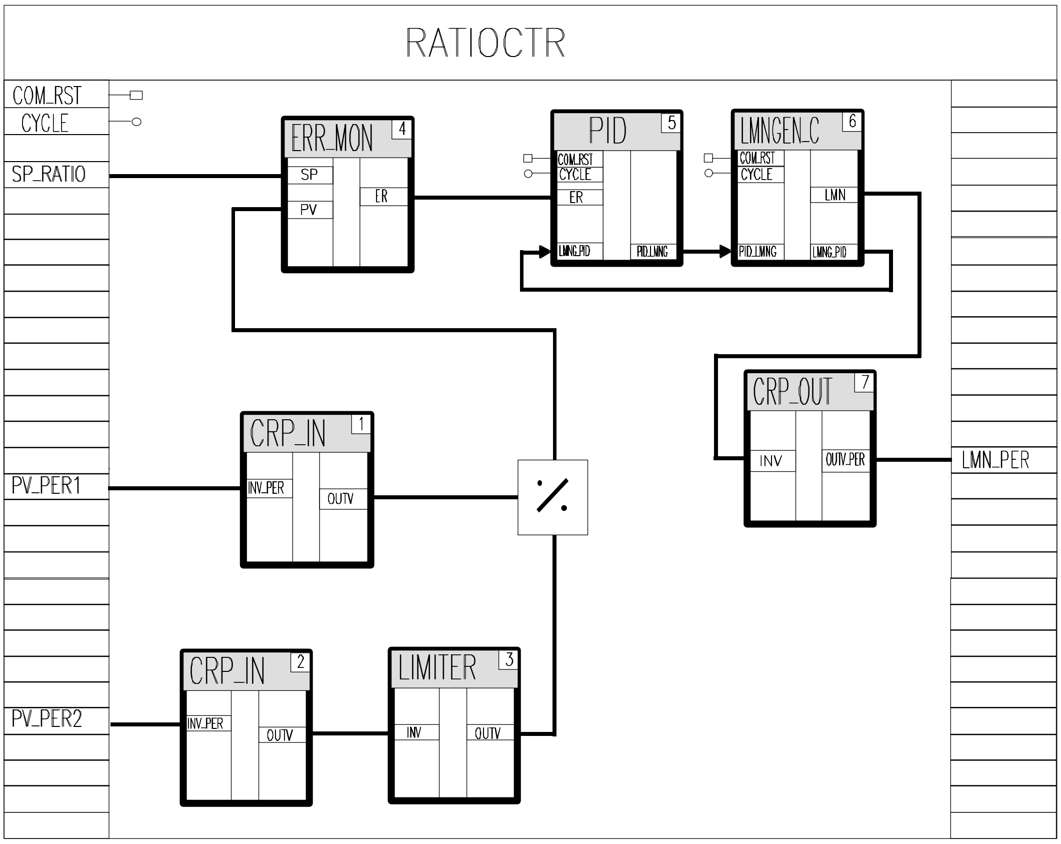


Figure 3-15 Block Interconnections of RATIOCTR

Functional Description

The ratio setpoint is set with the input parameter SP_RATIO. The peripheral process variables PV_PER1 and PV_PER2 are converted to floating-point values by CRP_IN and the ratio is formed. The floating-point value of PV_PER2 is limited by the LIMITER so that no division by zero is possible. The error signal is routed to the PID algorithm PID. The manipulated value processing block LMNGEN_C generates the analog manipulated variable LMN that is converted to the peripheral format by CRP_OUT.

Complete Restart

During a complete restart, each block is called individually. Blocks with a complete restart routine run through this routine.

3.6 Example 5: Multiple-Loop Ratio Controller

Overview

Example 5 is called EXAMPLE05 and is a multiple loop ratio controller. It contains a primary controller and three secondary controllers. There is no simulated process with this example.

Control Loop

Figure 3-16 shows the application of Example 5 in a complete control loop.

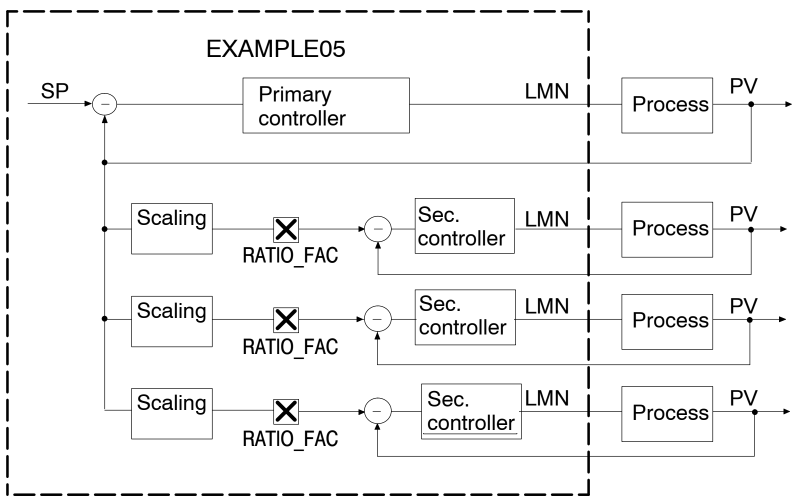
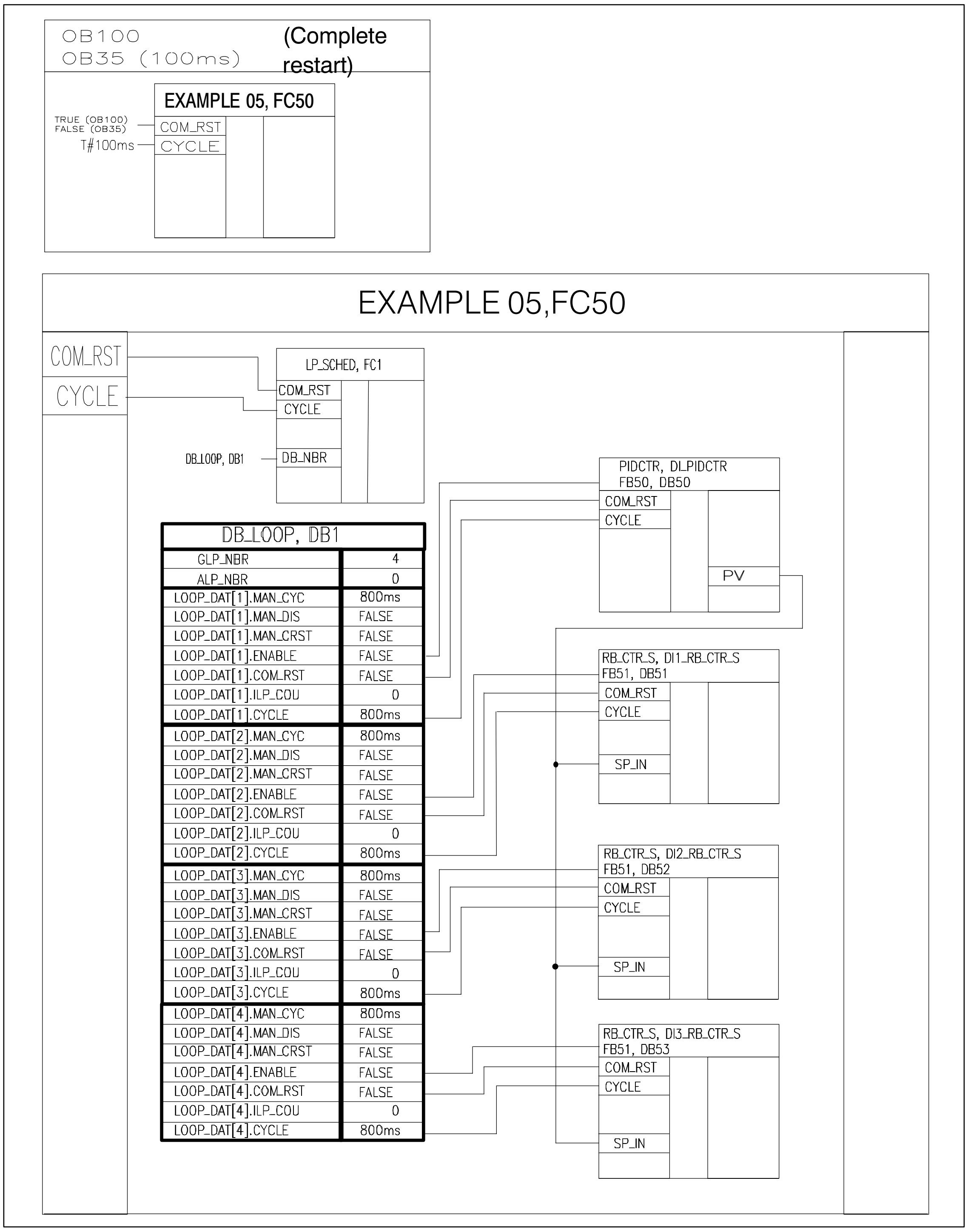


Figure 3-16 Control Loop with Example 5

Block Call and Interconnection

Figure 3-17 shows the block call and the interconnection of Example 5.



DB_LOOP, DB1	
GLP_NBR	4
ALP_NBR	0
LOOP_DAT[1].MAN_CYC	800ms
LOOP_DAT[1].MAN_DIS	FALSE
LOOP_DAT[1].MAN_CRST	FALSE
LOOP_DAT[1].ENABLE	FALSE
LOOP_DAT[1].COM_RST	FALSE
LOOP_DAT[1].ILP_COU	0
LOOP_DAT[1].CYCLE	800ms
LOOP_DAT[2].MAN_CYC	800ms
LOOP_DAT[2].MAN_DIS	FALSE
LOOP_DAT[2].MAN_CRST	FALSE
LOOP_DAT[2].ENABLE	FALSE
LOOP_DAT[2].COM_RST	FALSE
LOOP_DAT[2].ILP_COU	0
LOOP_DAT[2].CYCLE	800ms
LOOP_DAT[3].MAN_CYC	800ms
LOOP_DAT[3].MAN_DIS	FALSE
LOOP_DAT[3].MAN_CRST	FALSE
LOOP_DAT[3].ENABLE	FALSE
LOOP_DAT[3].COM_RST	FALSE
LOOP_DAT[3].ILP_COU	0
LOOP_DAT[3].CYCLE	800ms
LOOP_DAT[4].MAN_CYC	800ms
LOOP_DAT[4].MAN_DIS	FALSE
LOOP_DAT[4].MAN_CRST	FALSE
LOOP_DAT[4].ENABLE	FALSE
LOOP_DAT[4].COM_RST	FALSE
LOOP_DAT[4].ILP_COU	0
LOOP_DAT[4].CYCLE	800ms

Figure 3-17 Block Call and Interconnection of Example 5

Primary Controller

The primary controller is block PIDCTR from Example 3. Its functions are described in Section 3.4.1 on Page 3-14.

Secondary controller

The secondary controller is block RB_CTR_S. This block is a PID step controller for integrating actuators that can be used as a secondary controller in a multiple loop ratio or blending controller. Figure 3-18 shows the block interconnections of RB_CTR_S.

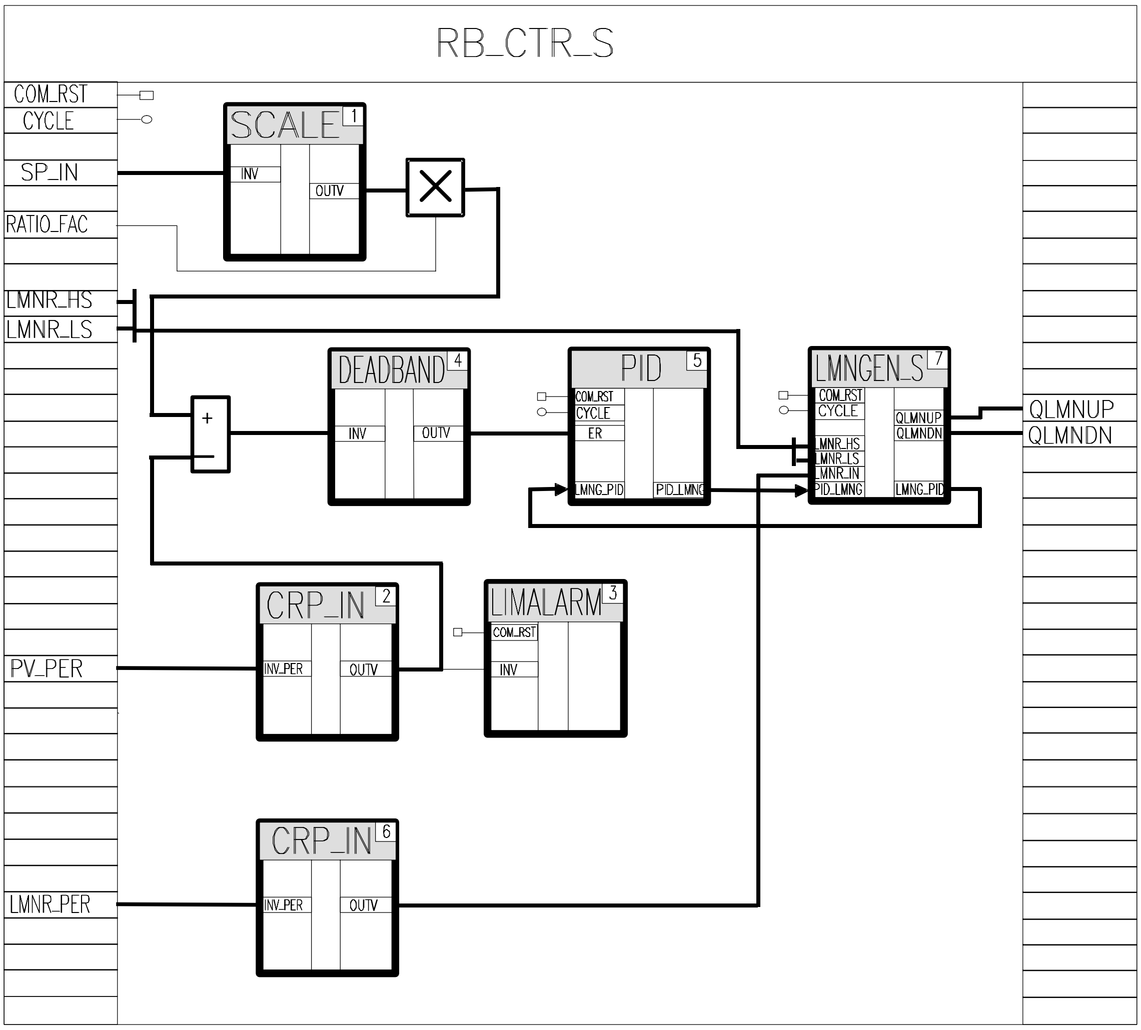


Figure 3-18 Block Interconnections of RB_CTR_S

Functional Description of the Secondary controllers

The functionality of the secondary controller RB_CTR_S is analogous to that of the step controller PIDCTR_S from Example 1 (see page 3-4). The input of the secondary controller is adapted to the output of the process by a scaling factor and multiplied by selected ratio or blending factor.

Complete Restart

During a complete restart, each block is called individually. Blocks with a complete restart routine run through this routine.

3.7 Example 6: Blending Controller

Overview

Example 6 is called EXAMPLE06 and is a blending controller. It includes a primary controller and three secondary controllers. There is no simulated process with this example.

Control Loop

Figure 3-19 shows the application of Example 6 in a complete control loop.

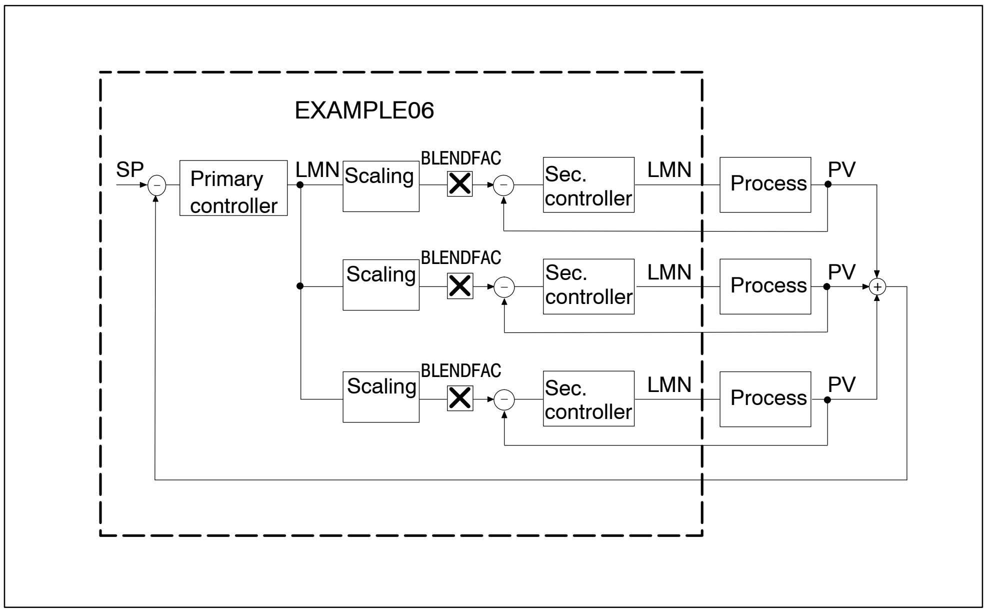


Figure 3-19 Control Loop with Example 6

Block Call and Interconnection

Figure 3-20 shows the block call and the interconnection of Example 6.

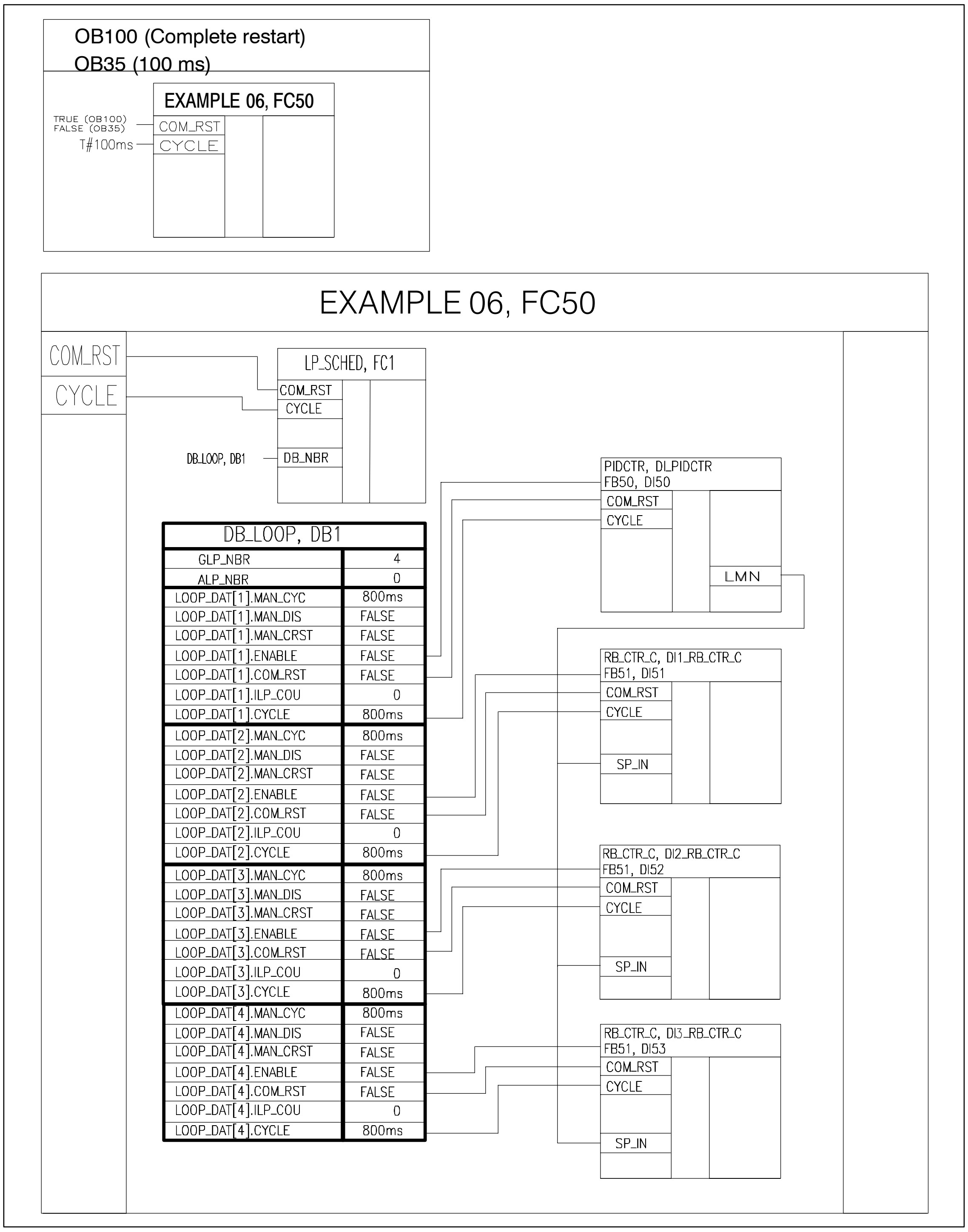


Figure 3-20 Block Call and Interconnection of Example 6

Primary controller

The primary controller is block PIDCTR from Example 3. Its functions are described in Section 3.4.1 on page 3-14.

Secondary Controller

The secondary controller is block RB_CTR_C. This block is a continuous PID controller that can be used as a secondary controller in a multiple loop ratio or blending controller. Figure 3-21 shows the block interconnections of RB_CTR_C.

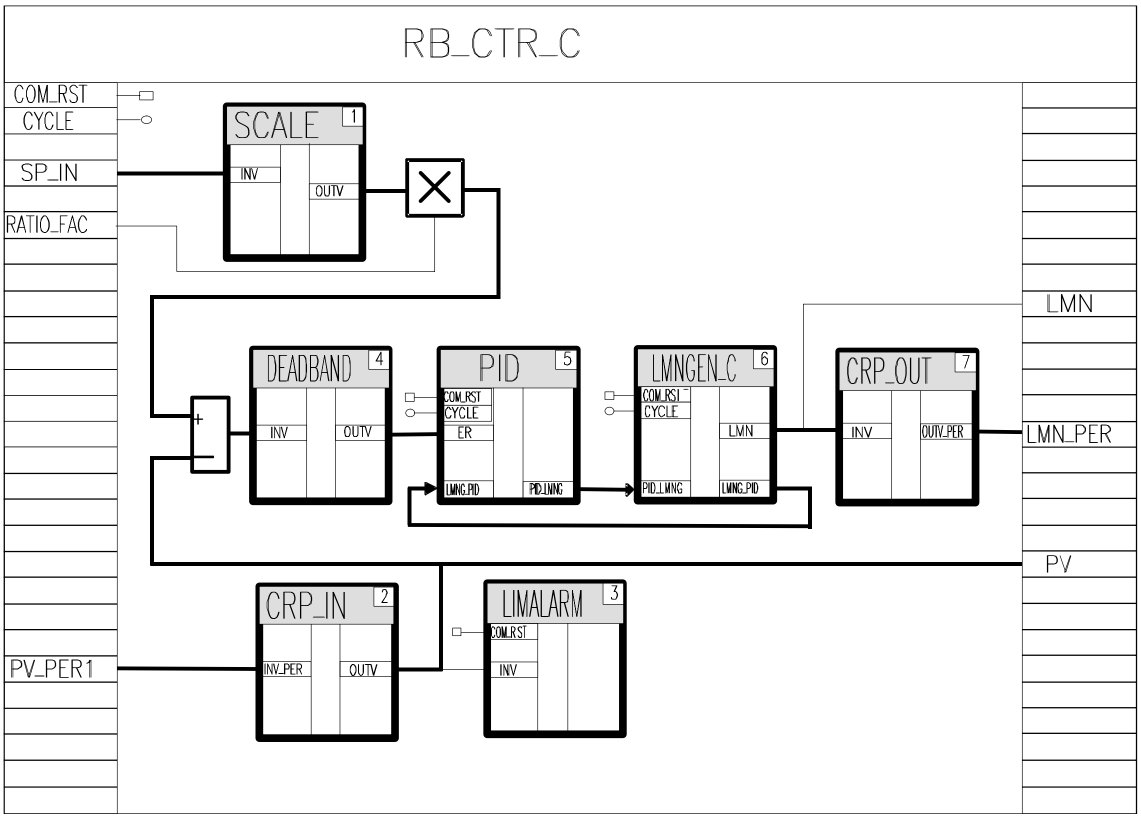


Figure 3-21 Block Interconnections of RB_CTR_C

Functional Description of the Secondary controllers

The functionality of the secondary controller RB_CTRL_C is analogous to that of the continuous PID controller PIDCTR_C from Example 2 (see page 3-9). The input of the secondary controller is adapted to the output of the process by a scaling factor and multiplied by selected ratio or blending factor.

Complete Restart

During a complete restart, each block is called individually. Blocks with a complete restart routine run through this routine.

3.8 Example 7: Cascade Controller

Overview

Example 7 is called EXAMPLE07 and is a cascade controller. It includes a primary controller and a secondary controller. There is no simulated process with this example.

Control Loop

Figure 3-22 shows the application of Example 7 in a complete control loop.

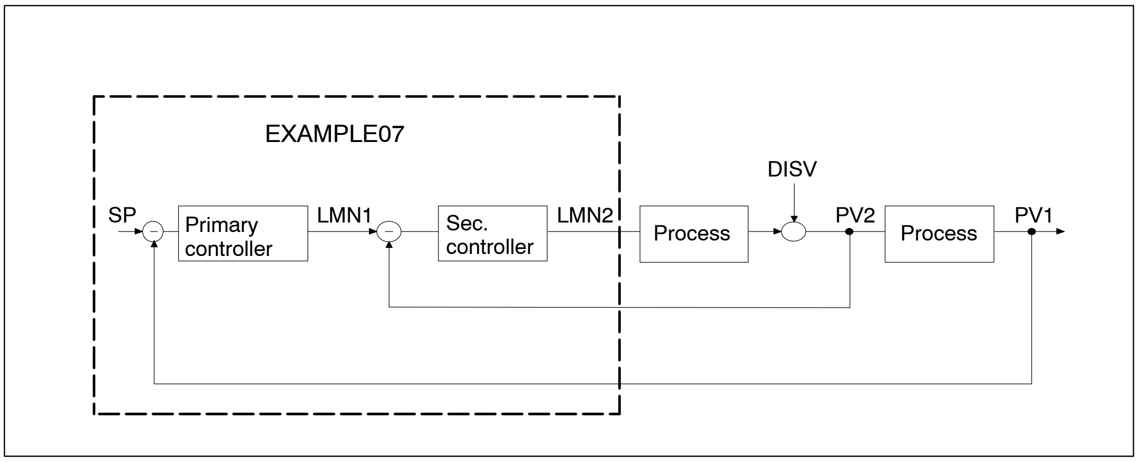


Figure 3-22 Control Loop with Example 7

Block Call and Interconnection

Figure 3-23 shows the block call and the interconnection of Example 7.

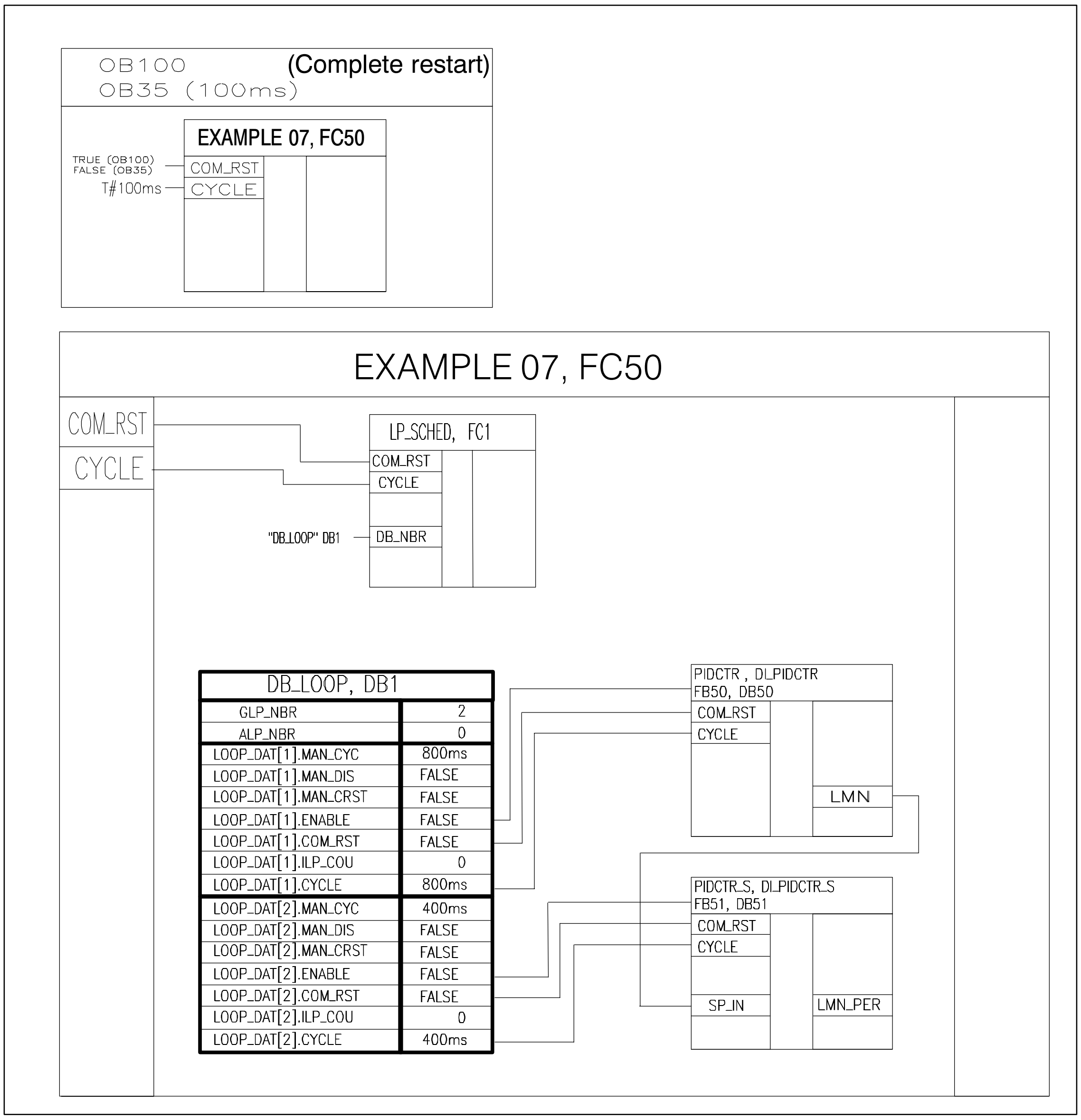


Figure 3-23 Block Call and Interconnection of Example 7

Primary controller

The primary controller is block PIDCTR from Example 3. Its functions are described in Section 3.4.1.

Secondary controller

The secondary controller is block PIDCTR_S from Example 1. Its functions are described in Section 3.2.1 on page 3-6.

3.9 Example 8: Controller with Precontroller (CTRC_PRE)

Overview

Example 8 is called EXAMPLE08 and is a controller with precontroller. There is no simulated process with this example.

Control Loop

Figure 3-24 shows the application of Example 8 in a complete control loop.

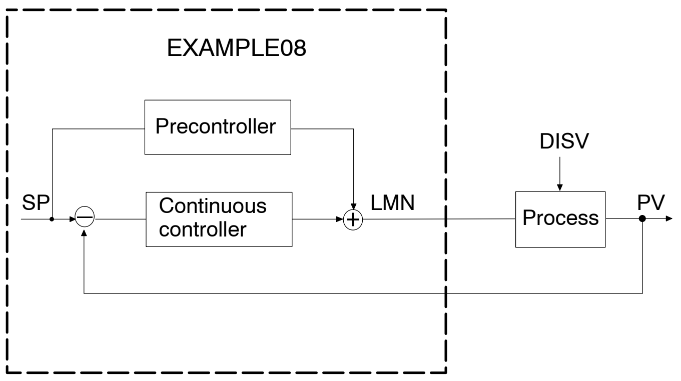


Figure 3-24 Control Loop with Example 8

Block Call

Figure 3-25 shows the block call of Example 8.

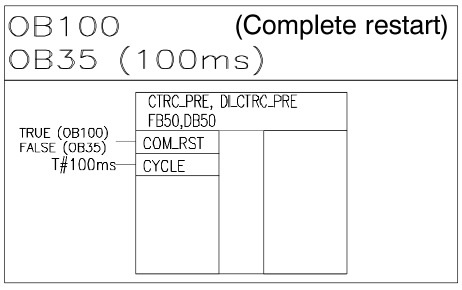


Figure 3-25 Block Call of Example 8

Application

The block CTRC_PRE is a continuous PID controller with precontroller. Figure 3-26 shows the block interconnections of CTRC_PRE.

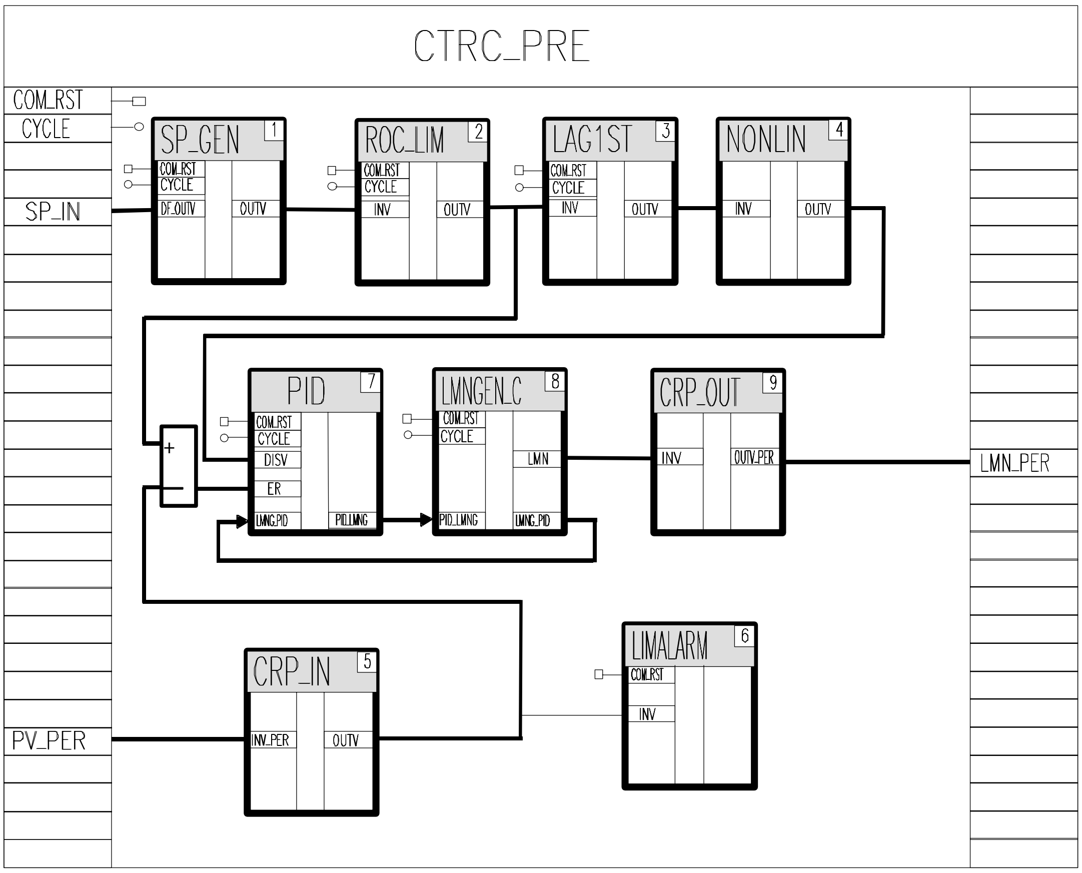


Figure 3-26 Block Interconnections of CTRC_PRE

Functional Description

The functionality of the controller with precontroller is analogous to that of the fixed setpoint controller with continuous output PIDCTR_C from Example 2. The precontroller consists of a 1st-order time lag with a static, non-linear characteristic parallel to the PID algorithm.

Complete Restart

During a complete restart, each block is called individually. Blocks with a complete restart routine run through this routine.

3.10 Example 9: Controller with Feedforward Control (CTR_C_FF)

Overview

Example 9 is called EXAMPLE09 and is a controller with feedforward control. There is no simulated process with this example.

Control Loop

Figure 3-27 shows the application of Example 9 in a complete control loop.

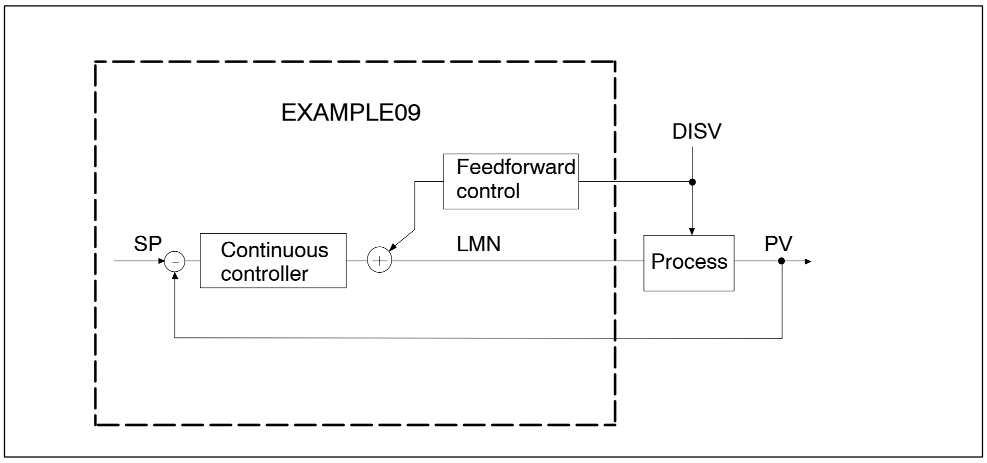


Figure 3-27 Control Loop with Example 9

Block Call

Figure 3-28 shows the block call of Example 9.

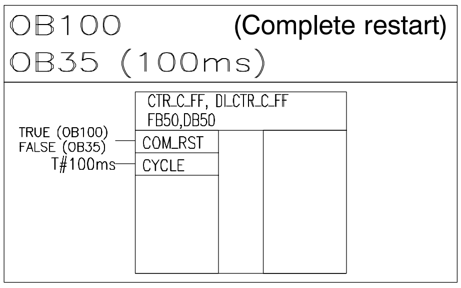


Figure 3-28 Block Call of Example 9

Application

The block CRT_C_FF is a PID controller with feedforward Control for continuous actuators. Figure 3-29 shows the block interconnections of CTR_C_FF.

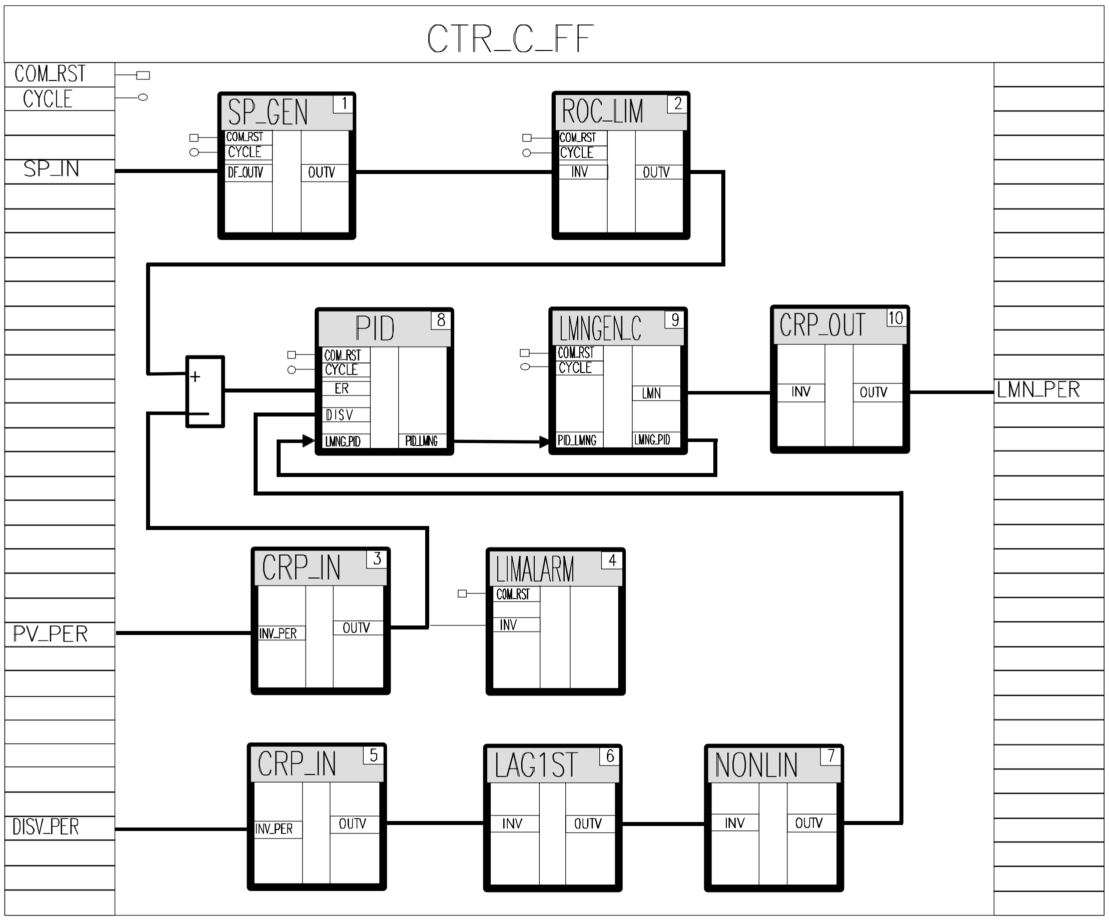


Figure 3-29 Block Interconnections of CTR_C_FF

Functional Description

The setpoint generator SP_GEN sets the setpoint whose rate of change is limited by the limiter ROC_LIM. The peripheral process variable is converted to a floating-point value by CRP_IN and monitored by the limit value monitor LIMALARM to check that it does not exceed selected limit values. The error signal is routed to the PID algorithm. The peripheral disturbance value is converted to a floating-point value by CRP_IN, filtered with LAG1ST and linearized. The manipulated value processing block LMNGEN_C generates the analog manipulated variable LMN that is converted to peripheral format by CRP_OUT.

Complete Restart

During a complete restart, each block is called individually. Blocks with a complete restart routine run through this routine.

3.11 Example 10: Range Splitting Controller (SPLITCTR)

Overview

Example 10 is called EXAMPLE10 and is a range splitting controller. There is no simulated process with this example.

Control Loop

Figure 3-30 shows the application of Example 10 in a complete control loop.

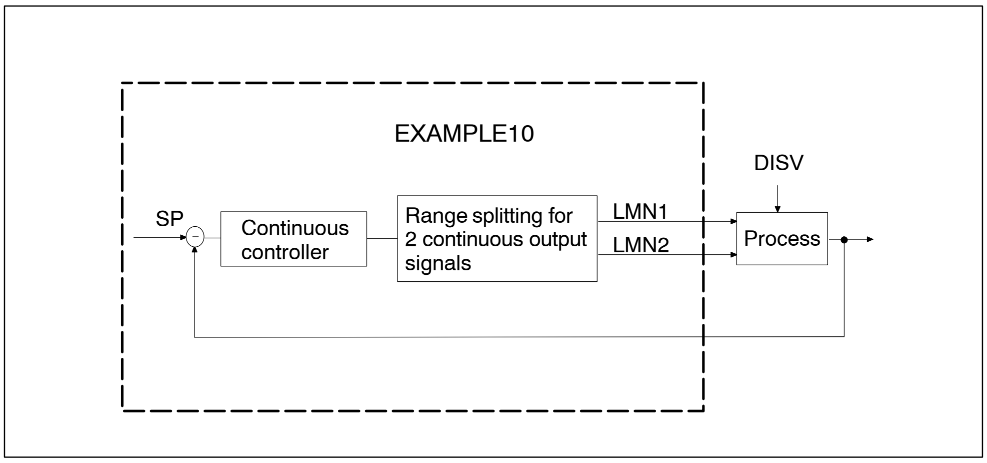


Figure 3-30 Control Loop with Example 10

Block Call

Figure 3-31 shows the block call of Example 10.

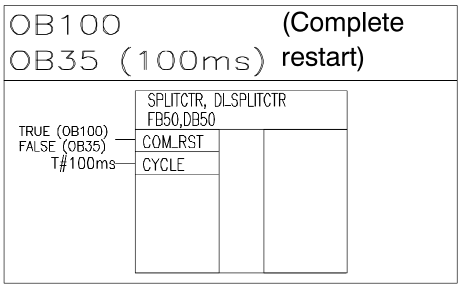


Figure 3-31 Block Call of Example 10

Application

The block SPLITCTR is a PID controller with range splitting for 2 continuous actuators. Figure 3-32 shows the block interconnections of SPLITCTR.

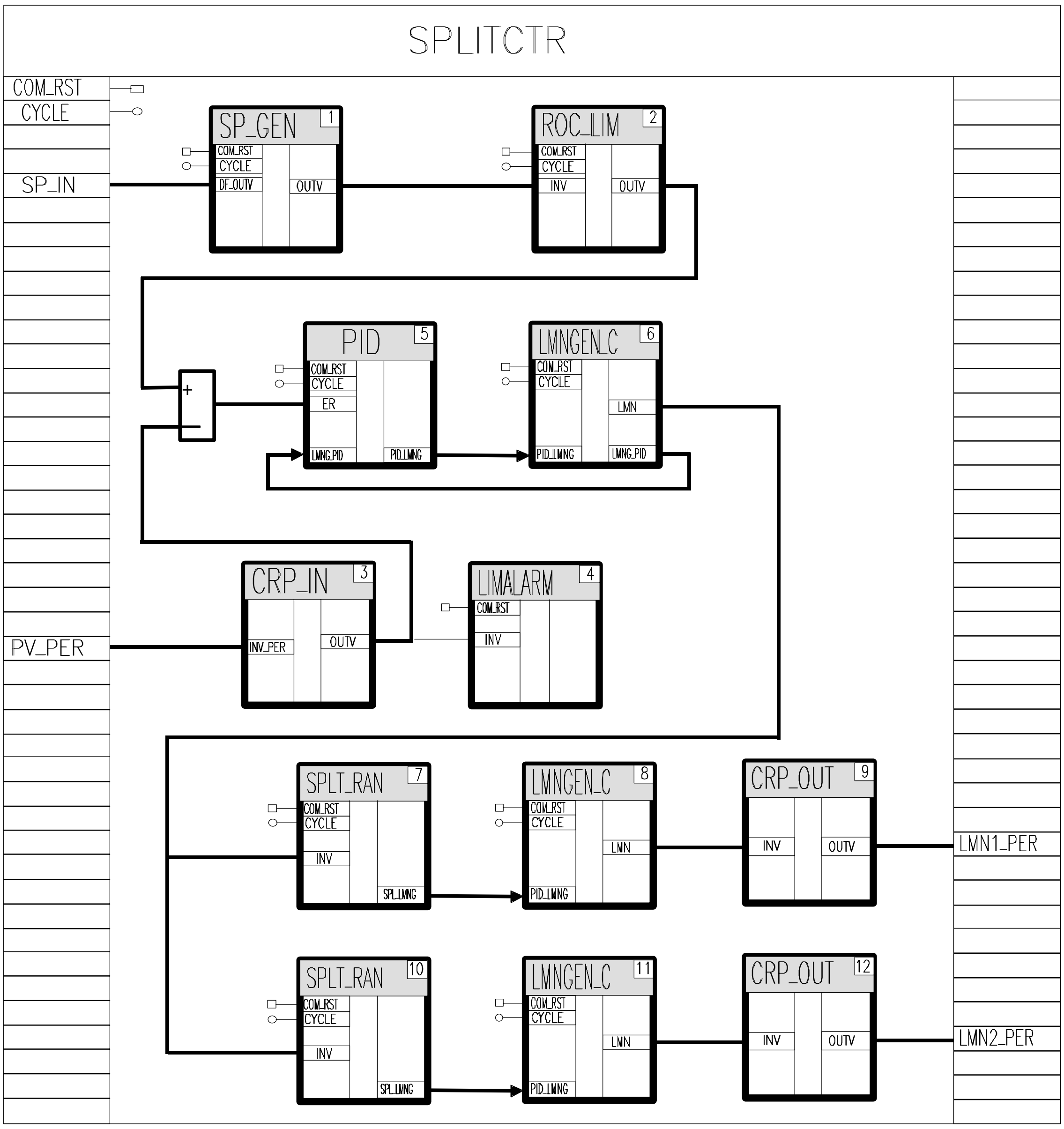


Figure 3-32 Block Interconnections of SPLITCTR

Functional Description

The setpoint generator SP_GEN sets the setpoint whose rate of change is limited by the limiter ROC_LIM. The peripheral process variable is converted to a floating-point value by CRP_IN and monitored by the limit value monitor LIMALARM to check that it does not exceed selected limit values. The error signal is routed to the PID algorithm. The manipulated value processing block LMNGEN_C generates the analog manipulated variable LMN. The manipulated value range is split into two ranges by two SPLT_RAN blocks. For each range, LMNGEN_C calculates an analog manipulated variable that is converted to the peripheral format by CRP_OUT.

Complete Restart

During a complete restart, each block is called individually. Blocks with a complete restart routine run through this routine.

3.12 Example 11: Override Controller (OVR_CTR)

Overview

Example 11 is called EXAMPLE11 and ist ein Override Controller. There is no simulated process with this example.

Control Loop

Figure 3-33 shows the application of Example 11 in a complete control loop.

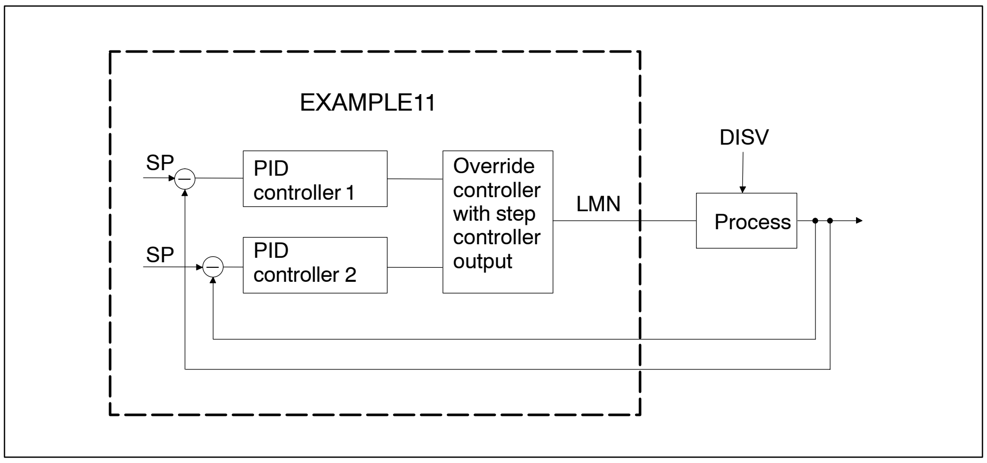


Figure 3-33 Control Loop with Example 11

Block Call

Figure 3-34 shows the block call of Example 11.

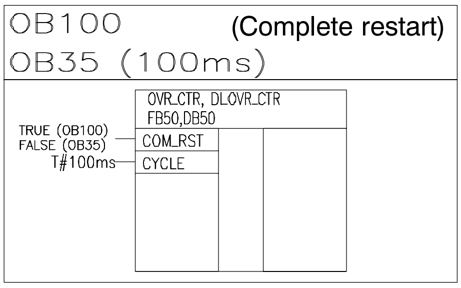


Figure 3-34 Block Call of Example 11

Application

The block OVR_CTR is an override controller. Two PID controllers are connected to one step controller output. Figure 3-35 shows the block interconnections of OVR_CTR.

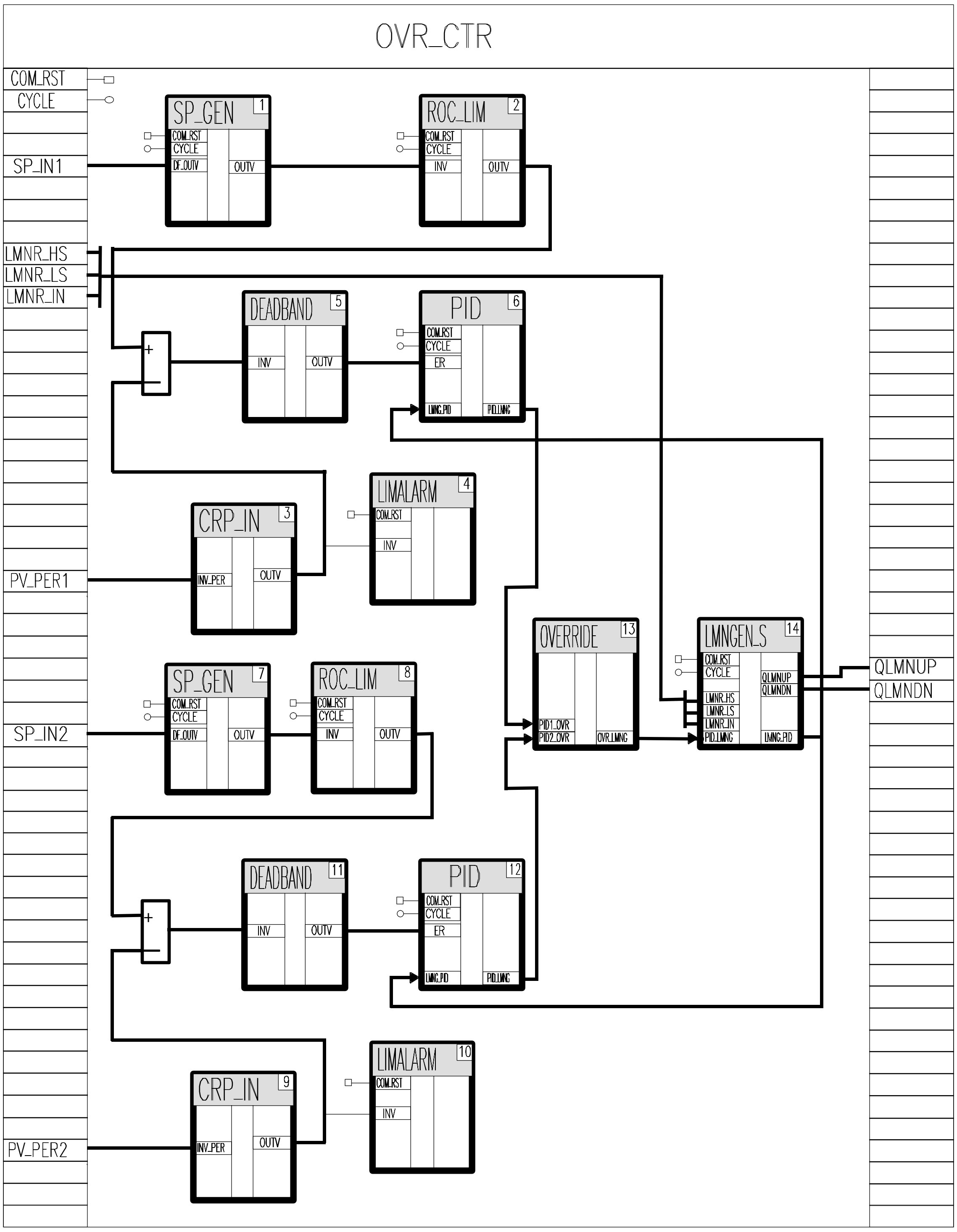


Figure 3-35 Block Interconnections of OVR_CTR

Functional Description

The setpoint generators SP_GEN sets the setpoints whose rates of change are limited by the limiter ROC_LIM. The peripheral process variables are converted to floating-point values by CRP_IN blocks and monitored by the limit value monitors LIMALARM to check that they do not exceed selected limit values. The error signals are led to the PID algorithm. The manipulated values of the two PID blocks are applied to an OVERRIDE block. Here, either the maximum or the minimum of the two manipulated values is determined and passed to the manipulated value processing block LMNGEN_S. In the override controller, LMNGEN_S can only operate in the “step controller with position feedback signal“ mode.

Complete Restart

During a complete restart, each block is called individually. Blocks with a complete restart routine run through this routine.

3.13 Example 12: Multiple Variable Controller

Overview

Example 12 is called EXAMPLE12 and is a multiple variable controller. There is no simulated process with this example.

Control Loop

Figure 3-36 shows the application of Example 12 in a complete control loop.

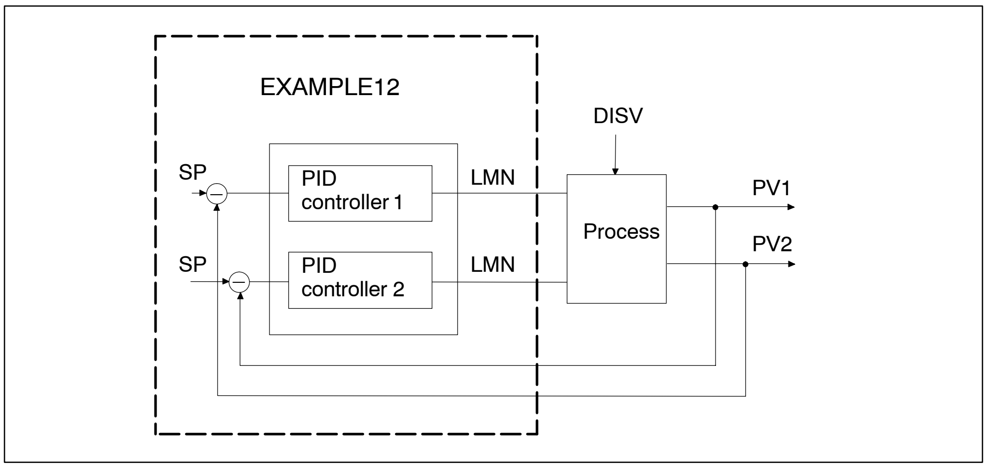


Figure 3-36 Control Loop with Example 12

Block Call

Figure 3-37 shows the block call of Example 12.

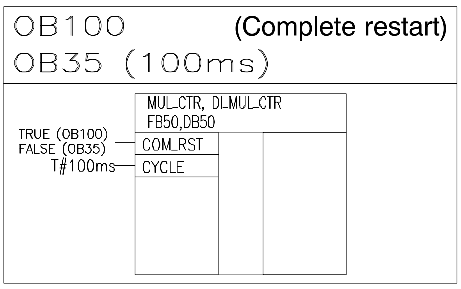


Figure 3-37 Block Call of Example 12

Application

The block MUL_CTR is a multiple variable controller. Two PID controllers with continuous outputs are connected to the process. Figure 3-38 shows the block interconnections of MUL_CTR.

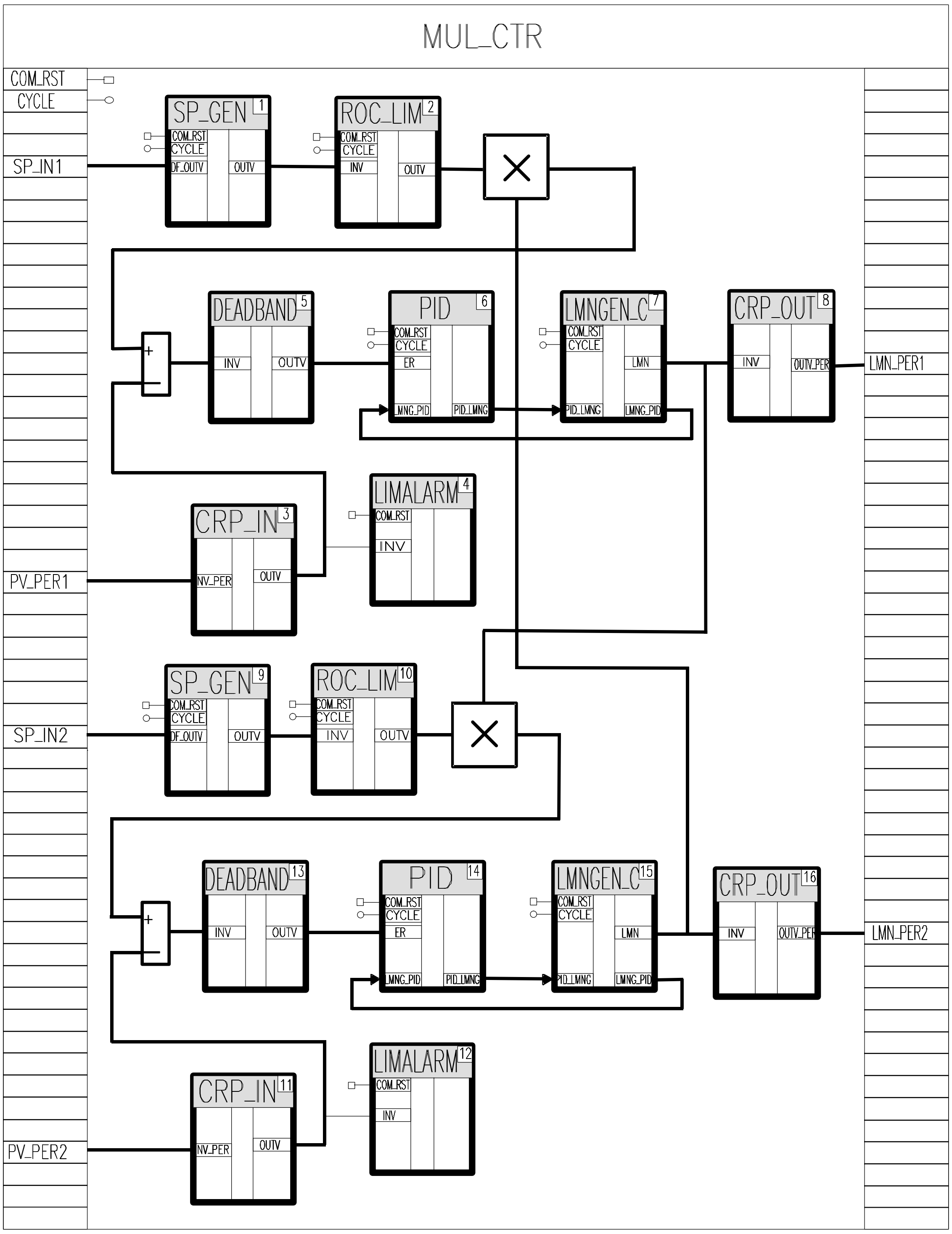


Figure 3-38 Block Interconnections of MUL_CTR

Functional Description

The functionality of each PID controller is analogous to the continuous controller PIDCTR_C in Section 3.3.1 on page 3-10.

In this case, the manipulated value of one controller is multiplied by the manipulated value of the other controller.

Complete Restart

During a complete restart, each block is called individually. Blocks with a complete restart routine run through this routine.

Technical Data 4

4.1 Run Times

Block Name		CPU 313 in µs	CPU 314 in µs	CPU 315 CPU 315-2DP in µs	CPU 412-1 CPU 413-1 CPU 413-2DP in µs	CPU 414-1 CPU 414-2DP in µs	CPU 416-1 CPU 416-2DP in µs
A_DEAD_B	FB1	170	160	130	31	30	9
CRP_IN	FB2	60	60	60	21	30	6
CRP_OUT	FB3	220	210	180	42	30	12
DEAD_T	FB4	330	320	260	60	39	17
DEADBAND	FB5	210	200	160	32	30	10
DIF	FB6	710	690	550	92	55	26
ERR_MON	FB7	350	340	270	49	34	14
INTEG	FB8	510	500	400	74	44	20
LAG1ST	FB9	670	650	520	94	56	27
LAG2ND	FB10	1140	1110	880	158	86	44
LIMALARM	FB11	610	590	470	65	41	18
LIMITER	FB12	170	170	140	30	30	9
LMNGEN_C	FB13	410	390	320	64	41	18
LMNGEN_S	FB14	1470	1430	1160	184	115	57
NONLIN	FB15	410	400	320	74	45	20
NORM	FB16	430	420	330	67	42	19
OVERRIDE	FB17	180	170	150	35	30	9
PARA_CTL	FB18	150	140	120	30	30	9
PID	FB19	1460	1420	1150	184	113	56
PULSEGEN	FB20	200	200	170	50	33	12
RMP_SOAK	FB21	200	200	160	40	30	11
ROC_LIM	FB22	680	660	530	90	55	24
SCALE	FB23	130	120	100	23	30	8
SPLT_RAN	FB24	110	100	90	23	30	7
SP_GEN	FB25	350	340	270	58	35	15
SWITCH	FB26	90	80	70	25	30	7
LP_SCHED	FC1	340	330	280	79	42	26

4.2 Work Memory Requirements

Block Name		FB length in memory (in bytes)	FB length when running (in bytes)	DB length in memory (in bytes)	DB length when running (in bytes)
A_DEAD_B	FB1	898	692	186	44
CRP_IN	FB2	182	70	122	20
CRP_OUT	FB3	206	96	114	14
DEAD_T	FB4	532	394	142	22
DB_DEADT (with 10 historical values)	DB3			138	40
DEADBAND	FB5	232	120	114	16
DIF	FB6	410	268	158	30
ERR_MON	FB7	558	360	206	52
INTEG	FB8	488	314	168	36
LAG1ST	FB9	534	368	156	30
LAG2ND	FB10	690	516	190	46
LIMALARM	FB11	390	240	152	28
LIMITER	FB12	262	140	124	20
LMNGEN_C	FB13	1576	1280	276	80
LMNGEN_S	FB14	2578	2152	360	110
NONLIN	FB15	826	672	138	18
DB_NONLI (with starting point and 4 curve points)	DB4			146	42
NORM	FB16	234	122	130	24
OVERRIDE	FB17	362	214	146	28
PARA_CTL	FB18	406	232	234	82
PID	FB19	1560	1242	340	98
PULSEGEN	FB20	1110	872	190	34
RMP_SOAK	FB21	1706	1500	212	62
DB_RMPSK (with starting point and 4 curve points)	DB2			146	42
ROC_LIM	FB22	1242	980	222	50
SCALE	FB23	136	32	114	16
SPLT_RAN	FB24	304	180	138	28
SP_GEN	FB25	658	484	164	40
SWITCH	FB26	238	116	118	18
LP_SCHED	FC1	1104	972	280	79
DB_LOOP (with 5 control loops)	DB1			190	64

4.3 Rules of Thumb

Run Time

You can calculate the total run time with the following formula:

```
  Run time of the called blocks   (from Modular PID Control)
+ number of block calls * constant
----------------------------------
= total run time
```

You can find out the number of block calls as follows:

```
  Blocks called   (from Modular PID Control)
+ user FBs called
----------------------------------
= number of block calls
```

The following **constants** apply to the CPUs:

CPU	Constant
CPU 313	105 µs
CPU 314	100 µs
CPU 315, CPU 315-2 DP	80 µs
CPU 412-1, CPU 413-1	23 µs
CPU 414-1, CPU 414-2 DP	12 µs
CPU 416-1, CPU 416-2 DP	9 µs

Memory Requirements

The rule of thumb for memory requirements relates to the work memory.

```
  Memory required by the FBs used   (from Modular PID Control)
+ memory required by instance DB data of called FBs   (from Modular PID Control)
+ number of block calls * 120 bytes
--------------------------------------------------------------------------
= total memory required
```

You can find out the number of block calls as follows:

```
  Called blocks  (from Modular PID Control)
+ called user FBs
---------------------------------------------
= number of block calls
```

Configuration Tool for Modular PID Control 5

Requirements

STEP 7 must be installed correctly on your programming device/PC.

Diskettes

The software is supplied on CD.

Installation

To install the software, follow the steps below:

1. Insert the CD in the CD drive of your programming device/PC.
2. Start the dialog for installing software under WINDOWS by double-clicking the “Add/Remove Programs” icon in the “Control Panel”.
3. Select the CD drive and the file Setup.exe and start installation.
   The configuration tool is then installed on your system.
4. Follow the instructions displayed by the installation tool step by step.

Readme File

Important up–to–date information about the supplied software is stored on the readme file. You will find this file in the Windows Start menu via START > SIMATIC > Product Information > English.

Purpos*e*

The configuration tool is intended to make installation, startup and testing of your controller easier leaving you free to concentrate on the real control problem in hand.

Functions of the configuration tool

Each function has its own window. You can also call a function more than once, in other words, you can, for example, display the loop monitors of more than one controller at the same time.

Monitoring Controllers

Using the curve recorder function, you record the values of selected variables in the control loop over time and display them. Up to four variables can be displayed at the same time.

With the loop monitor function, you can display the relevant control variables (setpoint, manipulated variable, and process variable) of a selected controller.

Process Identification

Using the process identification function, you can find out the optimum controller setting for a particular control loop. The characteristics of the process are identified experimentally. Based on these characteristics, the ideal controller parameters are calculated and saved for future use.

Manual Control

The loop monitor function allows you to modify or set new values for relevant variables of the control loop.

Integrated Help

The configuration tool has an integrated help system that supports you while working with the tool. You can call the help system as follows:

- With the menu command **Help ▶ Contents**
- By pressing the F1 key
- By clicking the Help button in the individual dialogs.

References A

Further Reading

The following books deal with the basics of control engineering:

/350/ User manual: *SIMATIC 7,*
Standard Control

/352/ J. Gißler, M. Schmid: Vom Prozeß zur Regelung. Analyse, Entwurf, Realisierung in der Praxis. Siemens AG. ISBN 3-8009-1551-0.

Index