SIEMENS

SIMATIC

Standard PID Control

Manual

This manual is part of the documentation package with the order number
6ES7830-2AA21-8BG0

Edition 03/2003

A5E00204510-02

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

Disclaim of Liability

We have checked the contents of this manual for agreement with the hardware and software described. Since deviations cannot be precluded entirely, we cannot guarantee full agreement. However, the data in this manual are reviewed regularly and any necessary corrections included in subsequent editions. Suggestions for improvement are welcomed.

Technical data subject to change.

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 configuration tool and explains how you use it.

Required Basic Knowledge

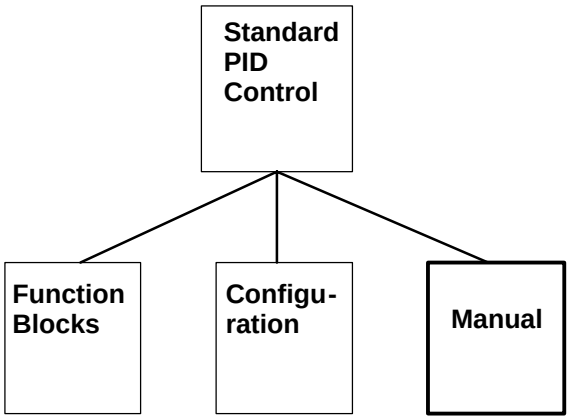
To understand this manual, you should be familiar with automation and process control engineering.

In addition, you should know how to use computers or devices with similar functions (e.g programming devices) under Windows 95/98/2000 or NT operating systems. Since Standard 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 Standard PID Control V5.1 and Standard PID Control Tool V5.1.

Place of this Documentation in the Information Environment



The “Standard PID Control” software product includes three separate products:

- The product ”Standard PID Controller FB” consists essentially of the two controller blocks PID_CP and PID_ES.
- The product “Standard PID Control Tool” consists essentially of the tools for configuring the controller blocks.
  This product is referred to simply as “configuration tool” in this manual.
- This manual is a separate product and describes both the product ”Standard PID Control FB” and the configuration tool ”Standard PID Control Tool”.

The “Standard PID Control” Software Package

The “Standard PID Control” software package provides a comprehensive concept for implementing control functions in the SIMATIC S7 programmable logic controllers. The controller is completely programmed with its full range of functions and features for signal processing. To adapt a controller to your process, you simply select the subfunctions you require from the complete range of functions. The time and effort required for configuration is therefore reduced to omitting functions you do not require. In all these tasks, you are supported by the configuration tool.

Since configuration is restricted to selecting or, in some cases, extending basic functions, the concept of the Standard PID Control is easy to learn. Even users with only limited knowledge of control systems will create high-quality controls.

Finding Your Way

- Chapter 1 provides you with an overview of the Standard PID Control.
- Chapter 2 explains the structure and the functions of the Standard PID Control.
- Chapters 3 helps you to design and start up a Standard PID Control.
- Chapters 4 explains the signal processing in the setpoint/process-variable channel and in the controller.
- Chapters 5 explains the signal processing in the continuous controller output.
- Chapters 6 explains the signal processing in the step controller output.
- Chapters 7 shows you how to work with the loop scheduler and introduces examples of controller structures.
- Chapters 8 contains technical data and block diagrams.
- Chapters 9 contains parameter lists for the Standard PID Control.
- Chapters 10 provides you with an overview of the configuration tool.
- Appendices A contains the literature list.
- Important terms are explained in the glossary.
- The index helps you to access areas containing keywords easily and fast.

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* x.y). References to other manuals and documentation are indicated by numbers in slashes /.../. These numbers refer to the titles of manuals listed in the Appendix.
- Instructions for you to follow are marked by a black dot.
- Sequences of activities are numbered or explained as explicit steps.
- Alternative courses of action or decisions you need to take are indicated by a dash.

Further Information

This manual is intended as a reference work that provides you with the information you will require to work with the standard controller. You do, however, require a broader scope of information that is available in the following manuals: **/70/**, **/71/**, **/100/**, **/101/**, **/231/**, **/232/**, **/234/**, **/352/**.

Further Support

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

`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 (0) 180 5050-222 Fax: +49 (0) 180 5050-223 E-Mail: adsupport@ siemens.com GMT: +1:00		
Europe / Africa (Nuernberg) **Authorization** Local time: Mon.-Fri. 8:00 to 17:00 Phone: +49 (0) 180 5050–222 Fax: +49 (0) 180 5050-223 E-Mail: adsupport@ siemens.com GMT: +1:00	**United States** (Johnson City) **Technical Support and Authorization** Local time: Mon.-Fri. 8:00 to 17:00 Phone: +1 (0) 423 262 2522 Fax: +1 (0) 423 262 2289 E-Mail: simatic.hotline@ sea.siemens.com GMT: –5:00	**Asia / Australia** (Beijing) **Technical Support and Authorization** Local time: Mon.-Fri. 8:00 to 17:00 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 via our representatives database.
- Information on field service, repairs, spare parts and more under “Services”.

Contents

Product Overview Standard PID Control 1

1.1 The Product ”Standard PID Control”

Concept of ”Standard PID Control”

The software product ”Standard PID Control” essentially consists of two **function blocks** (FBs) which contain the algorithms for generating control and signal-processing functions for continuous or step controllers. It is a pure software control in which a standard function block incorporates the functionality of the controller.

The behavior of the controller itself and the properties of the functions in the measuring and adjusting channel are realized or simulated by means of the numberic algorithms of the function block. The data required for these cyclic calculations are saved in control-loop-specific data blocks. An FB is only required once to create several controllers.

Every controller is represented by an **instance DB** which must be created application-specifically. When the ”Standard PID Control Tool” is used, this DB is created ’implicitly. This means that the design of a specific controller is limited to specifying the structural and value parameters in the editing windows of the user interface. The instance DB is created by the configuration tool.

The calculation of the algorithms for a certain controller is carried out in the processor of the S7 automation system (AS) in the set time intervals (sampling times). The calculation results and thus the updated values of the input and output variables (measuring and manipulated variables) and status signals (limits) are stored in the corresponding instance DB or transferred to the process periphery.

In order to process several control loops which are to be executed at different intervals – but equidistantly – depending on the inertia of the respective process, a **controller call distribution function** (Loop Scheduler = LP_SCHED) is available through which the configuration of extensive plant controls becomes structured and thus simple. In addition, even utilization of the CPU is ensured.

Overview of the Basic Functions

In many controlling tasks not only the classic PID controller as a process-influencing element is of importance, but high requirements are also placed on the signal processing function.

A controller formed by means of the ”Standard PID Control” software package thus consists of a number of sub-functions which you can configure separately. In addition to the actual controller with the PID algorithm functions for conditioning the setpoint and process variables as well as for revision the calculated manipulated variable are also integrated.

Display and monitoring functions are also included (not displayed in the overview scheme).

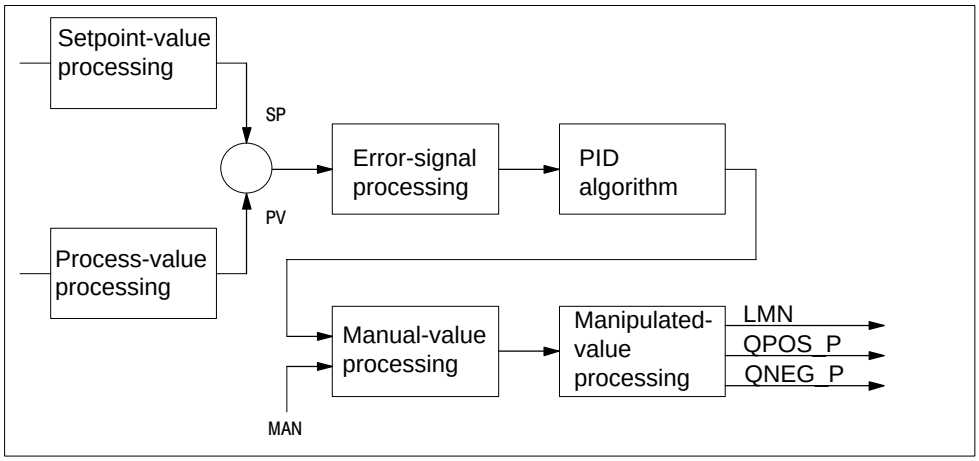


Figure 1-1 Function Overview of the Software Block ”Continuous Controller”

Creating the Control

The software package ”Standard PID Control” can be used to configure a controller for a specific control task. Its function set can be planned to be limited. So-called tuning switches can be used to activate or de-activate sub-functions or to set complete branches inactive. Only the function parts remaining in the reduced structure then have to be configured.

The creation of a closed-loop control from its structuring through the parameter configuration to its call at the correct time by the system program is possible to a great extent without programming. STEP 7 knowledge is required.

For information on structuring the instance DB please refer to *Chapter 9* of this manual. One datum, i.e. one line is reserved for each structure or value parameter. The structure as well as the desired properties of the control can be specified by editing the corresponding entries.

However this procedure is not advisable since it does not allow clear structuring. The configuration tool specially conceived for Standard PID Control simplifies this task considerably.

Note

The configuration tool cannot be used to configure the LP-SCHED block. Its functionality is defined exclusively by means of entries in the respective data block.

1.2 The ”Standard PID Control Software Product ”

Product Structure: ”Standard PID Control”

After the “Standard PID Control” product has been installed, your programming device/personal computer contains a STEP 7 block library called “Standard PID Control”. This contains two standard function blocks, a standard function, templates for data blocks as well as the STEP 7 project “zEn28_03_StdCon” with 6 examples and the text on getting started.

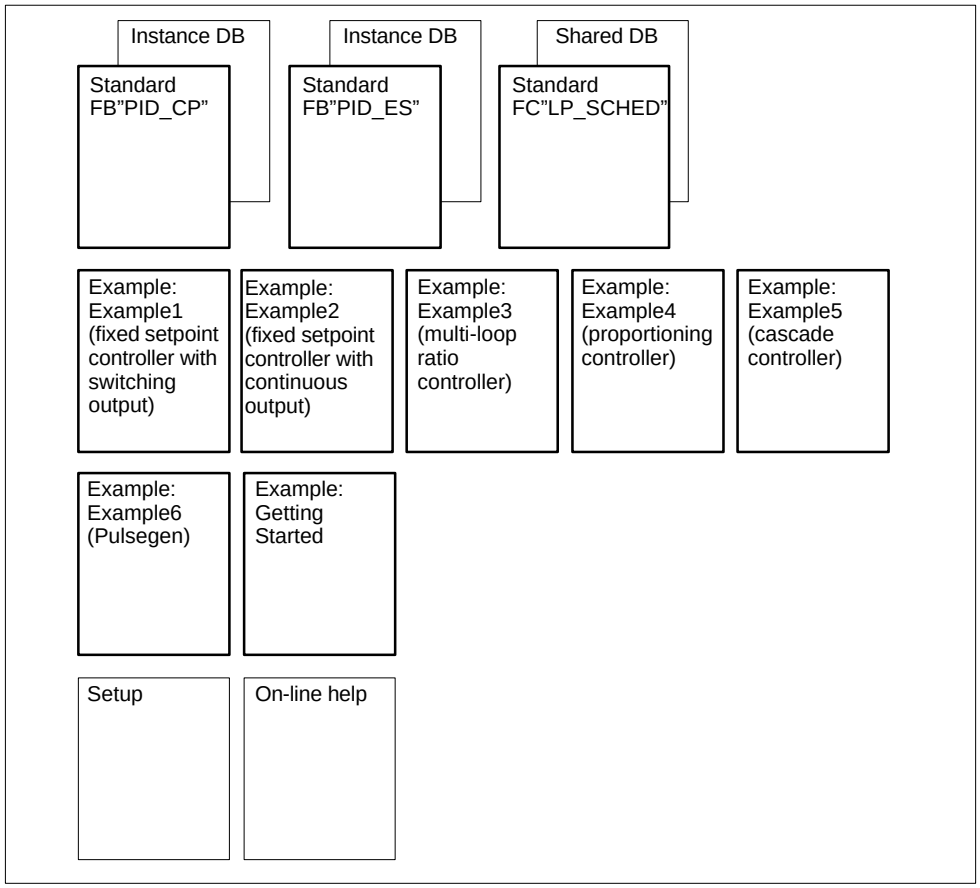


Figure 1-2 Contents of the ”Standard PID Control” Software Package

- The Standard FB **PID_CP** contains all the control-specific functions of a continuous PID controller including a pulse output for proportional final controlling elements.
- The standard FB **PID_ES** contains all the control-specific functions of a PID controller with three-step output.
- Th standard FC **LP_SCHED** controls the call distribution of the individual controllers within a watchdog-interrupt OB for applications with many control loops. The block also takes over the initialization of the controller structure when starting up the CPU or the automation system.

In addition the software package contains a **setup program** for installing the ”Standard PID Control” on programming devices/personal computers as well as the **on-line help** which makes information on the sub-functions and individual parameters available during your practical work.

Predefined Application Structures

The scope of delivery of the ”Standard PID Control” is supplemented by data structures (instance DBs) for the controller types used most often or for the most important multi-loop controls.

You can use these ready-to-use structural examples (Example1 to Example6) if creating a controllers from its very beginning is too troublesome or if you want to avoid errors while creating coupled controller structures.

The following example structures are available:

Designation	Provided functionality	Comment
Example1	Fixed setpoint controller with switching output – integrating final controlling elements (for example motor drives)	”PID step controller” with three-step response
Example2	Fixed setpoint controller with continuous output – for proportional final controlling elements	”Analog PID controller”
Example3	Multi-loop ratio control	The ratio of two process variables is kept constant
Example4	Blending control	The components to be blended are kept to a constant percentage and the total quantity controlled
Example5	Cascade control	Improvement of the control behavior by including process variables in lower-level control loops
Example6	Continuous controller with pulse outputs and system simulation	

”Configuration of Standard PID Control”

The functions of the software package ”Configuration of the Standard PID Control” are described in *Chapter 10* of this manual.

1.3 The Application Environment and the Field of Application

Hardware Environment

The controllers created with the ”Standard PID Control” software package can be executed on the:

- S7-300- und S7-400 (CPU with floating point and watchdog interrupt)
- C7-CPUs
- Win AC

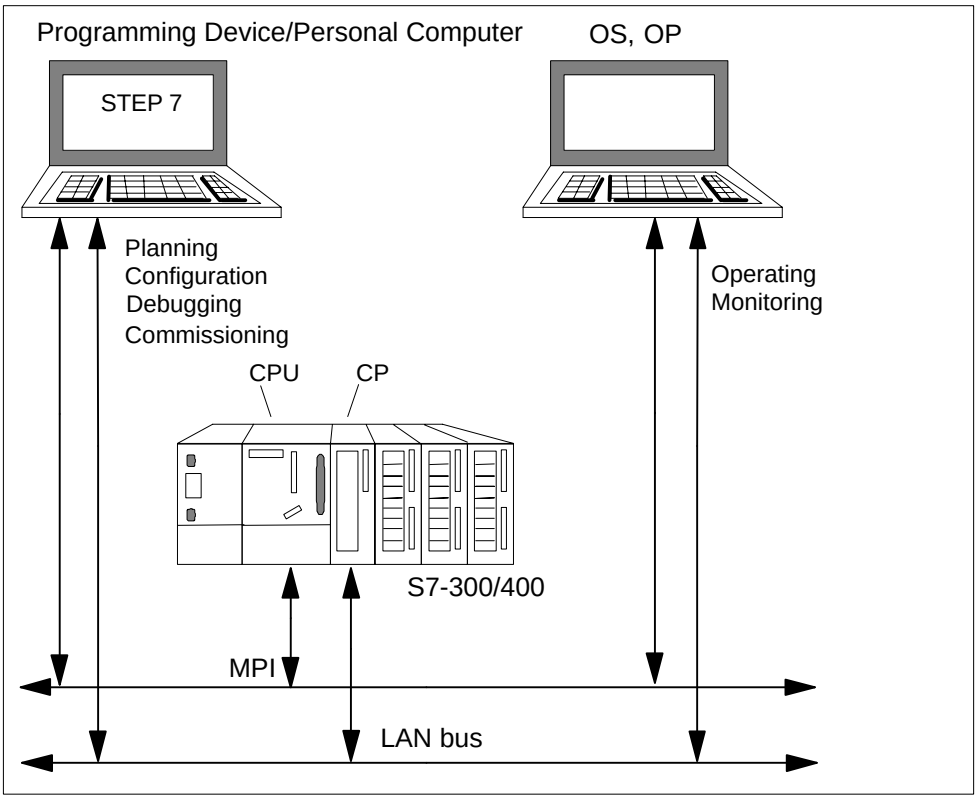


Figure 1-3 Application Environment of the ”Standard PID Control” Software Package

Software Environment

The ”Standard PID Control” software package is conceived for use in the STEP 7 program group.

The creation software for standard controls can be installed locally on a programming device/personal computer or in a network on a central network drive.

The System Frame

Since digital realization of controller functions always require a high degree of computational operations (word processing), it is important to have an idea of the load on the CPU available. The following guidelines can be used:

- Extent of code of a function block

  (PID_CP or PID_ES): $\leq$ 8 KBytes
- Data per controller $\leq$ 0.5 KBytes
- Basic data for minimum run times (processing times) of a PID controller on different automation systems are included in *Section 8.1* (Technical Data).
- The size of the required area in the user memory and thus the number of control loops which can thus be installed theoretically on the basis of the amount of memory available (at 50 % utilization of the work memory by the control tasks) is included in the Technical Data (*refer to Section 8.1*).
- There are no memory requirements for an L stack.
- Interrupts are not delayed by the processing of the control FB.

Controller Call Distribution

If many controllers or controllers with high sampling times have to be called, the extent of the priority class model is not sufficient with regard to the watchdog interrupt OBs. The controller call distribution function LP_SCHED (Loop Scheduler) allows several controllers with different sampling times to be called equidistantly in a watchdog interrupt OB.

The tasks of the call distribution are:

- Controlling the calls of the individual controllers within a (watchdog interrupt) priority class.
- Calling the installed standard controllers when the CPU is first started up.

Possible Applications and Limitations of the Standard PID Control

The control function implemented by processing an FB can basically be used for any application. The control performance and the speed in which actual control loops are processed only depends on the performance of the CPU being used.

With any given CPU, a compromise must be made between the number of controllers and the frequency at which the individual controllers have to be processed. The faster the control loops have to be processed, in other words the more often the manipulated variables must be calculated per unit of time, the less the number of controllers that can be installed.

The standard function blocks PID_CP and PID_ES allow you to generate and operate software controllers based on the conventional PID algorithm of the Standard PID Control. Special functions in terms of handling process signals on the controller are not included.

There are no restrictions to the type of process that can be controlled. Both slow processes (temperatures, tank levels) and very fast processes (flow rate, motor speed) can be controlled.

Forms of Applications of the Standard PID Control:

- Fixed setpoint control with P, PI, PD, PID controller
- Fixed setpoint control with continuous P, PI, PD, PID controller
- Fixed setpoint control with feedforward control
- Cascade control (step controller only in secondary loop)
- Ratio control (two loops)
- Blending control

Range of Functions of the Standard PID Control

By configuring the functions contained in the “Standard PID Control” product, you can create controllers with the following characteristics and modes:

- Adjustment of the setpoint by a ramp soak
- Limitation of the rate of change of the reference input and (with controllers with a continuous output) of the manipulated variable
- Limitation of the absolute values of the reference input and (with controllers with a continuous output) of the manipulated variable
- Suppression of noise in the process variable or setpoint branch by filtering the error signal
- Suppression of high frequency oscillations in the process variable branch by delaying the process variable signal
- Linearization of quadratic functions of the process variable (flow control with differential pressure sensors)
- Possibility of calling your ”own functions” in the setpoint, process variable and/or manipulated variable branch
- Manual mode (controlling the manipulated variable from a programming device or OP/OS)
- Monitoring two upper and two lower limits for the process variable and/or error signal
- Monitoring of the rate of change of the process variable
- The option of including a P and D action in the feedback path of the controller

Designing Digital Controllers

2

2.1 Process Characteristics and Control

Process Characteristics and the Controller

The static behavior (gain) and the dynamic characteristics (time lag, dead time, reset times etc.) of the process to be controlled have a significant influence on the type and time response of the signal processing in the controller responsible for keeping the process stable or changing the process according to a selected time schedule.

The process has a special significance among the components of the control loop. Its characteristics are fixed either by physical laws or by the machinery being used and can hardly be influenced. A good control result is therefore only possible by selecting the controller type best suited to the particular process and by adapting the controller to the time response of the process.

Precise knowledge of the type and characteristic data of the process to be controlled is indispensable for structuring and designing the controller and for selecting the dimensions of its static (P mode) and dynamic (I and D modes) parameters.

Process Analysis

To design the controller, you require exact data from the process that you obtain by means of a transfer function following a step change in the setpoint. The (graphical) analysis of this (time) function allows you to draw conclusions about the selection of the most suitable controller function and the dimensions of the controller parameters to be set.

The configuration tool supports you to a large extent during the phase of process analysis.

Before describing the use of the Configuration Standard PID Control tool the next sections briefly look at the most common processes involved in automation. You may possibly require this information to help you to decide the best procedure for the analysis and simulation of the process characteristics.

Type and Characteristics of the Process

The following processes will be analyzed in greater detail:

- Self-regulating process
- Self-regulating process with dead time
- Process with integral action

Self-regulating Process

Most processes are self-regulating, in other words, after a step change in the manipulated variable, the process (controlled) variable approaches a new steady-state value. The time response of the system can therefore be determined by plotting the curve of the process variable with respect to time PV(t) after a step change in the manipulated variable LMN by a value greater than 1.5% of its total range.

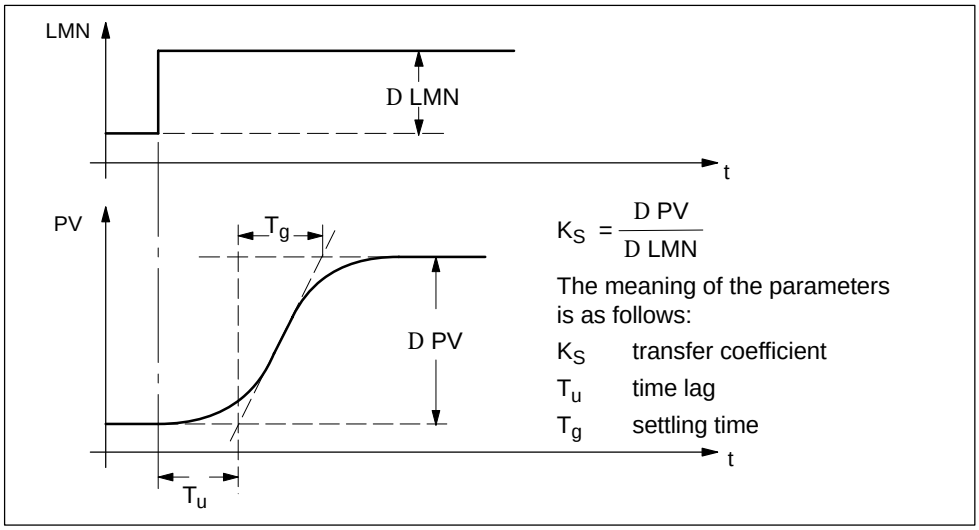


Figure 2-1 Step Response of a Self-Regulating Process (first order)

If the process response within the manipulated variable range is linear, the transfer coefficient K_S indicates the gain of the control loop. From the ratio of the time lag to the settling time T_u/T_g, the controllability of the process can be estimated. The smaller this value is, in other words the smaller the time lag relative to the settling time, the better the process can be controlled.

According to the values $_u$ and T_g, the time response of a process can be roughly classified as follows:

$T_u < 0.2$ min and $T_g < 2$ min $\rightarrow$ fast process

$T_u > 0.5$ min and $T_g > 5$ min $\rightarrow$ slow process

The absolute value of the settling time therefore has a direct influence on the sampling time of the controller: The higher T_g is, in other words the slower the process reaction, the higher the sampling time that can be selected.

Self-Regulating Process with Dead Time

Many processes involving transportation of materials or energy (pipes, conveyor belts etc.) have a time response similar to that shown in Figure 2-2. This includes a start-up time T_a made up of the actual dead time and the time lag of the self-regulating process. In terms of controllability of the process it is extremely important that T_t remains small relative to T_g or in other words that the relationship $T_t/T_g \leq 1$ is maintained.

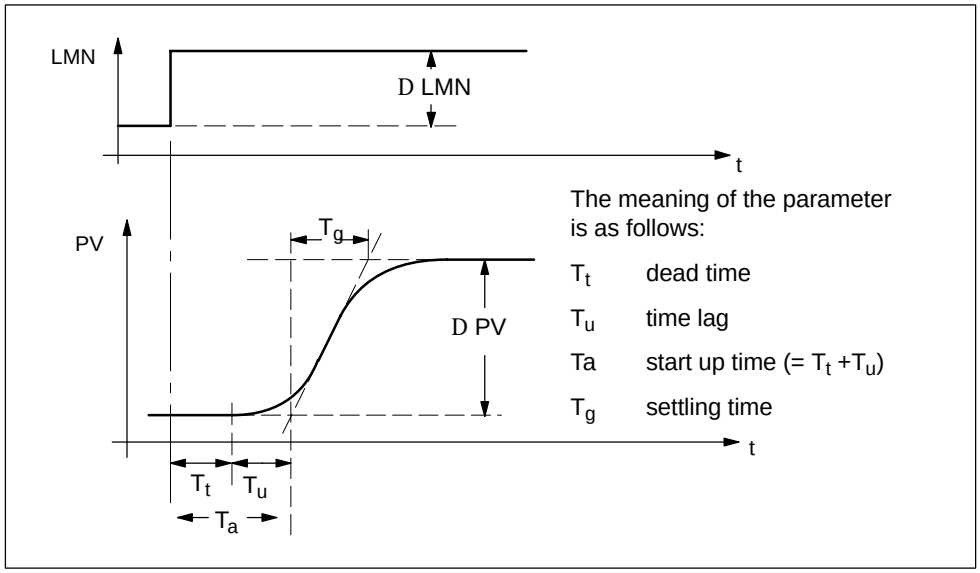


Figure 2-2 Step Response of a Self-Regulating Process with Dead Time (T_t-PT Process)

Since the controller does not receive any signal change from the transmitter during the dead time, its interventions are obviously delayed and the control quality is therefore reduced. When using a standard controller, such effects can be partly eliminated by choosing a new location for the measuring sensor.

Process with Integral Action

Here, the slope of the ramp of the process variable (PV) after changing the manipulated variable by a fixed amount is inversely proportional to the value of the integration time constant (reset time) TI.

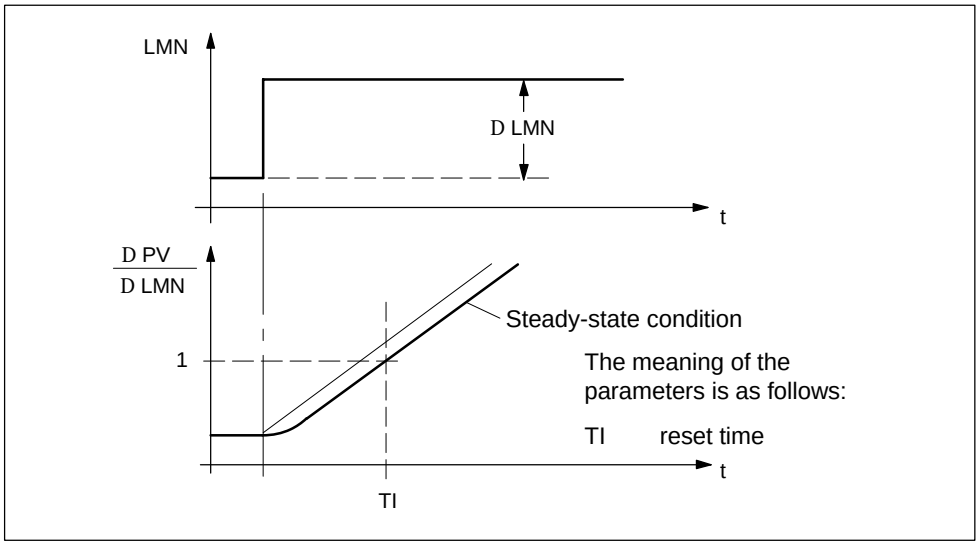


Figure 2-3 Step Response of a Non Self-Regulating Process (I Process)

Processes with an I component are, for example liquid level processes in which the level can be raised or lowered at different rates depending on the opening of the final control element. Important processes involving the I action are also the commonly used motor drives with which the rate of change of a traversing movement is directly proportional to the speed of the drive.

If no disturbance variables occur before the I element of a process with integral action (which is usually the case), a controller without I action should be used. The effects of a disturbance variable at the process input can usually be eliminated by feedforward control without using an I action in the controller.

2.2 Identifying Process Characteristics

Process Identification

As already mentioned, the investigation and identification of a given process response requires two steps:

1. The recording of the transfer function of the process after a step change in the manipulated variable.
2. The evaluation of the recorded or saved transfer function to determine a suitable controller structure and the optimum controller parameters.

1. Recording the Transfer Function

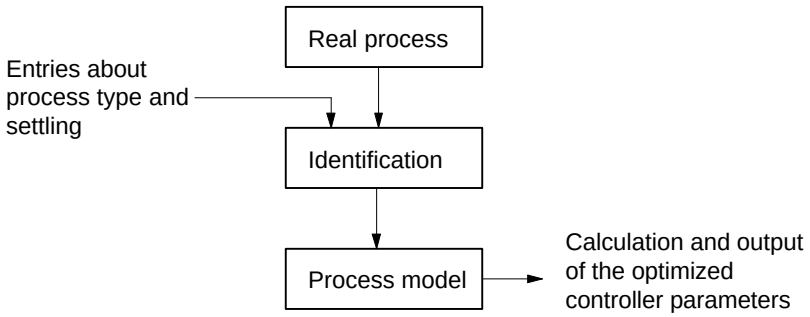
When Step 1 is executed, you are supported to a great extent by sub-function for process identification available in the configuration tool.

Comments in the dialog boxes provide you with background information about the current actions. Input boxes or output boxes are opened automatically at certain steps in the procedure.

2. Determining the Controller Data

For the actual process identification (Step 2) all you need to do is specify the tuning mode (a periodic or with 10% overshoot) and then start the automatic process identification by the system.

The following diagram illustrates the method used by the system for process identification:



The results of the process identification are displayed in a window. You can either save the PI or PID parameters in the database or discard the results and repeat the identification using different process data or different settings.

Process Identification and Type of Loop

A process identification can be done in the following modes as shown for the various types of processes:

	Data Acquired	Loop	Process	Process Stimulation
1.	On-line	disconnect. (manual mode)	without I component	Step change in manipulated variable: 
2.	On-line	connected (automatic mode)	without I component	Step change in the setpoint: 
3.	On-line	disconnect. (manual mode)	with I component	Pulse-shaped change in the manipulated variable: 
4.	On-line	connected (automatic mode)	with I component	Pulse-shaped setpoint change: 
5.	Off-line	Loop data from archive		

2.3 Feedforward Control

Feedforward Control

Disturbance variables affecting the process must be compensated by the controller. Constant disturbance variables are compensated by controllers with an I action. The control quality is not affected.

Dynamic disturbance variables, on the other hand, have a much greater influence on the quality of the control. Depending on the point at which the disturbance affects the control loop and the time constants of sections of the loop after the disturbance, error signals of differing size and duration occur that can only be eliminated by the I action in the controller.

This effect can be avoided in situations where the disturbance variable ”measuring” can be measured. By feeding the measured disturbance variable forward to the output of the controller, the disturbance variable can be compensated and the controller reacts much faster to the disturbance variable.

The standard controller has a signal input DISV for the disturbance variable. This disturbance variable can be switched to the summation point at the output of the PID controller by means of a structure switch (Figure 2-4).

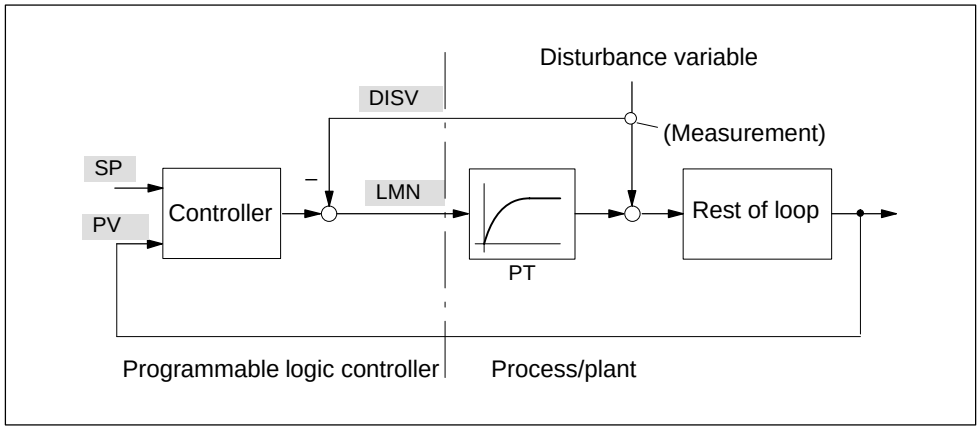


Figure 2-4 Compensating a Disturbance Affecting Process Input (Signal Names of the Standard PID Control)

2.4 Multi-Loop Controls

Processes with Inter-dependent Process Variables

The Standard Controller product contains prepared examples (Example3 to Example5, see *Chapter 7*) with which you can implement multi-loop controls quickly and easily. Using such control structures always has advantages when dealing with processes that have interdependent process variables.

The next sections describe the design of these controller structures and how they can be used.

Multi-loop Ratio Controls (Example3)

Whenever the relationship between two or more process variables in a process is more important than keeping its absolute values constant, **ratio control** is necessary (Figure 2-5).

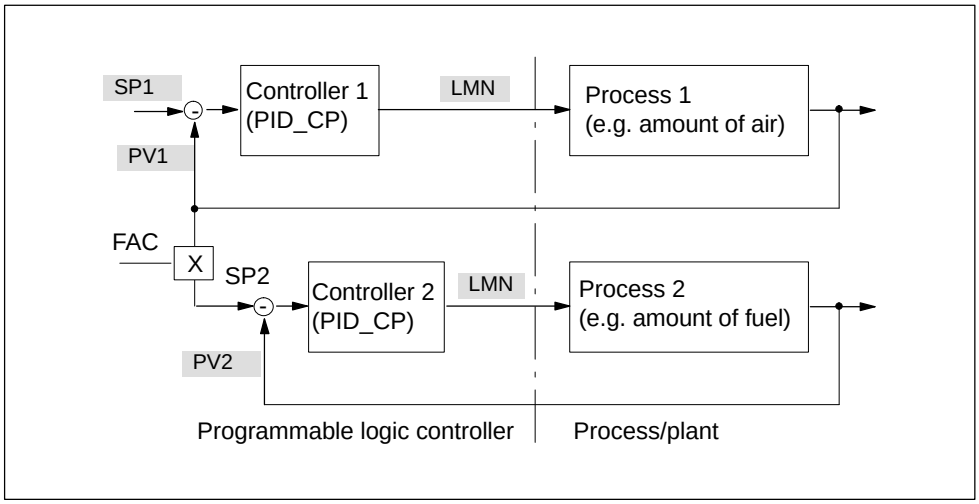


Figure 2-5 Ratio Control With Two Loops (Example3)

Generally the process variables that must be maintained in a preset ratio involve flow rates or volumes as found in combustion processes. In Figure 2-5, the amount of fuel in control loop 2 is controlled in a ratio selected with **FAC** to the amount of air set at **SP1**.

Blending Control (Example4)

In a blending process, both the total amount of materials to be mixed and the ratio of the components making up the total product must be kept constant.

Based on the principle of ratio control, these requirements result in a control structure in which the amount of each component of the mixture must be controlled. The setpoints of the components are influenced by the fixed proportion or ratio factors (FAC) and by the manipulated variable of the controller responsible for the total amount (Figure 2-6).

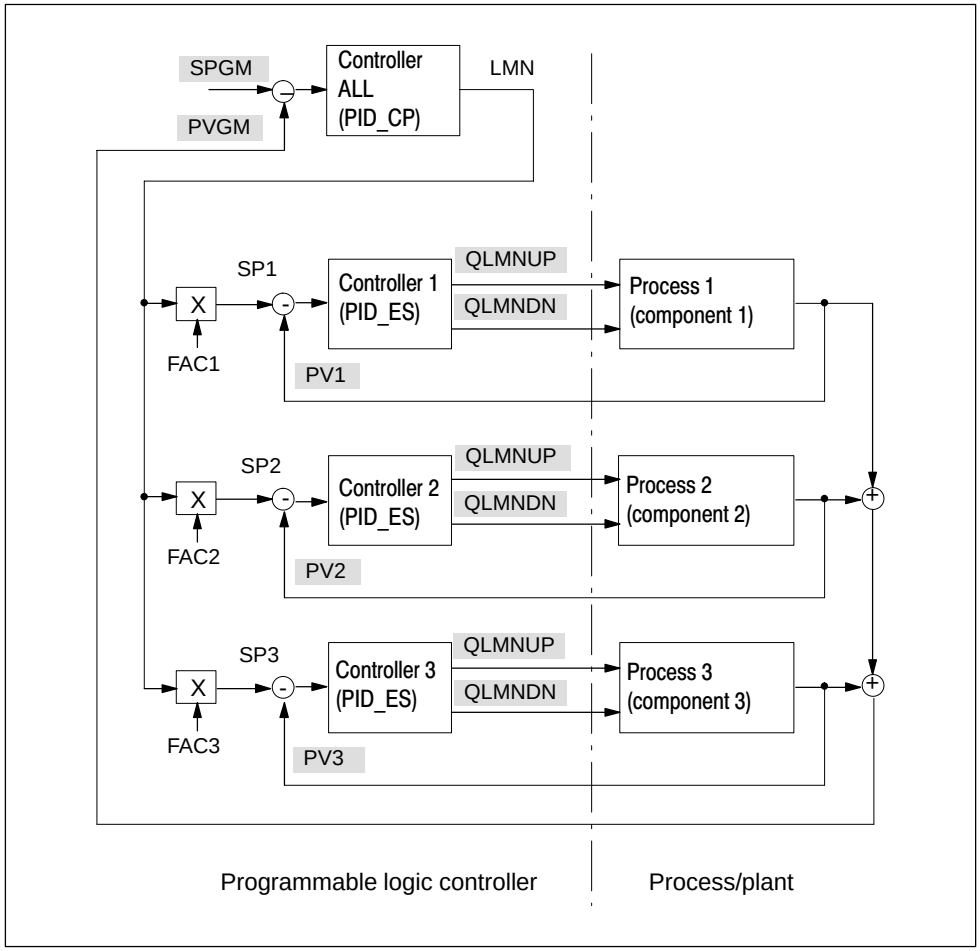


Figure 2-6 Blending control for three components (Example4)

The controller structure for the blending control (Example4) contains a controller with a continuous output (PID_CP) for controlling the total amount ALL and three step controllers (PID_ES) for the secondary control loops of the individual components 1 to 3, that make up the total amount according to the factors FAC1 to FAC3 (addition).

Cascade Control (Example5)

If a process includes not only the actual process variable to be controlled but also a secondary process variable that can be controlled separately, it is usually possible to obtain better control results than with a single loop control.

The secondary process variable PV2 is controlled in a secondary control loop (Figure 2-7). This means that disturbances from this part of the system are compensated before they can affect the quality of the primary process variable PV1. Due to the structure, inner disturbance variables are compensated more quickly since they do not occur in the entire control loop. The setting of the primary controller can then be made more sensitive allowing faster and more precise control with the fixed setpoint SP.

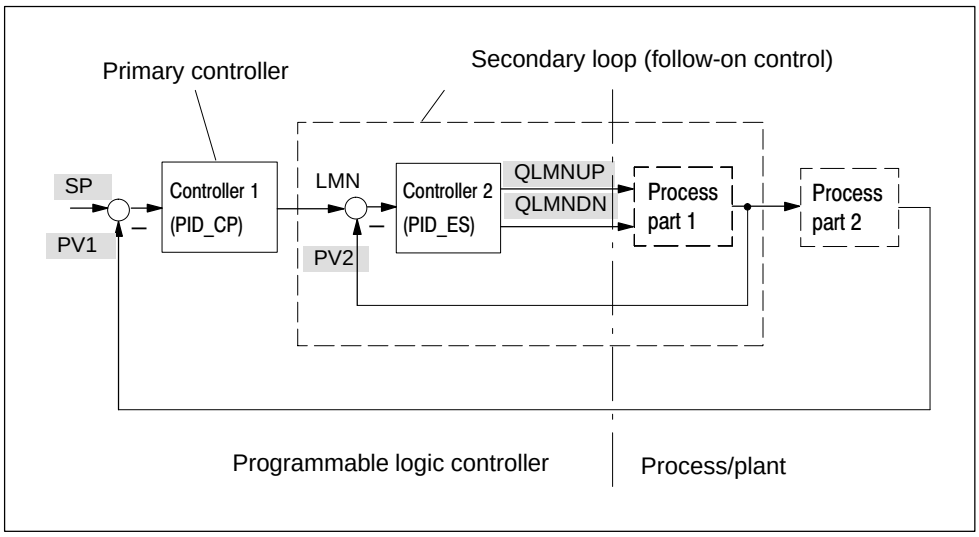


Figure 2-7 Two-Loop Cascade Control System (Example5)

The controller structure for cascade control (Example5) contains a controller with a continuous output (PID_CP) for controlling the reference input (setpoint) of the secondary loop and a step controller (PID_ES) to control the secondary process variable PV2 (secondary controller).

2.5 Structure and Mode of Operation of the Standard PID Control

Sampling Control

The controllers that can be implemented with the Standard PID Control are always digital sampling controllers (DDC=direct digital control). Sampling controllers are time-controlled, in other words they are always processed at equidistant intervals (the sampling time or CYCLE). The sampling time or frequency at which the controller is processed can be selected.

Figure 2-8 illustrates a simple control loop with the standard controller. This diagram shows you the names of the most important variables and the abbreviations of the parameters as used in this manual.

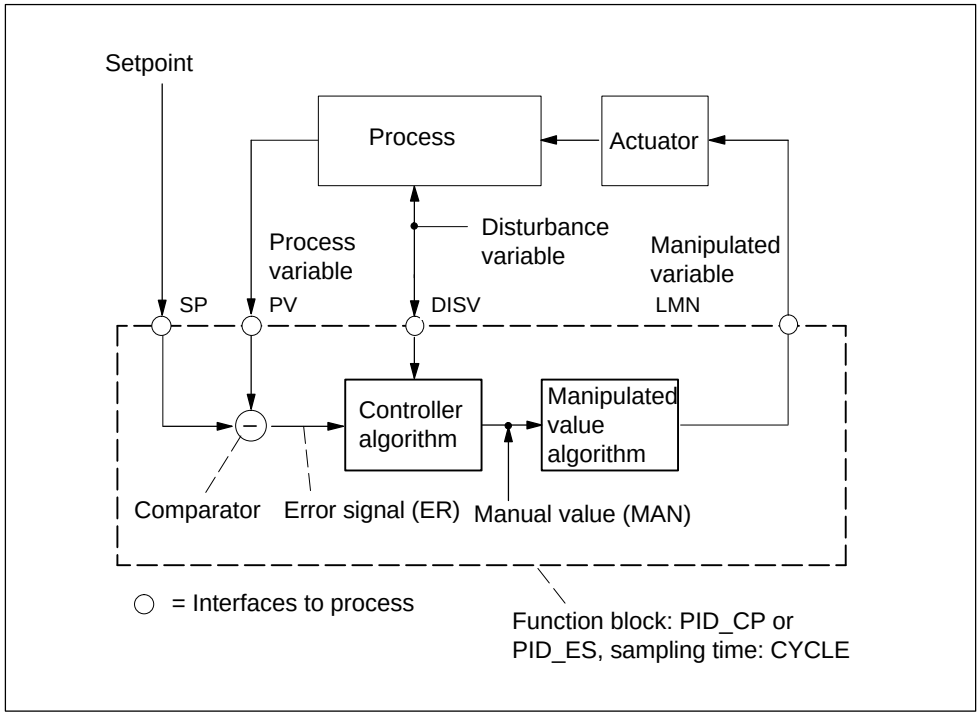


Figure 2-8 Sampling Controller of the Standard PID Control in the Closed Loop

The control functions implemented in the function blocks PID_CP and PID_ES are pure software controllers. The input and output values of the controllers are processed using digital algorithms on a CPU.

Since the processing of the controller blocks in the processor of the CPU is serial, input values can only be acquired at discrete times and the output values can only be output at defined times. This is the main characteristic of **sampling control.**

Control Algorithm and Conventional Control

The control algorithm on the processor simulates the controller under real-time conditions. Between the sampling instants, the controller does not react to changes in the process variable PV and the manipulated variable LMN remains unchanged.

Assuming, however, that the sampling intervals are short enough so that the series of sampling values realistically approximates the continuous changes in the measured variable, a digital controller can be considered as quasi continuous. With the Standard PID Control, the usual methods for determining the structure and setting characteristic values can be used just as with continuous controllers.

This requirement for creating and scaling controllers with the Standard PID Control package is met when the sampling time (CYCLE) is less than 20% of the time constant of the entire loop.

If this condition is met, the functions of the Standard PID Control can be described in the same way as those of conventional controllers. The same range of functions and the same possibilities for monitoring control loop variables and for tuning the controller are available.

The Functions of the “Standard PID Control”

The following diagrams illustrate the preconfigured controller structures of the Standard PID Control as block diagrams. Figure 2-9 represents the continuous controller with the signal processing branches for the process variable and setpoint, the controller and the manipulated variable branch. You can see which functions must be implemented after the signal conditioning at the input and which are not required.

The range of functions of the ”Standard PID Control” is rigid, but can be extended by a user-defined function (FC) in each of the signal processing branches.

Figures 2-10 and 2-11 represent the manipulated value generation with the step controller in the versions with and without position feedback. This makes clear that in the absence of position feedback, a quasi position-proportional feedback signal is generated from the “on” times of the binary outputs.

- You will find detailed descriptions of the functions in *Chapters 4* to *7* of the manual. Background and context-specific information is also available in the on-line help system.
- The structure diagrams in the following section contain details with parameter names and structure or mode switches (*see Section 2.6*).
- You will find a detailed illustration of the entire signal flow in the continuous controller and in the step controller in the block diagrams in *Section 8.2*.

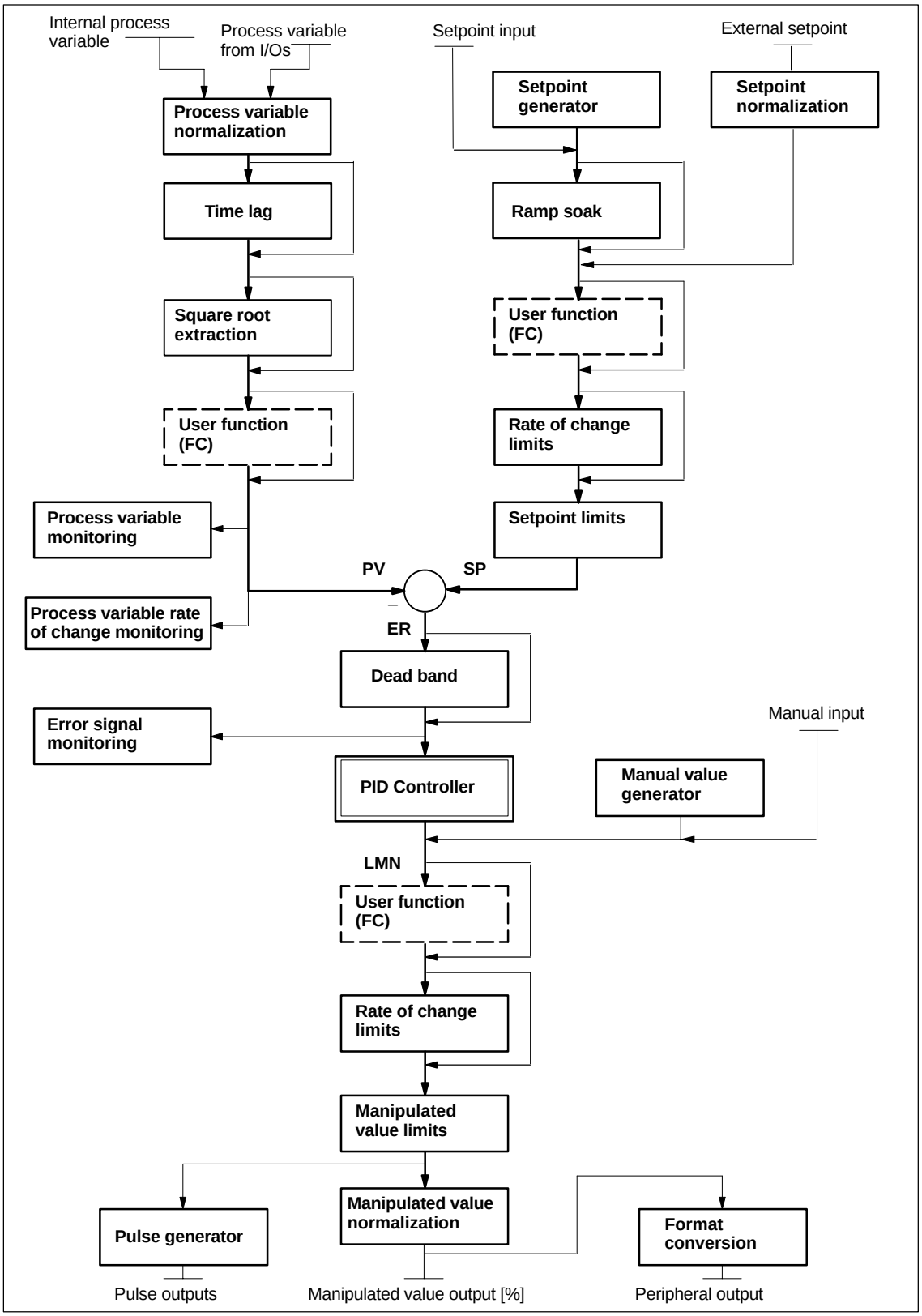


Figure 2-9 Sequence of Functions of the Standard PID Control (continuous controller)

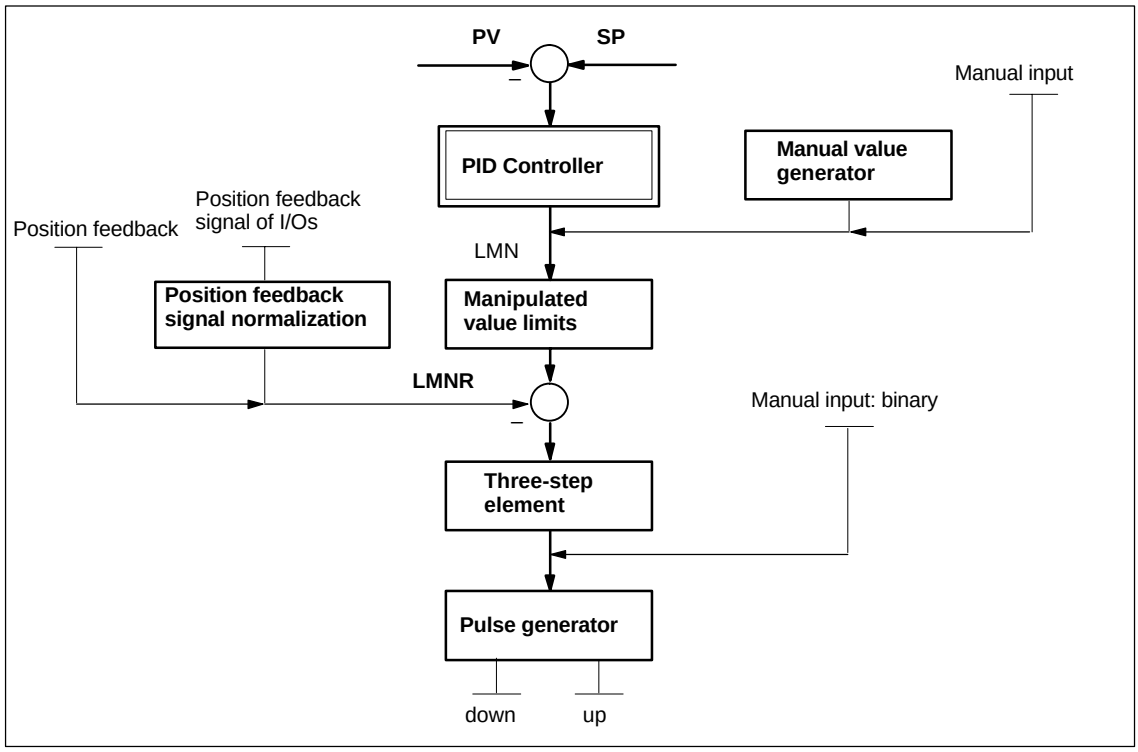


Figure 2-10 Manipulated Variable Branch of the Step Controller with Position Feedback Signal

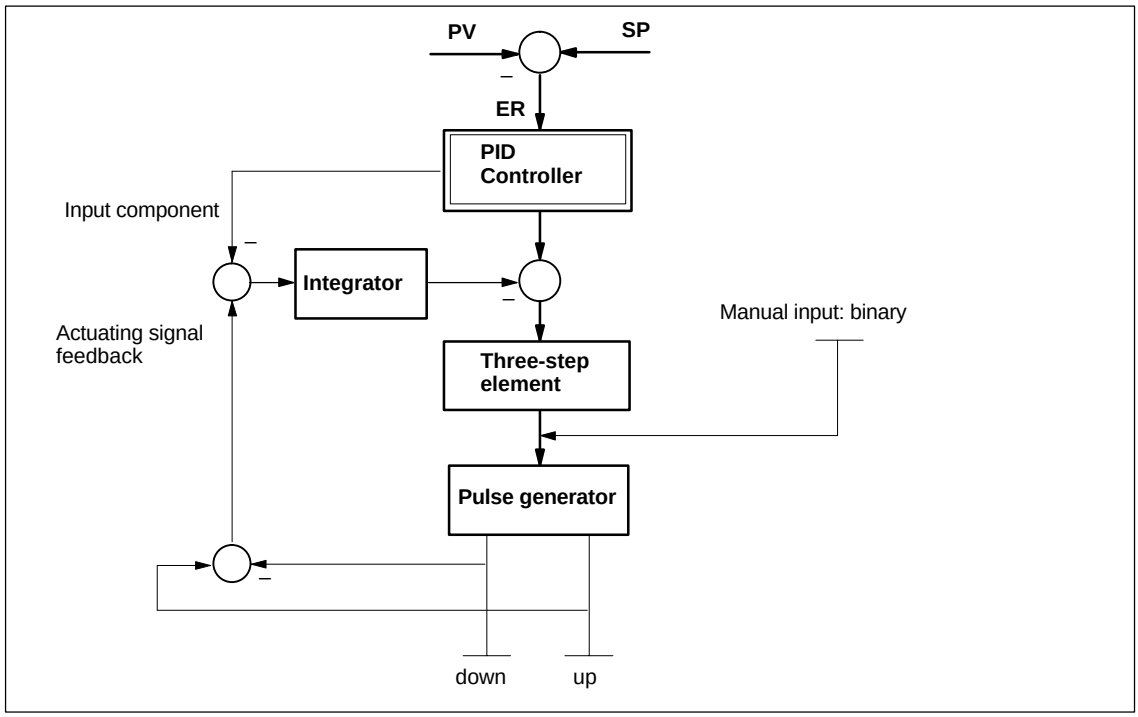


Figure 2-11 Manipulated Value Branch of the Step Controller without Position Feedback Signal

2.6 Signal Flow Diagrams

Signal Flow Diagrams

The following diagrams are overviews of the functions of the Standard PID Control. The number of software switches with which you can select the functions you require is particularly clear to see.

Analogous to the representation of the switches in the configuration tool, the black dot in the switch symbols indicates that the switching symbol has the Boolean value (0=FALSE or 1=TRUE) next to the switch and that the signal path is switched via this dot. The switching signals (binary signals) are indicated by broken lines.

In the diagrams, the subfunctions are represented with the default switch bits for the default signal paths. In the initial situation, practically all the switching signals have the value FALSE (exceptions: P_SEL, I_SEL and MAN_ON=TRUE).

This means that the setpoint is set via SP_INT, the same applies to the input of the process value via PV_IN. The controller function is set to a normal PI controller with the P function in the forward branch. The loop is open and the manipulated variable is influenced in the percentage range by the MAN input. All other functions are either passive or if they cannot be deactivated, they are assigned marginal parameter values so that they have no effect

Symbols and Identifiers in the Signal Flow Diagrams

The names of the connectable process variables are shown on a shaded background. This allows you to recognize where the controller structure can be connected to the S7 I/Os or directly to the measurement components and actuators of the process.

Parameter names including “OP” (for example SP_OP/SP_OP_ON) indicate that an intervention using the configuration tool of the Standard PID Control is possible at this point. The configuration tool has its own interface to the controller FB.

Interim values in the signal can be monitored at the measuring points MP1 to MP12. These interim values are required to match values before triggering smooth changeovers or to be able to check the current statuses of the controller. The measuring point values can be represented statically and dynamically in the curve recorder of the configuration tool.

To make the illustrations clearer, the parameters for setting and selecting the dimensions of the processing functions (algorithms) are indicated beside individual function fields. Please refer to the descriptions in the reference section and to the representation of the individual subfunctions in the following sections.

Signal Processing in the Setpoint Branch

- **Fixed setting of the setpoint value** (SP_GEN)
  With fixed setpoint controllers, the setpoint is selected using a switch at the setpoint generator SP_GEN and is then fixed.
- **Setpoint setting according to a time-controlled program** (RMP_SOAK)
  When controlling processes with different setpoints set according to a time-controlled program, the ramp soak function generates the required curve for the reference input and influences the process so that the process variable changes according to a defined profile.
- **Change limitation for the reference input**(SP_ROC)
  The conversion of setpoint step changes to a ramp-shaped increase or decrease in the reference input prevents large input changes to the process. The SP_ROC function limits the setpoint rate of change separately for the up rate and down rate and for positive and negative values in the reference input.
- **Absolute value limitation for the reference input** (SP_LIMIT)
  To prevent illegal process states occurring, the setpoint is limited by high and low limits (SP_LIMIT).

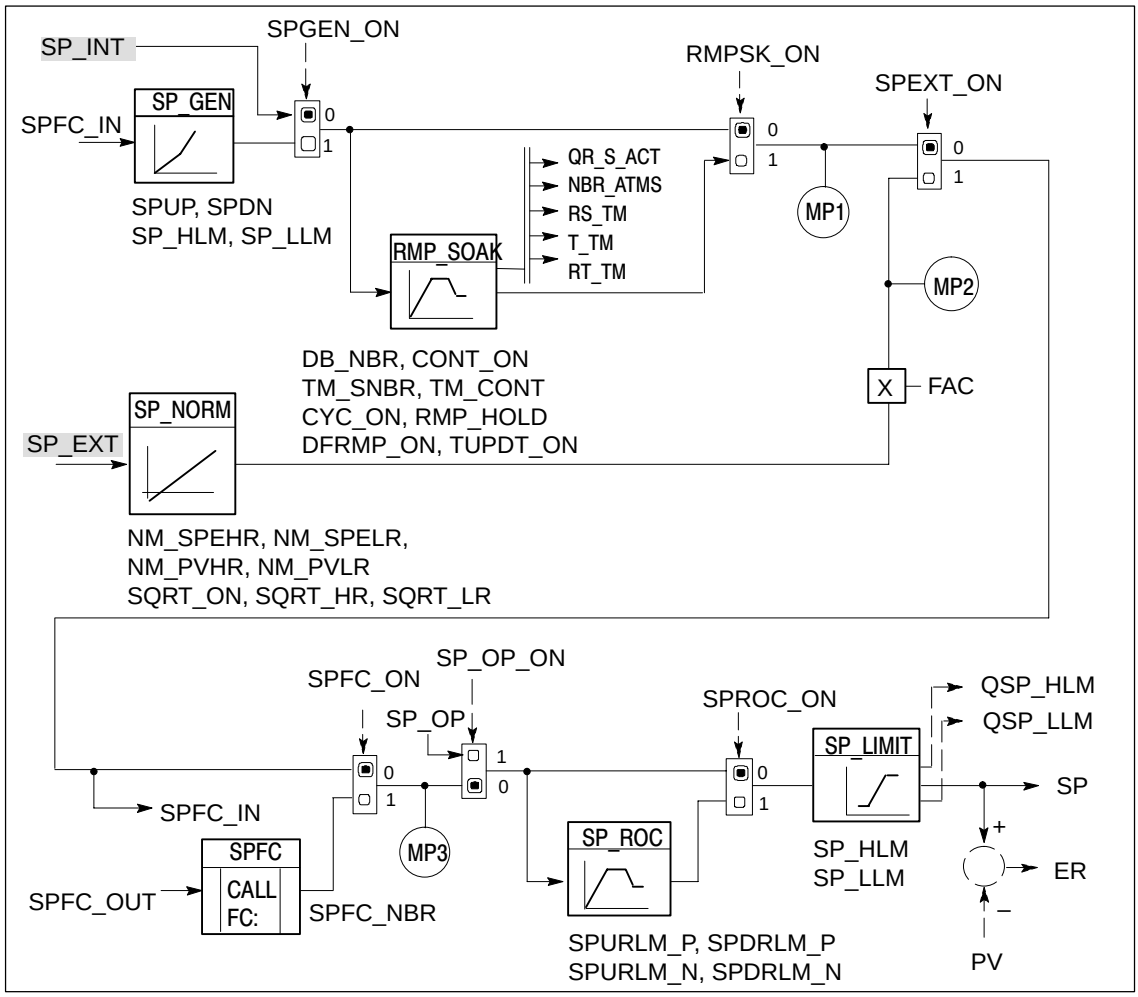


Figure 2-12 Signal Flow Diagram of Setpoint Processing

Signal Processing in the Setpoint Branch

- **Delay of the process variable** (LAG1ST)

  To reduce the effects of noise on process signals, a first order time lag is used in the process variable branch. This function dampens the analog process variable more or less depending on the time constant PV_TMLAG. Disturbance signals are therefore effectively suppressed. Overall, however, the time constant of the total control loop is increased, in other words, the control action becomes slower.

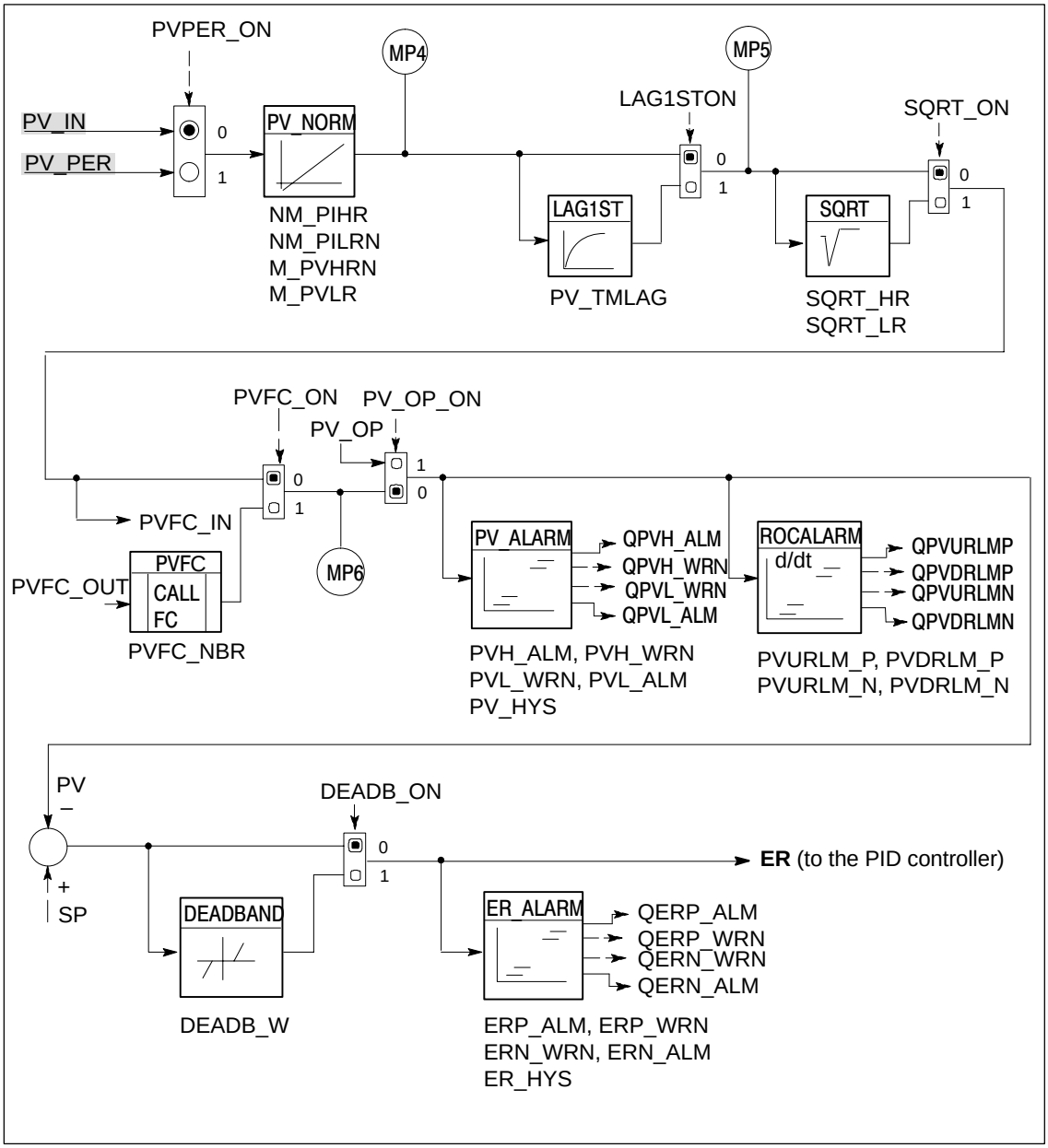


Figure 2-13 Signal Flow Diagram of the Process Variable and Error Signal Processing

- **Extracting the root of the process variable** (SQRT)
  When the relationship of the measured signal to the physical value is quadratic (flow measurement using a differential flow meter) the process variable must be linearized by extracting the root (square root algorithm). Only a linear value can be compared to the linear setpoint for the flow and processed in the control algorithm. For this reason, the SQRT function element can be included in the process value branch as an option.
- **Monitoring the Process Variable Rate of Change** (ROCALARM)
  If the rate of change of the process variable is extremely high or too high, this points to a dangerous process state to which the programmable logic controller may have to react. For this reason, the ROCALARM function generates alarm signals if selectable rates of change (positive or negative) are detected in the process variable. The alarm signals can then be further processed to suit the particular situation.
- **Monitoring the Absolute Value of the Process Variable and Error Signal**
  Two limit values are set for the process variable and the error signal and are monitored by the PV_ALARM and ER_ALARM functions.
- **Superimposing by Signal Noise** (DEADBAND)
  To filter out noise on the channels of the process variable or the external reference input, the error signal passes through a selectable dead band component. Depending on the amplitude of the noise, the dead band width can be selected for the signal transmission. Falsification of the transmitted signal must, however be accepted as a side effect of the selected dead band.

Signal Processing in the PID Controller

- **Normal PID Controller Function**
  The switch states shown in Figure 2-14 implement a PI controller with parallel processing of the signals of the P and I action. D-SEL = TRUE supplements the control algorithm for the parallel processing in the D branch. ± GAIN is used to determine the proportional gain or the gain of the controller. A negative sign means that the manipulated variable falls while the process variable is rising.
- **PD in the feedback path**
  If the P and D actions are moved to the feedback path (PFDB_SEL and DFDB_SEL = TRUE) then step changes in the setpoint do not result in step responses in the manipulated variable. The factor has a negative effect on the feedback influence.

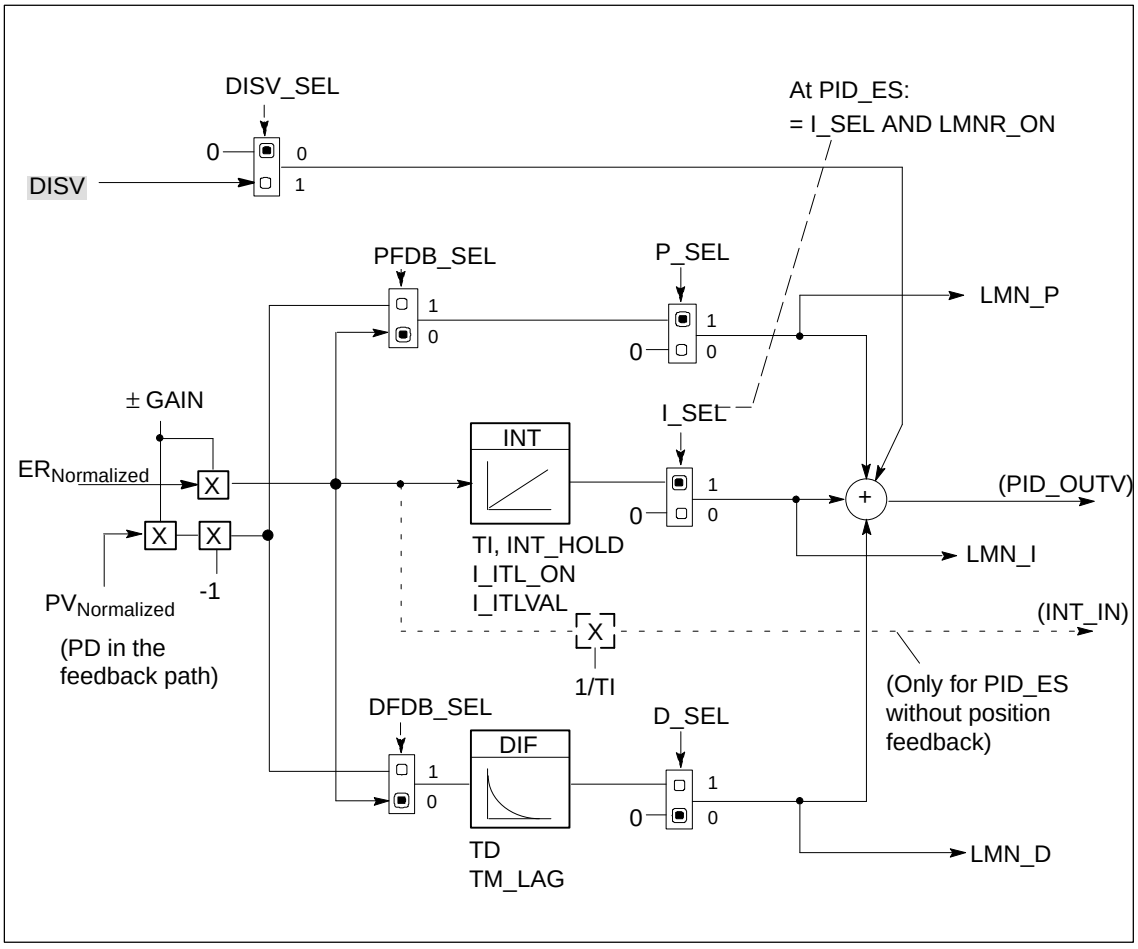


Figure 2-14 Signal Flow Diagram of the Control Function

Signal Processing in the Branch of the Analog Manipulated Variable

- **Fixed Setting of the Manual Value** (MAN_GEN)
  In the manual mode (open loop), the manipulated value is selected at the manual value generator MAN_GEN using a switch and is fixed.
- **Change Limitation of the Manipulated Variable** (LMN_ROC)
  Converting extremely fast step changes in the manipulated variable into a ramp-shaped rise or fall in the manipulated variable prevents sudden changes in the input to the process. The function (LMN_ROC) limits the manipulated value rate of change both up and down.
- **Absolute value limitation for the Manipulated Variable** (LMNLIMIT)
  To avoid illegal process states or to restrict the movement of an actuator, the upper and lower limits of the range of the manipulated variable are set with LMNLIMIT.
- **Activating the Cascade Control**
  Depending on the combination of the switching states of the Standard PID Control the OR gate generates an enable signal for the cascade coupling.

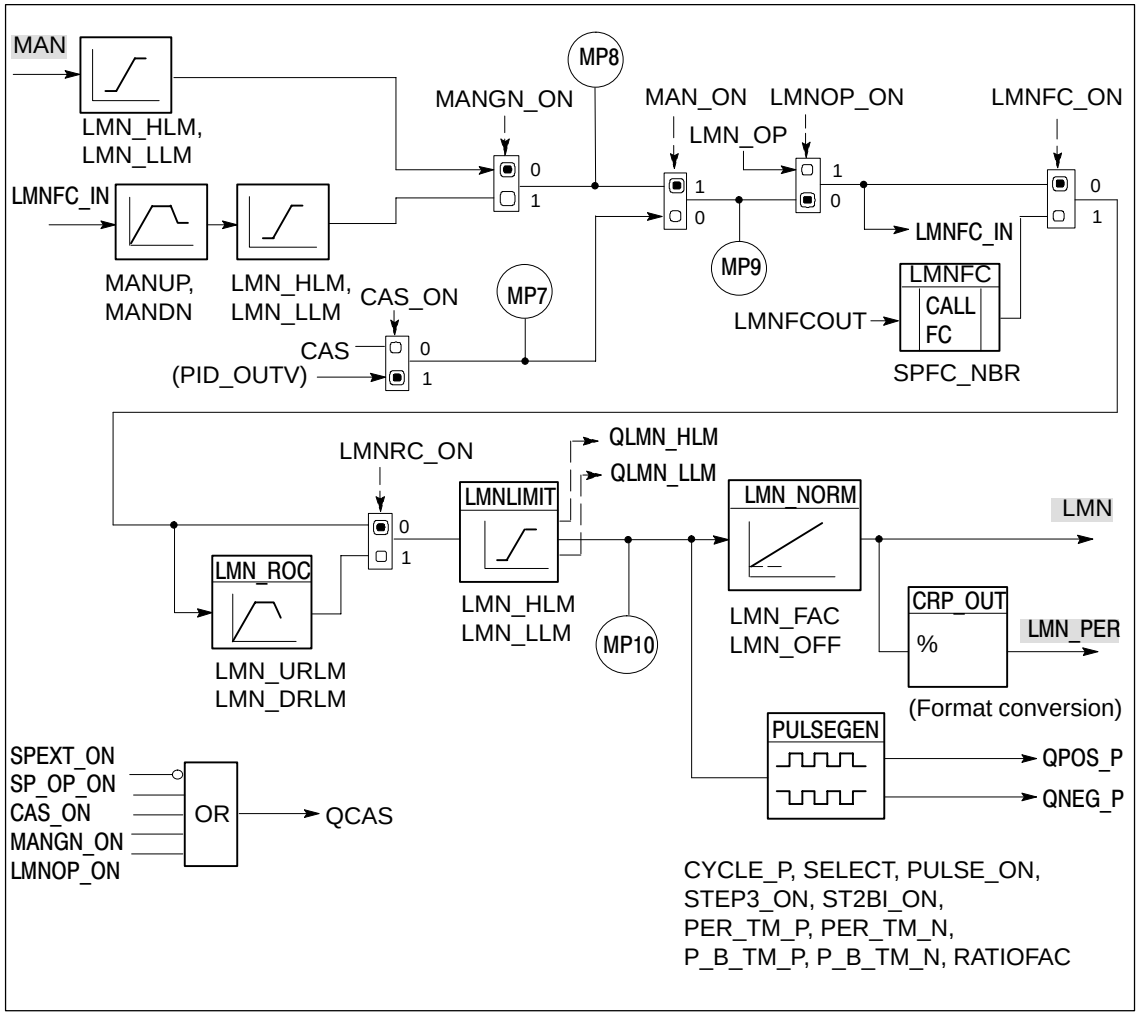


Figure 2-15 Signal Flow Diagram of Actuating Signal Generation with the Continuous Controller

Actuating Signal Processing: Step Controller With Position Feedback

- **Fixed Setting of the Manual Value and Manipulated Variable Limitation**
  The functions for setting the manual value and for limiting the absolute value of the output variables are the same as for the controller with a continuous output.
- **Forming the Binary Actuating Signal** (THREE_ST, PULSEOUT)
  Depending on the sign of the error signal, the three-step switch THREE_ST generates a positive or negative output pulse via the pulse shaping stage PULSEOUT, that is applied until the input variable disappears. The self-tuning hysteresis prevents the output switching too often.

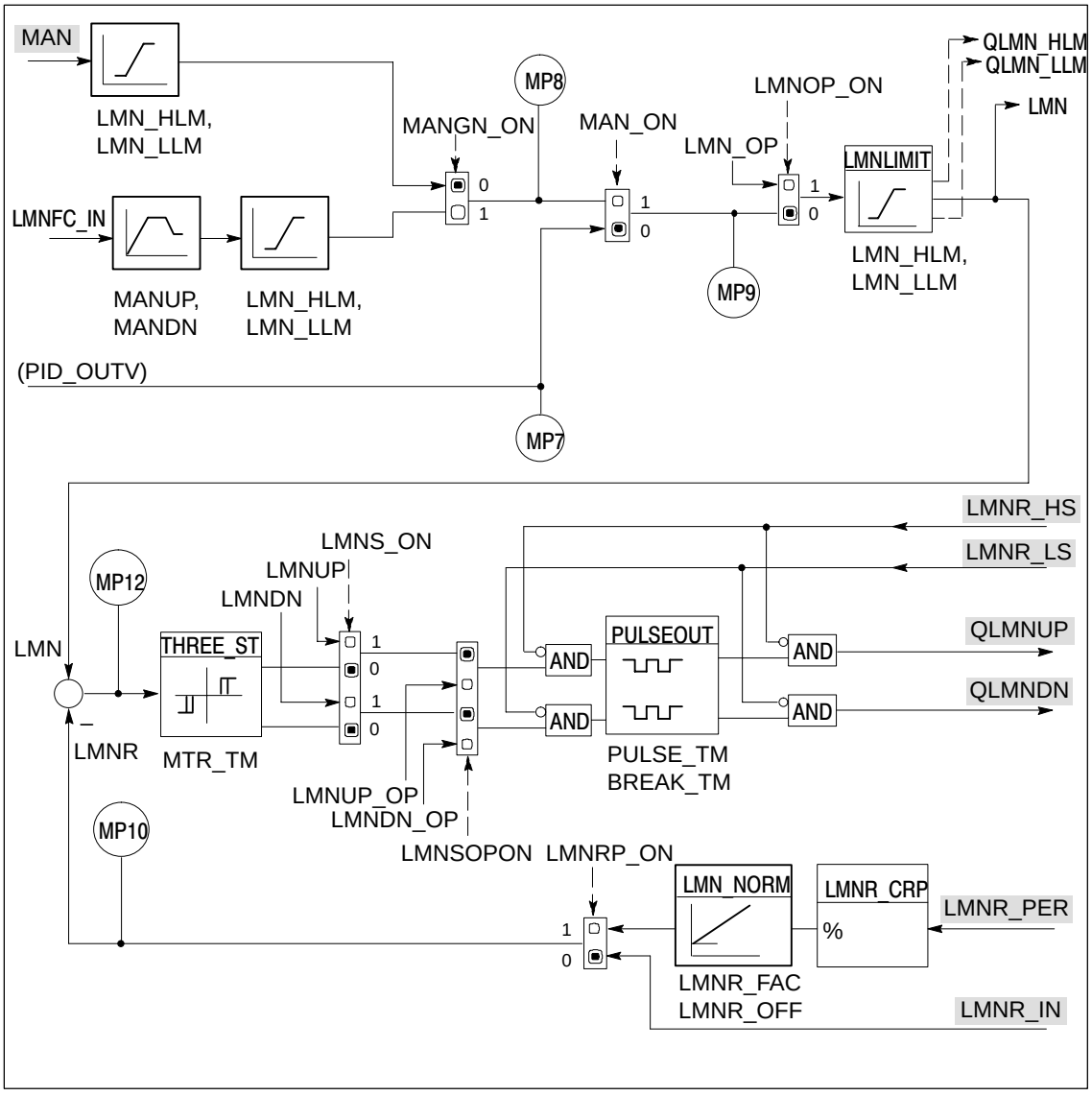


Figure 2-16 Signal Flow Diagram of Manipulated Variable Generation on the Step Controller with Position Feedback Signal (LMNR_ON = TRUE)

Actuating Signal Processing: Step Controller Without Position Feedback

- **Generation of the Binary Actuating Signal**
  The generation of the output signal through three-step switch with hysteresis and pulse-generating stage is identical for all step controllers. The time parameters for the consideration of the controller acting time of the motor drive (MTR_TM) and the setting of the pulse/pause duration (PULSE_TM and BREAK_TM) can be adjusted.
- **Simulation of the Position Feedback**
  The automatic acquisition of the control parameters with the process identification function of the configuration tool always requires a signal as an input variable representing the position of the actuator. The simulation function does not require parameter assignment and is irrelevant for the normal operation of the step controller.

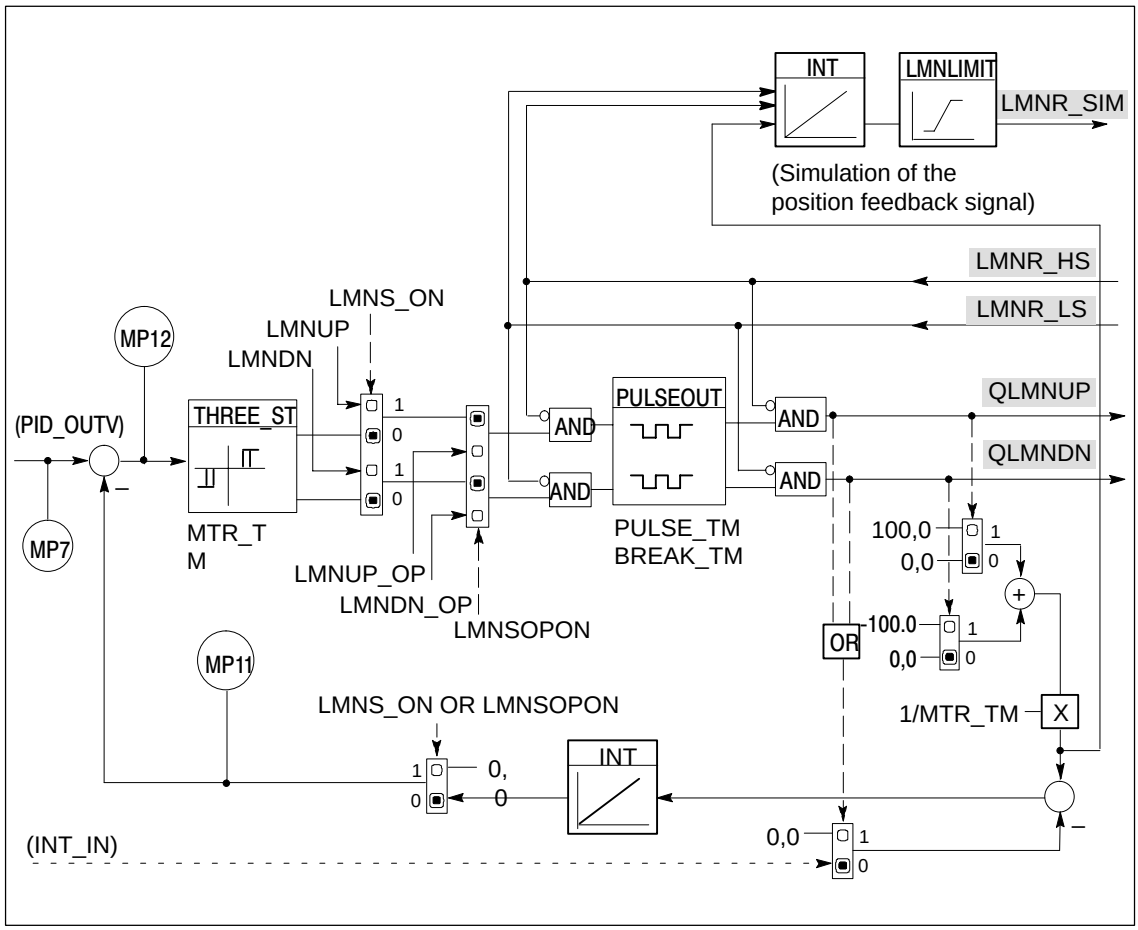


Figure 2-17 Block Diagram of Manipulated Variable Generation on the Step Controller Without Position Feedback Signal (LMNR_ON = FALSE)

Configuring and Starting the Standard PID Control

3

3.1 Defining the Control Task

Specifying the Task

Before you implement a control loop using the Standard PID Control package, you must first clarify the technical aspects of the process you want to automate, the programmable logic controller you will be using and the operating and monitoring environment. To specify the task in detail, you therefore require the following information:

1. You need to know the process you want to control, in other words the characteristic data of the process (gain, equivalent time constant, disturbance variables etc.).
2. You must choose the CPU on which you want to install the Standard PID Control.
3. You must define the signal processing and monitoring functions along with the basic functions of the controller.

Section 2.1 already described the process characteristics and how to determine the characteristic variables for the process response and if you are specifying a concrete task, you should refer to the information there. This section and contain information and explanations about identifying system characteristics and controller parameters using the configuration tool.

Using the configuration tool relieves you of many of the tasks (point 1.) necessary for identifying the characteristic process variables.

What You Should Know before Working with the Controller

Since the Standard PID Control package creates software controllers based on the standard function blocks (here PID_CP or PID_ES) from the range of S7 blocks, you should be familiar with handling S7 blocks and with the structure of S7 user programs (for example in the S7 STL programming language).

Although the functions of the implemented controller are defined solely by assigning parameters, the connection of the controller block to the process I/Os and its integration in the call system of the CPU requires knowledge that cannot be dealt with within the scope of this manual.

You require the following information:

- Working with STEP 7 (**/231/**)
- The basics of programming with STEP 7 (**/232/**, /234/)
- Data about the programmable logic controller you are using (**/70/**, **/71/**, **/100/**, **/101/**).

The Process

There are almost no restrictions in terms of the type and complexity of the processes that can be controlled with the Standard PID Control. Providing the system is a single input-single output system without a derivative transfer action and without all-pass components, all process types whether self-regulating processes or not, in other words without or with I components can be controlled (Figure 3-1).

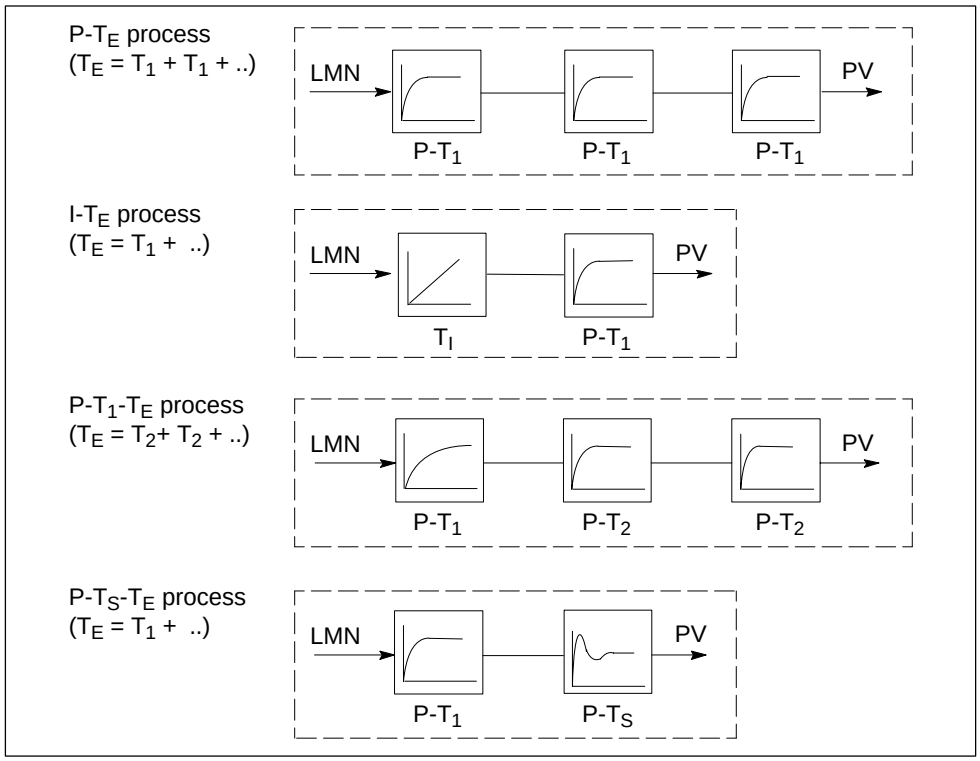


Figure 3-1 Types of Process that can be Controlled with Standard PID Control

The process variable (PV) to be processed by the Standard PID Control is always an analog physical variable (voltage, current, resistance etc.) that is digitized by an S7 analog input module and converted to the uniform STEP 7 PV_PER I/O signal. The values of these signals are saved in memory cells or areas of the CPU user memory. These areas can be addressed using absolute addresses or using symbolic addresses after making the appropriate entries in the symbol table of the CPU.

If, in special situations, the process variable exists as a floating point number, this value can be interconnected directly to the PV_IN input as the controlled variable (Figure 3-2).

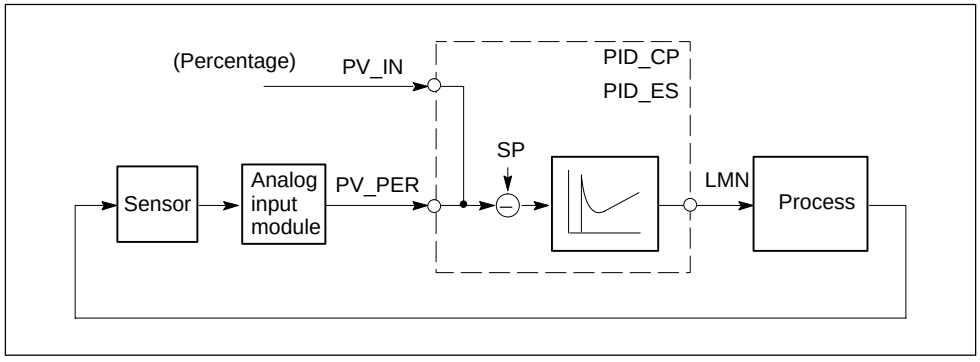


Figure 3-2 Interconnecting the Process Signals to the Standard PID Control

Type of Actuator

To select a suitable configuration for the Standard PID Control, the type of actuator used to influence the process variable is important. The type of signal required by the actuator determines the way in which signals are output in the manipulated variable branch (continuous or discontinuous) and therefore also the type of controller to be used (continuous controller or step controller).

In the great majority of cases, some form of valve will be used to adjust material or energy flow. Different actuating signals are required depending on the drives used to adjust these valves.

1. Proportional actuators with a continuous actuating signal.

   The opening of an orifice, the angle of rotation or a position is adopted proportional to the value of the manipulated variable, in other words within the actuating range, the manipulated variable operates in an analog manner on the process.

   The actuators in this group include pneumatic diaphragm actuators and electro-mechanical actuators with position feedback signals with which a positioning control loop can be created.

2. Proportional actuators with a pulse-width modulated signal.

   With these actuators, a pulse signal is output with a length proportional to the value of the manipulated variable at the sampling time intervals. This means that the actuator (for example a heating resistor or heat exchanger) is switched on for a length of time depending on the manipulated variable.

   The actuating signal can be either monopolar representing the states on or off or bipolar, representing for example the values open/close, forwards/backwards, accelerate/decelerate.

3. Actuators with an integral action and three-step actuating signal.

   Actuators are often driven by motors in which the duration of the “on” time is proportional to the travel of the valve plug. Despite different designs, these actuators all share the same characteristic in that they correspond to an integral action at the input to the process. The Standard PID Control with a step output provides the most economical solution to designing control loops including actuators with an integral action.

Selecting the Controller in Terms of the Actuating Signal

Depending on the type of actuating signal generated, the Standard PID Control provides various structures in the manipulated variable branch.

- Actuators complying with points 1. and 2. in the previous description are controlled with the PID_CP controller block. If a pulse duration modulated signal is required, the PULSEGEN block (FB) must be added to the controller FB.
- Actuators with an integral action (point 3.) are controlled by the PID_ES controller block. If a position feedback signal is not available from the actuator, the controller structure with a simulated feedback signal (LMNR_ON=FALSE) is used.

  If a transmitter is available to indicate the position of the actuator, the structure can be configured with a positioning control loop (LMNR_ON=TRUE), refer to (Figure 3-3 bottom example).

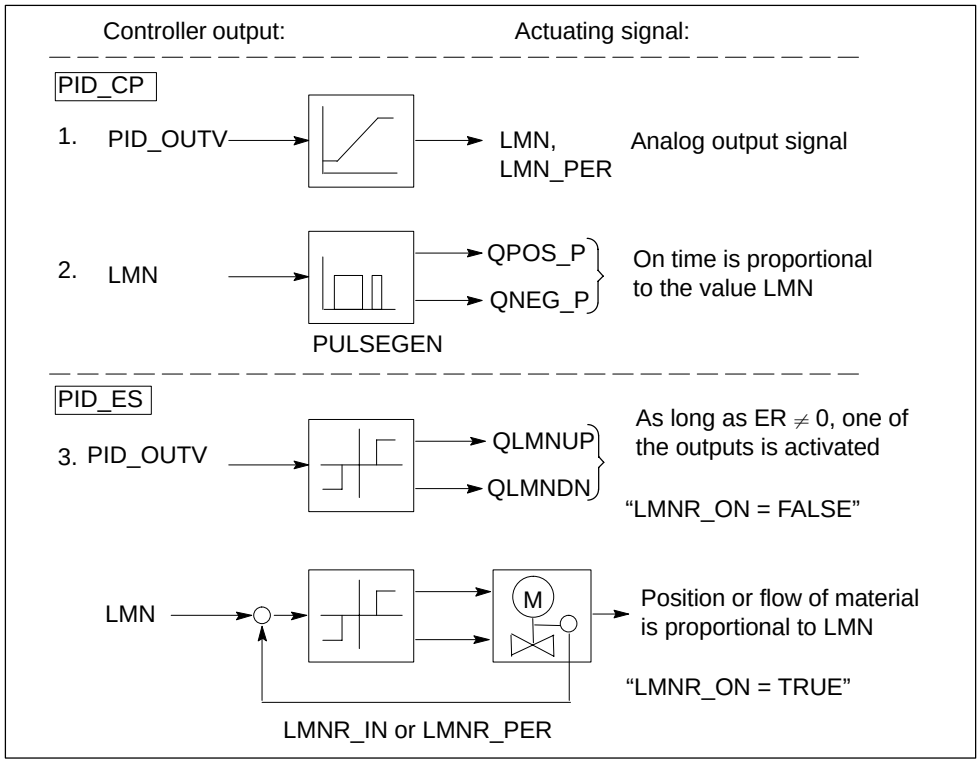


Figure 3-3 Actuating Outputs of Standard PID Control

Note

The manipulated variables are represented as digital numerical values in the floating point or peripheral (I/O) format or as binary signal states. Depending on the actuator being used, modules must always be connected to the output to convert the signals to the required type and to provide the required actuating energy.

Actuating Signals and Controller Blocks

The relationships between the type of signal used for the manipulated variable, the type of control and the configuration of the controller blocks required to implement them is shown in the following table.

Table 3-1 Manipulated Variable, Type of Control and Required Controller Blocks

Type of Signal of the Manipulated Variable	Values	Type of Controller	Controller Structure
Proportional	Floating point 0.0 to 100. 0 % or peripheral range	Continuous	PID_CP
Pulse duration modulated, with 2-step controller, outputs alternating	Bipolar or monopolar Positive output: TRUE Negative output: FALSE	Three-step/ two-step controller	PID_CP + PULSEGEN
Three-step discontinuous	Up – 0 – down	Step controller	PID_ES (LMNR_ON = FALSE)
Three-step discontinuous position feedback signal	Up – 0 – down 0..100 % or peripheral range	Step controller with position feedback signal	PID_ES (LMNR_ON = TRUE)

The explanations above should provide you with all the information you require to select a suitable configuration of the Standard PID Control for your particular situation. The best way of doing this and how you activate and dimension internal functions is explained in the following section.

Permanent Functions that Cannot be Deactivated

The functions for monitoring and limiting the signals in the branches for input and output signal processing are always active and cannot be deactivated. These include the following:

- setpoint limitation SP_LIMIT
- process variable absolute limit alarms and warnings PV_ALARM
- process variable rate of change alarms ROCALARM
- error signal limit alarms and warnings ER_ALARM
- and manipulated value limits LMNLIMIT

When you have decided which controller block to use and have defined its inputs/outputs you must always make sure that suitable values are assigned to the functions listed above.

Note

The defaults have been selected (usually at the extremes of the working ranges available) so that operation can be started without selecting any individual parameters. The parameters can then be adapted to the requirements.

3.2 Configuring a Project ”Configuring” (Checklist)

Generating the Control Project Configuration

Now that you have worked through the required control and monitoring functions (or information, refer to *Sections 2.5 and 3.1),* this section now shows you the step-by-step implementation of these functions.We recommend that you create your configuration following the steps outlined below (checklist):

Step	Activity	Function in Standard PID Control	Explanation
1.	Select the controller blocks or block configuration required for your controller structure. Select and copy a configuration example closest to the configuration you want to implement.	FB ”PID_CP” or ”PID_ES” or an example from Example1 ... Example6 or Getting Started	– Section 3.3
2.	Based on the selected example, configure the required controller by including or omitting preconfigured functions or by including your own.	– Set the structure switch in the block diagram of the configuration tool; – or set the switching bits of the structure switch in the instance DB (→block diagrams in Appendix A).	The data structure of the instance DB is supplied by the the relevant FB.
3.	Select the sampling time and and calls of the control loop: – Specify the startup response with OB100 – Decide on the sampling time and priority class, if necessary, change the call interval of the cyclic interrupt OB – Configure the loop scheduler to suit the number of loops on the CPU.	Parameter COM_RST Parameter: CYCLE, Organization block: OB35 Loop scheduler: LP_SCHED, included in examples Example 3 to Example 5	– Section 3.4, Section 3.5 – Section 7.1
4.	Assign parameters and use the conversion functions for the measuring range and zero point adaptation of the input/output signals	– Normalization of the external setpoint (SP_NORM) – Normalization of the external process variable (PV_NORM) – Manipulated value denormalization (LMN_NORM)	(→Chapter 4) – Section 3.6 – Section 3.6
5.	Configure the setpoint branch	– Setpoint generator (SP_GEN) – Ramp soak (RMP_SOAK) – Limits of the setpoint rate of change (SP_ROC) – Limits of the absolute values of the setpoint (SP_LIMIT)	(→Chapter 4) – The function is always active.

Step	Activity	Function in Standard PID Control	Explanation
6.	Configure the process variable branch	– Process variable time lag (LAG1ST) – Square root extraction (SQRT) – Monitor the absolute values of the process variable (PV_ALARM) – Monitoring the rate of change of the process variable (ROCALARM)	(→Chapter 4) – The function is always active. – The function is always active.
7.	Configure error signal generation	– Dead band of the error signal – Monitoring the error signal for absolute values (ER_ALARM)	(→Chapter 4) – The function is always active.
8.	Configure the manipulated value branch for continuous controllers	– Manual value generator (MAN_GEN) – Limits of the rate of change of the manipulated value (LMN_ROC) – Limits of the absolute values of the manipulated value (LMNLIMIT)	(→Chapter 5) – The function is always active.
	Configure the manipulated value branch for step controllers	– Manual value generator (MAN_GEN) – If there is a position feedback signal: Limits of the absolute values of the manipulated value (LMNLIMIT) – Operating parameters for three-step elements and and pulse generator stage (THREE_ST and PULSEOUT)	(→Section 6) – The function is always active.
9.	Configure controller	– PID controller structure and PID parameters – Operating point for P and PD controllers – Feedforward control (DISV)	(→Chapter 5)
10.	If necessary, include extra functions in the form of a user FC in the setpoint, process variable and/or manipulated value branch.	– SPFC (SPFC_ON = TRUE) – PVFC (PVFC_ON = TRUE) – LMNFC (LMNFC_ON = TRUE)	
11.	Load the configured controller on the CPU of the PLC.	– Load the project in the S7 Manager.	
12.	If required, perform an off-line test of the configured standard controller with the simulated third order delay process.	– Model process contained in Example 1 and Example 2	
13.	Interconnect the block inputs and outputs of the configured standard controller with the process I/Os.	– Program the connections of the inputs/outputs with the absolute or symbolic I/O addresses in the user memory of the CPU.	

The following section explains the activities for configuring individual functions or points in the checklist in greater detail where necessary. A schematic parameter assignment plan summarizes the functions of the Standard PID Control with all the configuration and function parameters.

Based on this plan, you can see which parameters belong to a function and the possible settings for the parameters.

3.3 Configuring the Standard PID Control

Parameter Assignment Plan for Configuring the Standard PID Control

If you want to create your configuration directly in the instance DB, the parameter assignment plans provide you with a graphic overview of the individual functions you need to select and assign parameters too.

When you are implementing an actual controller, remember that the configuration tool largely relieves you of the need to check your entries for completeness.

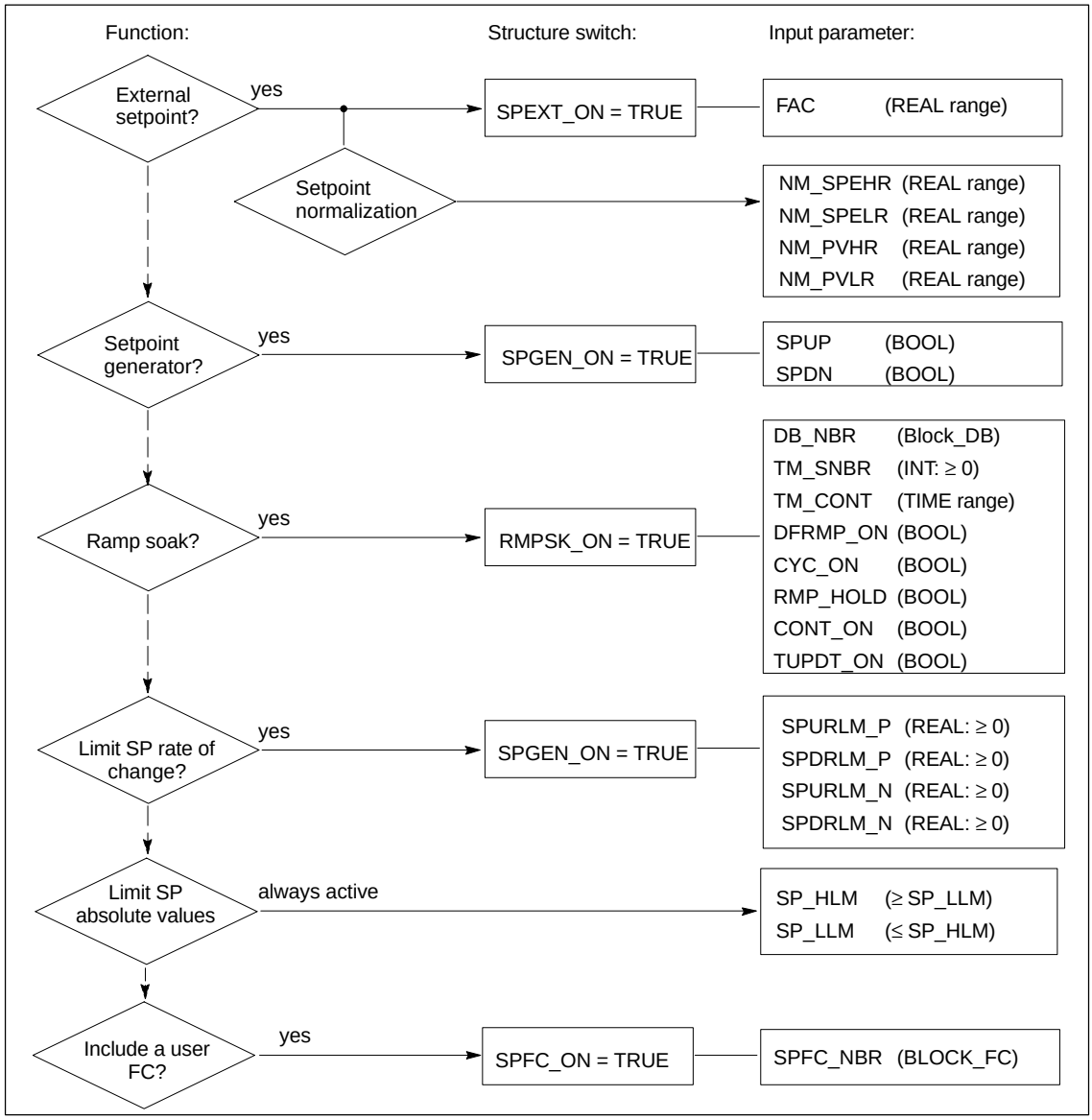


Figure 3-4 Configuration of the Setpoint Branch of the Standard PID Control (checklist points 4. and 5.)

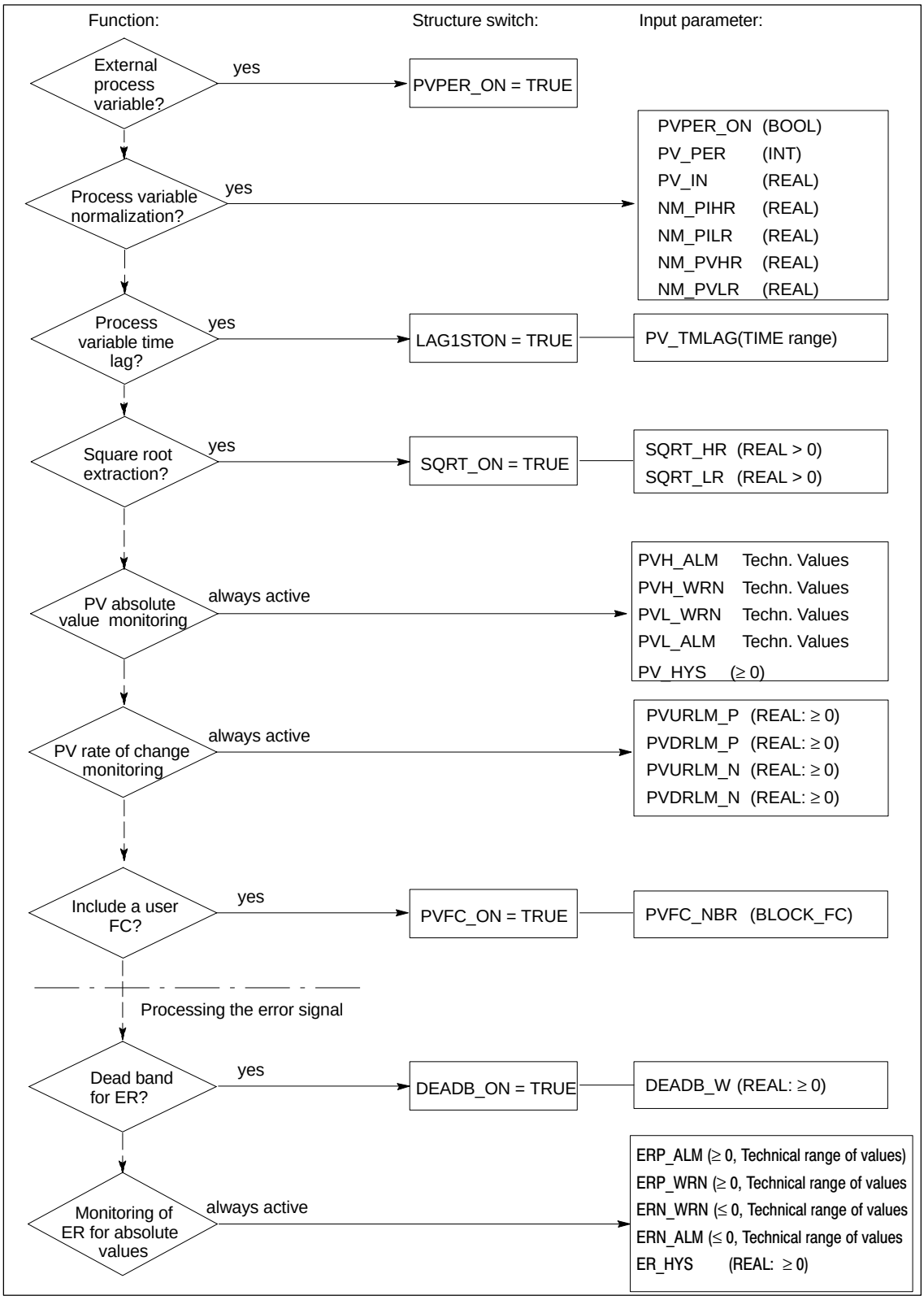


Figure 3-5 Configuration of the Actual Value and Error Value Branch of Standard PID Control (checklist points 6., 7. and 8).

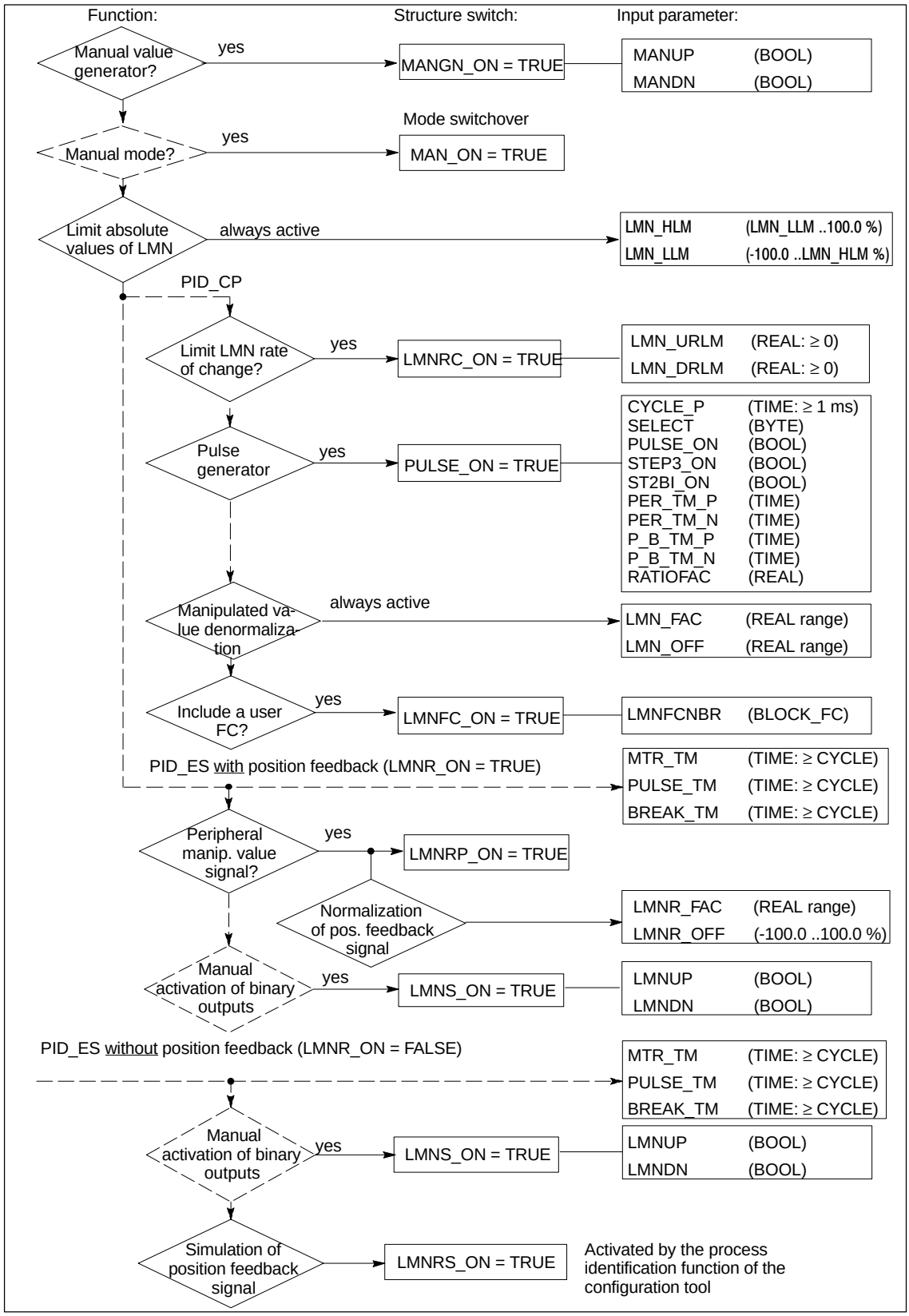


Figure 3-6 Configuration of the Manipulated Value (checklist point 8.)

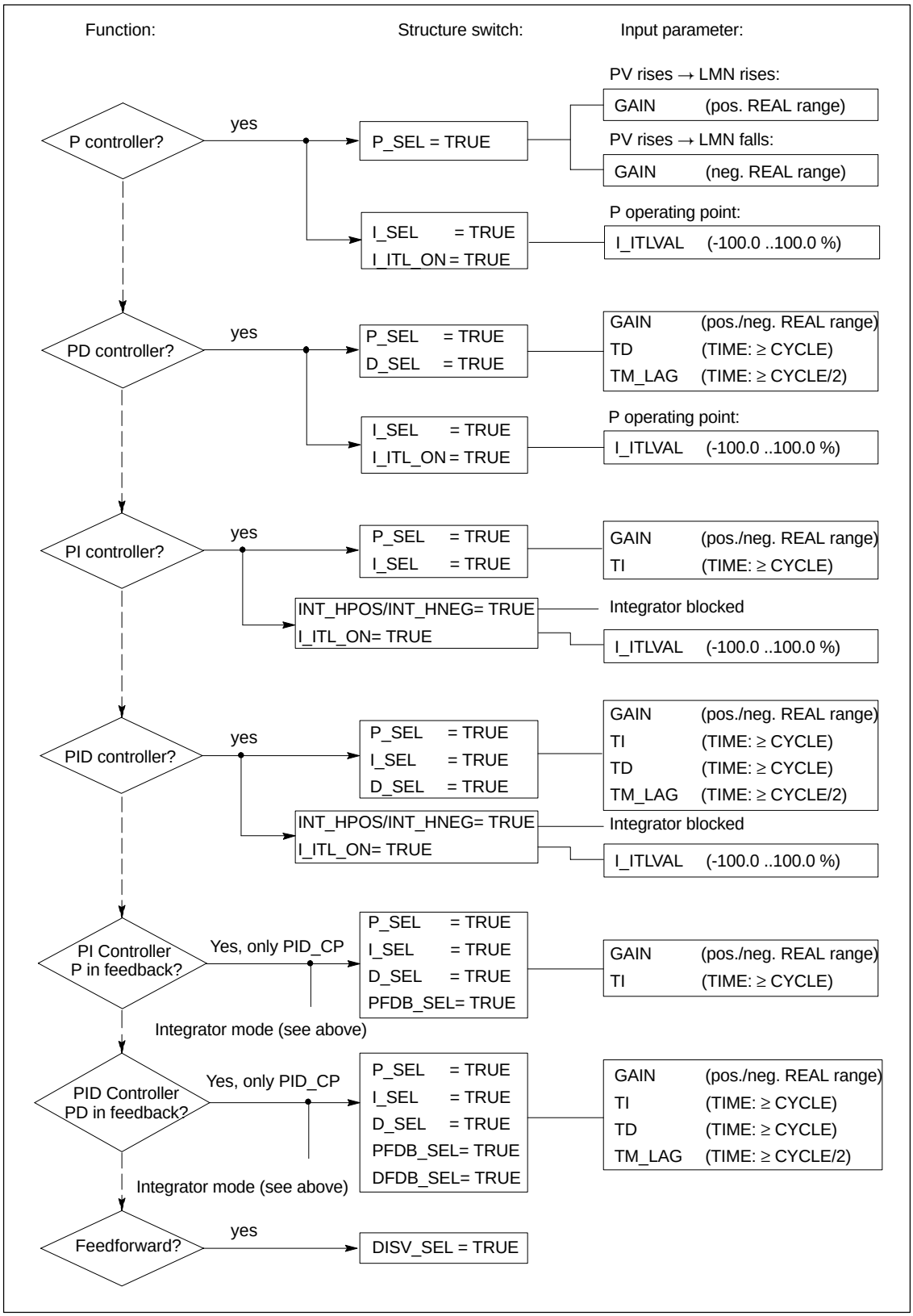


Figure 3-7 Configuration of the Controller Functions PID_CP and PID_ES (checklist point 9.)

The Configuration Tool

If the procedure for configuration is in the checklist (*Section 3.2*) or the information in the parameter plans is too complicated or would involve too much time, we recommend that you use the configuration tool for the Standard PID Controller.

The configuration tool contains the following tools with which you can configure the Standard PID Control quickly and free of errors:

Loop Editor

The block diagram of the loop editor contains the most important functions of the Standard PID Control represented as block symbols. By clicking the switch symbols (dark point) you can specify the signal flow you require both quickly and easily.

After you click a function field, the system opens a dialog box in which you can dimension the functions by making entries in parameter fields. If the function is not displayed in the block diagram as an explicit switching function, you can activate or deactivate it using the option buttons or check boxes.

3.4 The Sampling Time CYCLE

The Sampling Time: CYCLE

The sampling time is the basic characteristic for the dynamic response of the Standard PID Control. This decides whether or not the controller reacts quickly enough to process changes and whether the controller can maintain control in all circumstances. The sampling time also determines the limits for the time-related parameters of the Standard PID Control.

Selecting the sampling time is a compromise between several, often contradictory requirements. Here, it is only possible to specify a general guideline.

- The time required for the CPU to process the control program, in other words to run the function block, represents the lowest limit of the sampling time ($CYCLE_{min}$).
- The tolerable upper limit for the sampling time is generally specified by the process dynamics. The process dynamics is, in turn, characterized by the type and the characteristics of the process.

Equivalent System Time Constant

The most important influence on the dynamics of the control loop is the equivalent system time constant (T_E) that can be determined after entering a step change DLMN by recording the unit step response at the system input (Figure 3-8).

The system value T_E represents a useful approximation of the effective time lag caused by several P-T_1, P-T_S and T_t elements in the loop. If, for example the same PT $_1$-elements are connected in series, it is the sum of the single time constants.

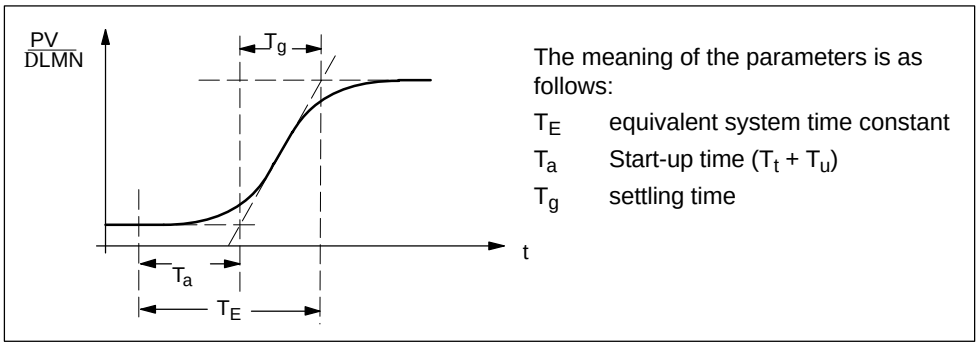


Figure 3-8 Determining the Equivalent System Time Constant T_E

Sampling Time Estimate

If a minimum speed is required for the control, you can specify a maximum sampling time $CYCLE_{max}$.

With P-T_E processes in which the first delay element is predominant and $T_1 \geq 0.5\ T_E$ make sure that:

$$CYCLE_{max} \leq 0.1 * T_E$$

For all other P-T_E-processes:

$$CYCLE_{max} \leq 0.2 * T_{E\ is\ adequate}$$

See ***/352/*** for a precise estimation of the sampling time.

Rule of Thumb for Selecting the Sampling time

Experience has shown that a sampling time of approximately $^1/_{10}$ of the time constant T_{EG} determining the step response of the closed loop produces results comparable with the conventional analog controller.

The total time constant of the closed loops is

obtained in a way similar to that shown in Figure 3-8, by entering a setpoint step change and evaluating the settling of the process variable.

$$CYCLE = \frac{1}{10}\ T_{EG}$$

3.5 How the Standard PID Control is Called

Calling the Standard PID Control

Depending on the sampling time of the specific controller, the controller block must be called more or less often but always at the same time intervals. The operating system of the S7 PLC calls the cyclic interrupt organization block OB35 every 100 ms. The cyclic interrupt clock rate can be configured from 1 ms to 1 minute. The standard setting for OB35 is 100 ms.

If you require several controllers or controllers with large sampling times, you should use the loop scheduler (LP_SCHED).

Complete restart:

When the controller FB is called during a complete restart (OB100), the complete restart bit COM_RST is set and the CYCLE sampling time is transferred. The complete restart routine in the FB then sets a defined initial status for the Standard PID Control.

Restart:

During a restart, processing continues at the status that existed when the interruption occurred. The controller continues working using the values that it had calculated at the time of the interruption.

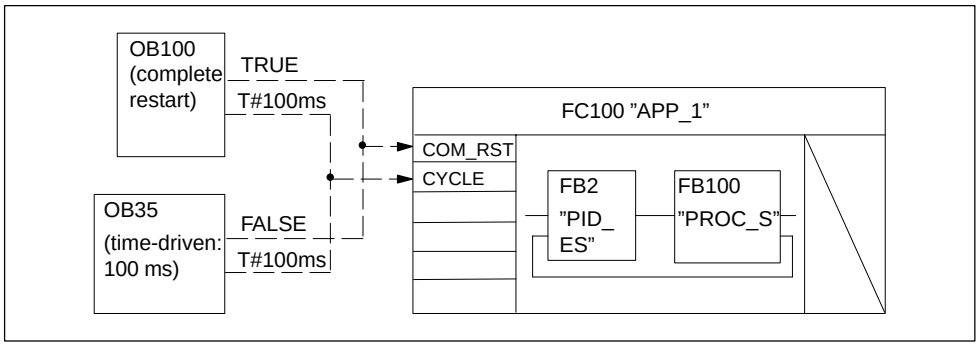


Figure 3-9 Connecting the Start-Up Blocks with the Sample APP_1

Note

In the case of the continuous closed-loop controller PID_CP without pulse output the sampling time is configured via the CYCLE call parameter.

In the case of the continuous closed-loop controller PID_CP with pulse output the the watchdog-interrupt cycle or the sampling time specified via the controller call distribution at the CYCLE_P call parameter (see Section 5.4.).

Using the Loop Scheduler

If the cyclic interrupts of the priority class system are inadequate for the required number of controllers or when controllers are being used with larger sampling times than the longest timebase of the existing cyclic interrupts, a loop scheduler must be integrated into the cyclic interrupt OB.

The loop scheduler LP_SCHED allows several controllers to be incorporated in one cyclic interrupt priority class. These can then be called more or less frequently but nevertheless at the same time intervals (*see Section 7.1*). This achieves a more uniform load on the processor.

The controller calls entered in the shared data block with the number DB_NBR specify the order and how often the controllers must be processed (Figure 3-10). For detailed information about assigning parameters to LP_SCHED, refer to *Section 7.1* of this manual.

You assign parameters using STEP 7. Parameters cannot be assigned for LP_SCHED using the configuration tool.

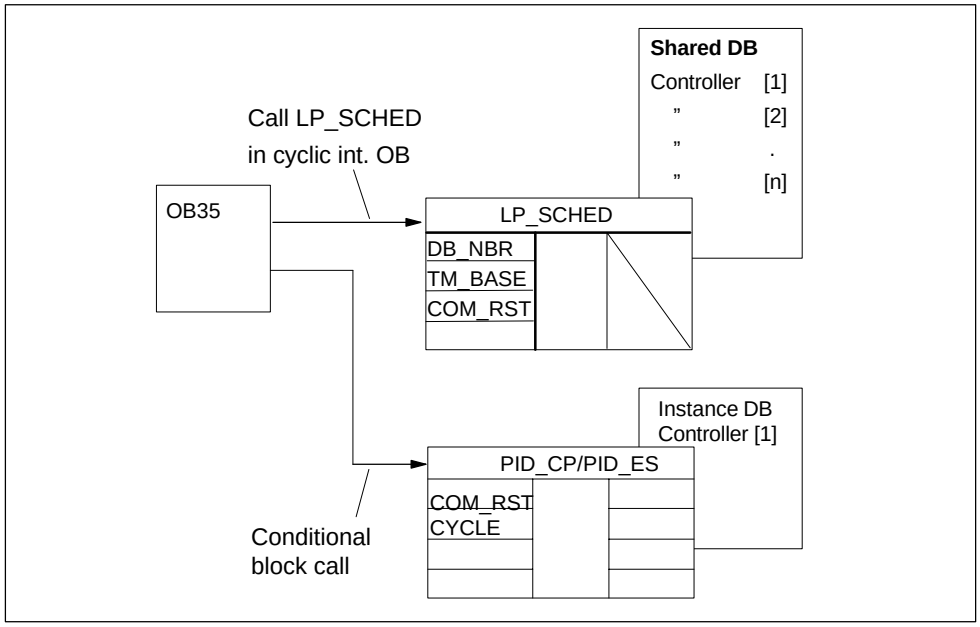


Figure 3-10 Calling a Controller with the Loop Scheduler LP_SCHED

3.6 Range of Values and Signal Adaptation (Normalization)

Internal Numerical Representation

When the algorithms in the function blocks of the Standard PID Control are processed, the processor works with numbers in the floating point format (REAL). The floating point numbers have the single format complying with ANSI/IEEE standard 754-1985:

Format:	DD (32 bits)
Range of values:	$-3.37 * 10^{38}$ ... $-8.43 * 10^{-37}$ and
	$8.43 * 10^{-37}$ ... $3.37 * 10^{38}$

This range is the total range of values for parameters in the REAL format. To avoid limits being exceeded during processing, the input signal SP_EXT which is an analog physical value is defined as a technical range of values:

Techn. Range of values: -10^5 ... $+10^5$

Time values are implemented and processed in the TIME format. A time value is a 32 bit long BCD number in which the four most significant bits are reserved for specifying the time base.

Format:	DD (32 bits)
Range of values:	0 ... +9 999 999 sec
Time base:	10 ms, 100 ms, 1 sec, 10 sec

Signal Adaptation

The normalization function at the input for the external setpoint allows any characteristic curve of transmitters or sensors to be adapted to the physical range of values of the Standard PID Control.

Signal Processing in the Setpoint/Process Variable Channels and PID Controller Functions

4

4.1 Signal Processing in the Setpoint Branch

4.1.1 Setpoint Generator (SP_GEN)

Application

Using a higher/lower switch, you can adjust the internal setpoint. The selected value can be monitored at MP1.

The SP_GEN Function

The SP_GEN function generates a setpoint that can be set or modified using switches. The output variable outv can be increased or decreased step-by-step via the binary inputs SPUP and SPDN.

The range of the setpoint is restricted by the high/low limits SP_HLM/SP_LLM in the setpoint branch. The numerical values of the limits (as percentages) are set using the corresponding input parameters. The signal outputs QSP_HLM and QSP_LLM indicate when these limits are exceeded.

To allow small changes to be made, the controller should not have a sampling time of more than 100 ms.

The rate of change of the output variable depends on the length of time the switches SPUP or SPDN are activated and on the selected limits as shown below:

During the first 3 seconds after setting SPUP or SPDN:

$$\frac{d\,outv}{dt} = \frac{SP\_HLM - SP\_LLM}{100\ s}$$

afterwards:

$$\frac{d\,outv}{dt} = \frac{SP\_HLM - SP\_LLM}{10\ s}$$

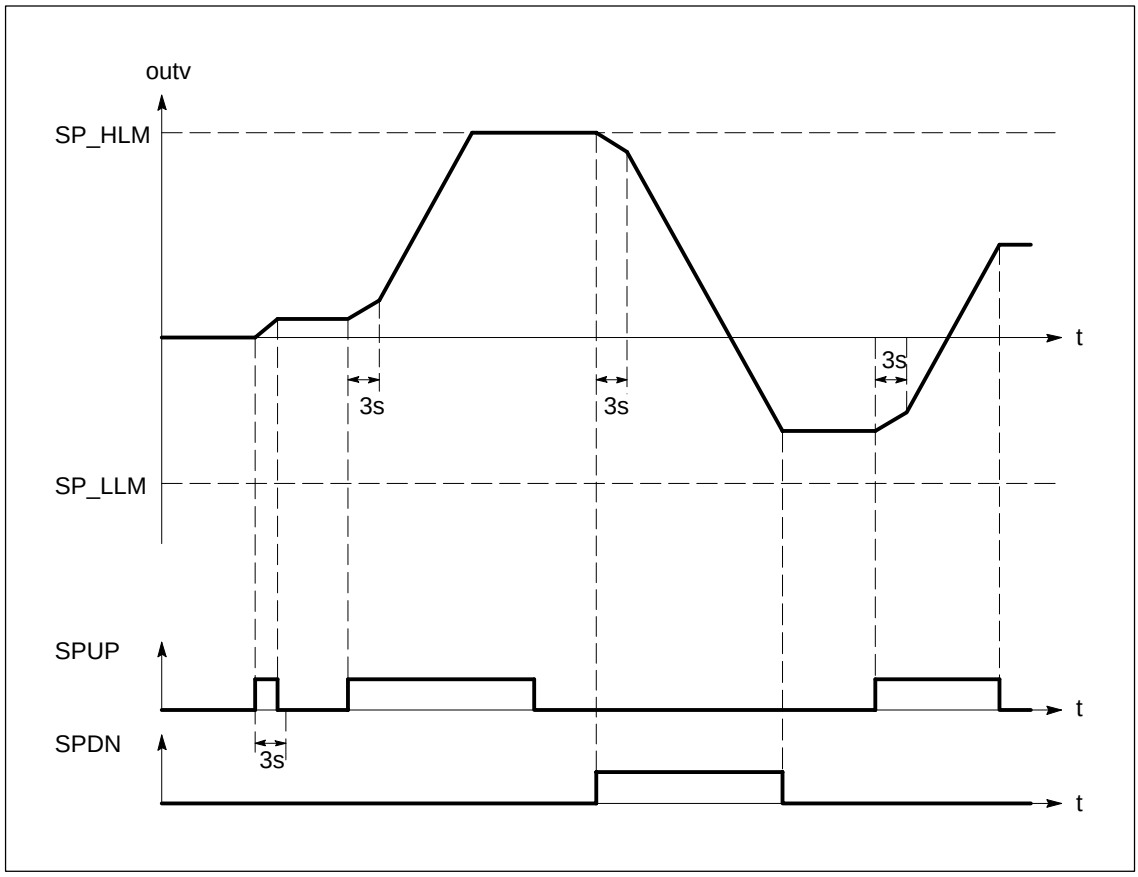


Figure 4-1 Changing outv as a function of the switches SPUP and SPDN

At a sampling time of 100 ms and a setpoint range of -100.0 to 100.0, the setpoint changes by 0.2 per cycle during the first three seconds. If SPUP is activated for longer, the rate of change then changes to a ten fold value, in this case 2 per cycle (Figure 4-1).

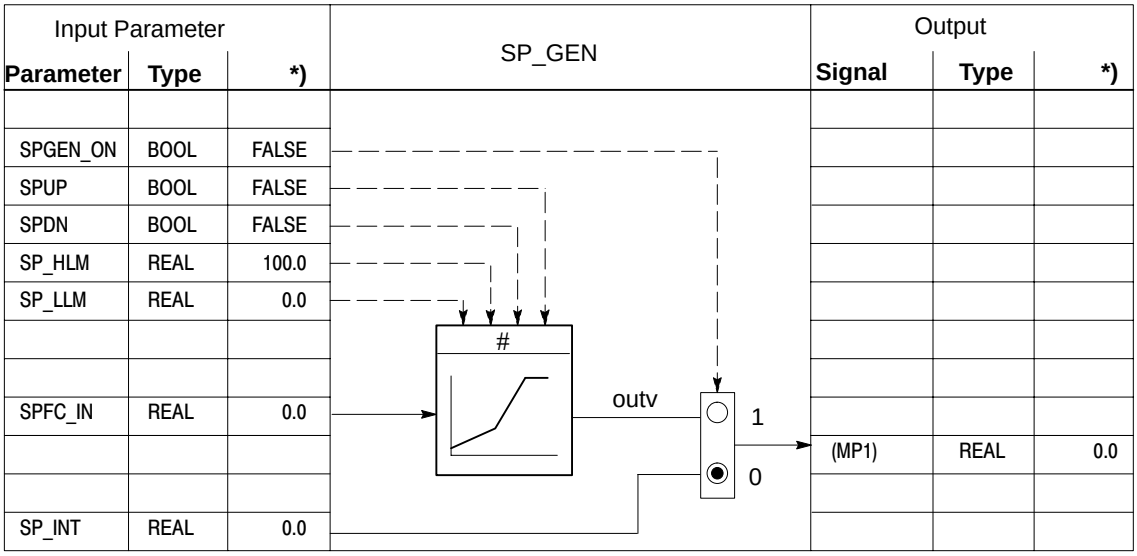
Start Up and Mode of Operation of the Setpoint Generator

- During a complete restart, the outv output is reset to 0.0.
- Switch on the setpoint generator (SPGEN_ON=TRUE), at output outv, the signal value SPFC_IN is then output. The transition to the setpoint generator from a different mode is therefore always smooth. As long as the SPUP and SPDN switches (up/down keys) are not activated, SPFC_IN is applied to the output.

Parameters of the SP_GEN Functions

The outv output parameter is an implicit parameter. It can be monitored using the configuration tool at measuring point MP1.

Parameter	Meaning	Permitted Values
SPFC_IN	Setpoint FC input	Technical range of values
SP_INT	Internal setpoint	Technical range of values



*) Default when the instance DB is created

Figure 4-2 Functions and Parameters of the Setpoint Generator

4.1.2 Ramp Soak (RMP_SOAK)

Application

If you want the setpoint SP_INT to be changed automatically over a period of time, for example when controlling processes according to a time-driven temperature program, you can configure a curve and activate the ramp soak RMP_SOAK. The curve is made up of a maximum of 256 coordinates.

The RMP_SOAK Function

The ramp soak RMP_SOAK in the setpoint branch supplies the output variable OUTV (Figure 4-3) according to a defined schedule. This function is started by setting the input bit RMPSK_ON. If the bit for cyclic repetition CYC_ON is set, the function is started again at the first time slice outv[1] after the last time slice outv[NPR_PTS] has been output. There is no interpolation between the last and first time slice when cyclic repetition is on.

The sequence of the ramp soak is defined by specifying a series of time slices (between coordinates) in a shared data block with the time values PI[i].TMV and the corresponding output values PI[i].OUTV. (Figure 4-3).

PI[i].TMV specifies the length of time of the time slices. There is linear interpolation between the coordinates.

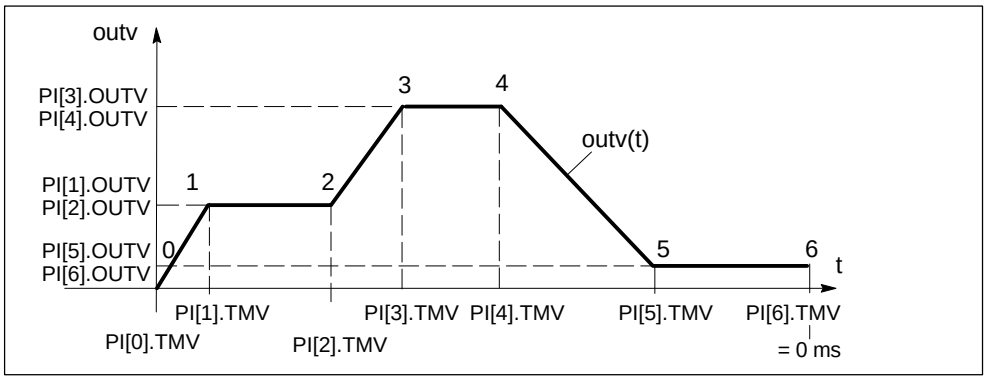


Figure 4-3 Ramp Soak with Start Point and Six Time Slices

Note

With n time slices the time value PI[n].TMV for the last time slice is 0 ms (end of processing). The processing time of a ramp soak is calculated from the initial value down to 0.

Note

During the interpolation of the ramp soak between the time slices, the output value may pause occasionally if the sampling time CYCLE is very small compared with the time between the time between the time slices PI[n].TMV. The ramp soak cannot produce flat linear forms arbitrarily because of the computational accuracy of the CPU. If the ramp soak is too flat, the output value will pause at the respective time slice for a while and then integrates with the minimum gradient to the next time slice.

Remedy: Reduce the time between the time slices by inserting additional time slices. This way you will get the ramp soak output closer to the desired flat ramp soak in a trapeze from.

Using the Ramp Soak

- The time slice parameters NPR_PTS, PI[i].TMV and PI[i].OUTV are located in a shared data block.
- The parameter PI[i].TMV must be specified in the IEC TIME format.
- The way in which the maximum 256 coordinates and time slices are counted is illustrated in the following diagram.

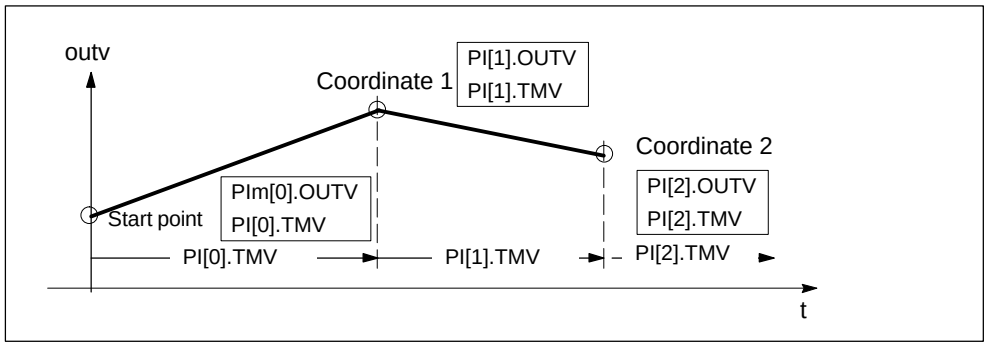


Figure 4-4 Counting the Coordinates and Time Slices

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

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

Configuring the Ramp Soak "configuring"

The number of configured coordinates (NBR_PTS) and the values for the setpoint SP assigned to the individual time slices can be monitored at MP1 and are located in a shared data block with the number DB_NBR (Table 4-2). The output of the ramp soak begins at start point [0] and ends with the coordinate [NBR_PTS].

Modes of the Ramp Soak

By influencing the control inputs, the following ramp soak statuses and operating modes can be implemented:

1. Ramp soak on for a single run.
2. Default value at output of ramp soak (for example SP_INT).
3. Repetition on (cyclic mode).
4. Hold processing of the ramp soak (hold setpoint value).
5. Set the time slice and time to continue (the remaining time RS_TM and the time slice number TM_SNBR are redefined).
6. Update the total processing time and total time remaining.

Modes

This table (Table 4-1) shows the values for the control inputs to set a particular mode:

Table 4-1 Modes of the Ramp Soak (RMP_SOAK)

Mode	RMPSK_ON	DFRMP_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	TRUE	TRUE					SP_INT or output of SP_GEN
3. Repetition on	TRUE	FALSE	FALSE		TRUE		outv(t) When completed: automatic start
4. Hold ramp soak	TRUE	FALSE	TRUE	FALSE			Current value of outv(t) is 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

*) Until the next time slice, the curve is not that set by the user

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

Activating the Ramp Soak

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

During a **complete restart**, the outv output is reset to 0.0 and the total time or total remaining time is calculated. When it changes to normal operation, the ramp soak is processed immediately from the start point according to the selected mode. If you do not require this, the parameter RMPSK_ON in the complete restart OB must be set to FALSE.

Danger

The block does not check whether a shared DB with the number DB_NBR exists or not and whether the parameter 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.

Preassigning the Output, Starting the Traveling Curve

If DFRMP_ON = TRUE, the output value of the ramp soak is set to the signal value SP_INT or the output value of SP_GEN. If DFRMP_ON = FALSE, the curve starts from this point.

Note

The switch DFRMP_ON only has an effect when the ramp soak is activated (RMPSK_ON = TRUE).

The changeover from DFRMP_ON=FALSE is followed by the linear adjustment of outv from the selected setpoint (for example SP_INT) to the output value of the current time slice number PI[NBR_ATMS].OUTV.

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

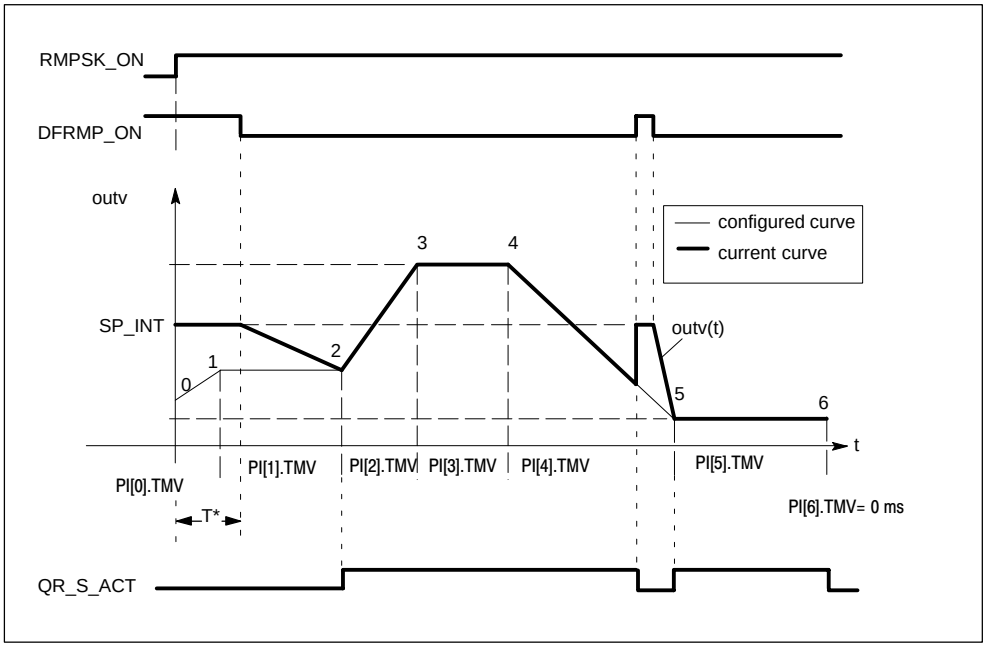


Figure 4-5 Influencing the Ramp Soak with the Default Signal DFRMP_ON

When the ramp soak is started with RMPSK_ON = TRUE, the fixed setpoint SP_INT is output until DFRMP_ON changes from TRUE to FALSE after the time T* (Figure 4-5). At this point, the time PI[0].TMV and part of the time PI[1].TMV has expired. The output value outv is moved from SP_INT to PI[2].OUTV.

The configured curve is only output starting at coordinate 2, where the output signal QR_S_ACT changes to the value TRUE. When the preassigned signal DFRMP_ON changes from FALSE to TRUE while the travel curve is being executed, the output value outv jumps without delay to SP_INT or to the output value of SP_GEN.

Cyclic Mode On

If the cyclic repetition mode is turned on (CYC_ON=TRUE), the ramp soak returns to the start point automatically after outputting the last time slice value and begins a new cycle.

There is no interpolation between the last time slice and the start point. The following must apply to achieve a smooth transition: PI[NBR_PTS].OUTV = PI[0].OUTV.

Hold Setpoint Value

With RMP_HOLD = TRUE, the value of the output variable (including the time processing) is frozen. When this is reset (RMP_HOLD = FALSE), the ramp soak continues at the point of interruption PI[x].TMV.

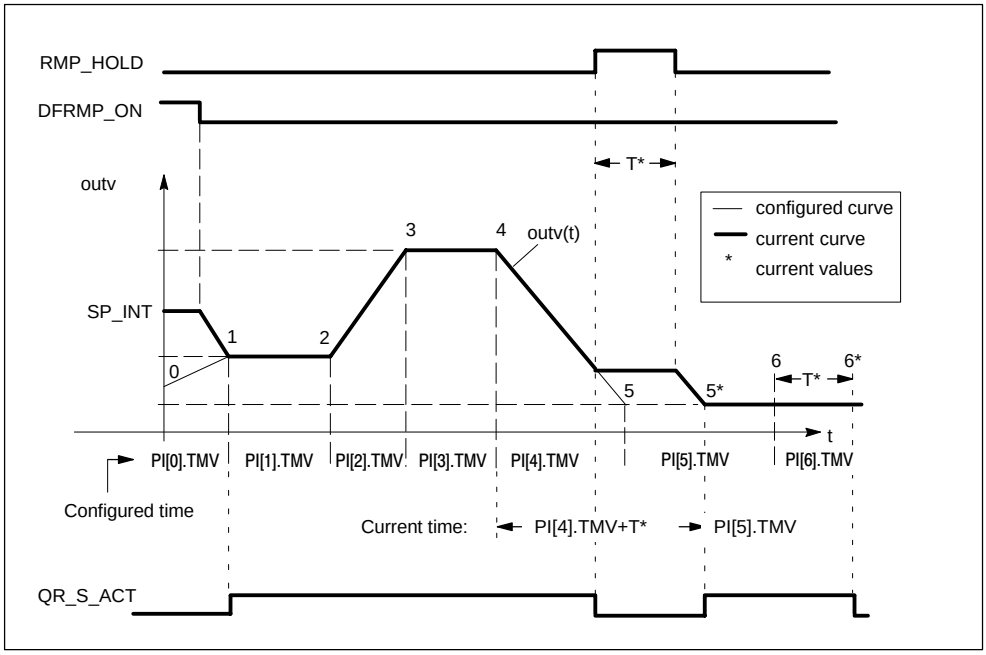


Figure 4-6 The Effect of the Hold Signal RMP_HOLD on the Ramp Soak

The processing time of the ramp soak is increased by the hold time T*. The ramp soak follows the configured curve from the time slice to the signal change for RMP_HOLD (FALSE → TRUE) and from time slice 5* to time slice 6*, in other words the output signal QR_S_ACT has the value TRUE (Figure 4-6).

If the CONT_ON bit is set, the frozen ramp soak continues from the selected point TM_CONT.

Selecting the Time Slice and Time to Continue

If the control input CONT_ON is set to TRUE to continue processing, then processing continues at the time TM_CONT with the time slice TM_SNBR. The time parameter TM_CONT determines the time remaining that the ramp soak requires until it reaches the destination time slice TM_SNBR.

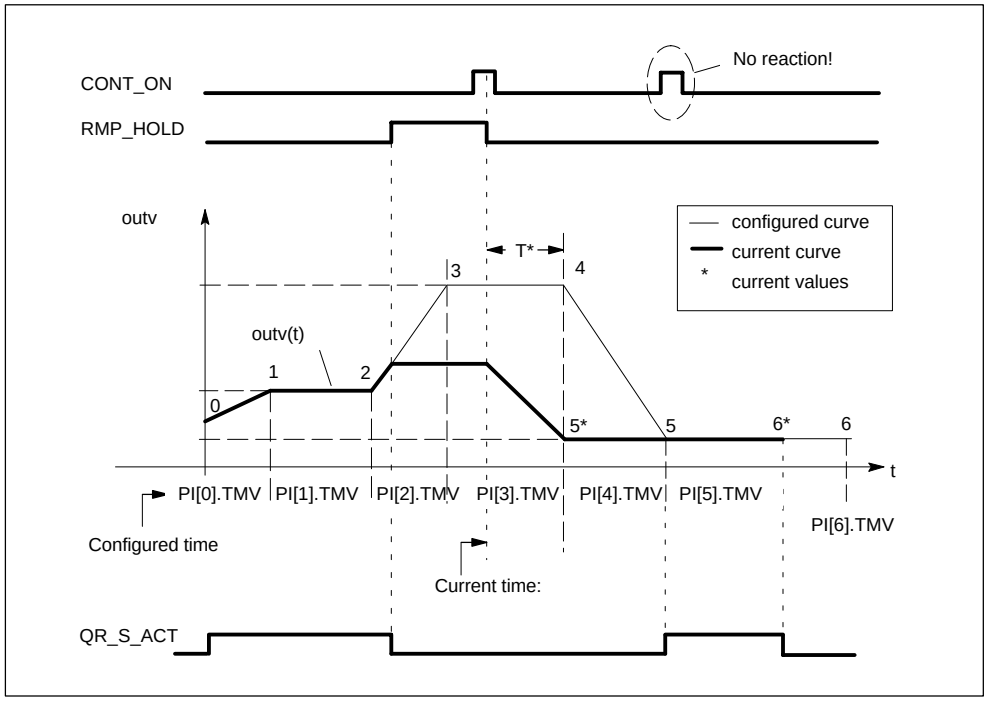


Figure 4-7 How the RMP_HOLD Hold Signal and the CONT_ON Continue Signal Affect the Ramp Soak

The following applies to the example (Figure 4-7): If RMP_HOLD = TRUE and CONT_ON = TRUE and if the following is selected

time slice number to continue	TM_SNBR = 5
and time remaining to selected time slice	TM_CONT = T*

then the configured coordinates 3 and 4 are omitted in the processing cycle of the ramp soak. After a signal change at RMP_HOLD from TRUE to FALSE the curve only returns to the configured curve starting at coordinate 5.

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

Updating the Total Time and Total Time Remaining

In every cycle, the current time slice number NBR_ATMS, the current time remaining until the time slice RS_TM is reached, the total time T_TM and the total time remaining until the end of the ramp soak RT_TM is reached are updated.

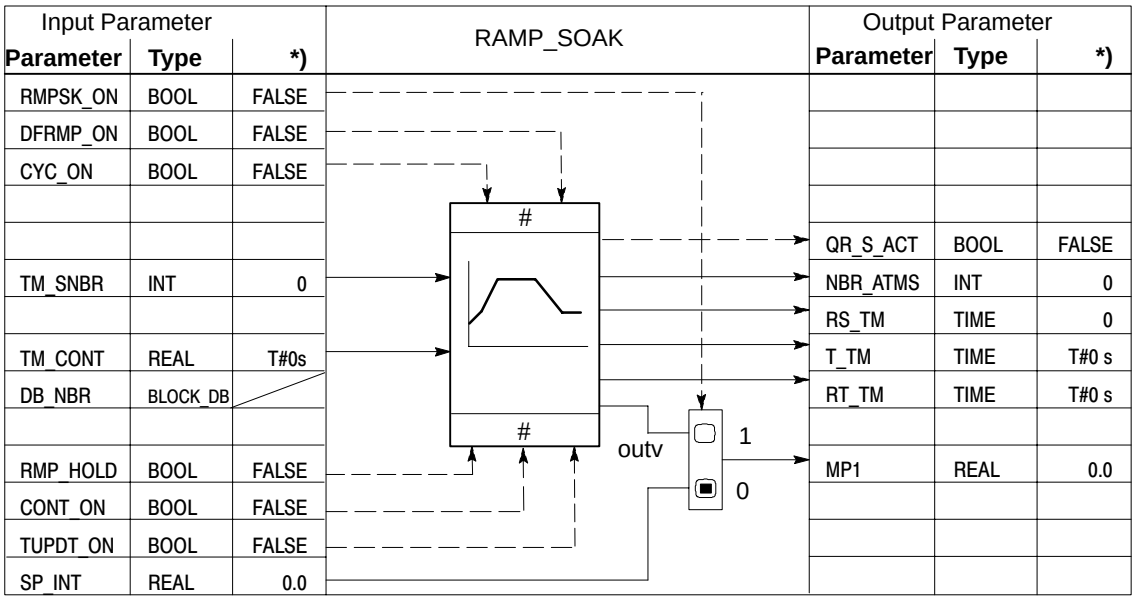
If there are on-line changes to PI[n].TMV, the total time and the total time remaining are changed. Since the calculation of T_TM and RT_TM greatly increases the run time of the function block if there are a lot of time slices, the calculation is only performed after a complete restart or when TUPDT_ON = TRUE. The time slices PI[0toNBR_PTS].TMV between the individual coordinates are totalled and indicated at the output for the total time T_TM and for the total remaining time RT_TM.

Please remember that the calculation of the total times requires a relatively large amount of CPU time.

Parameters of the RMP_SOAK Function

The output parameter outv is an implicit parameter and is accessible at the configuration tool via the measuring point MP1 (see Figure 2-12).

Parameter	Meaning	Permitted Values
TM_SNBR	Number of the next time slice	> 0 (no dimension)
TM_CONT	Time to continue	Entire range of values
SP_INT	Internal setpoint	Technical range of values



*) Default when the instance DB is created

Figure 4-8 Functions and Parameters of the Ramp Soak

The time slice coordinates and the number of time slices NBR_PTS are stored in a shared data block (Table 4-2).

Table 4-2 Shared Data Block (DB_NBR), with Default of Start Point and Four Time Slices

Parameter	Data Type	Comment	Permitted range of values	Default
NBR_PTS	INT	Number of coordinates	1 to 256	4
PI[0].OUTV	REAL	Output value [0]: start point	Entire range of values	0.0
PI[0].TMV	TIME	Time value [0]: start point	Entire range of values	T#1 s
PI[1].OUTV	REAL	Output value [1]: coordinate 1	Entire range of values	0.0
PI[1].TMV	TIME	Time value [1]: coordinate 1	Entire range of values	T#1 s
PI[2].OUTV	REAL	Output value [2]: coordinate 2	Entire range of values	0.0
PI[2].TMV	TIME	Time value [2]: coordinate 2	Entire range of values	T#1 s
PI[3].OUTV	REAL	Output value [3]: coordinate 3	Entire range of values	0.0
PI[3].TMV	TIME	Time value [3]: coordinate 3	Entire range of values	T#1 s
PI[4].OUTV	REAL	Output value [4]: coordinate 4	Entire range of values	0.0
PI[4].TMV	TIME	Time value [4]: coordinate 4	Entire range of values	T#0 s

4.1.3 Normalization of the External Setpoint (SP_NORM)

Application

If the external setpoint valueis not available in the physical unit of the process variable (for example as a % in case of a controller cascade), this value and its setting range have to be normalized to the physical unit of the process variable. This is carried out through the function “Normalization in the setpoint branch”.

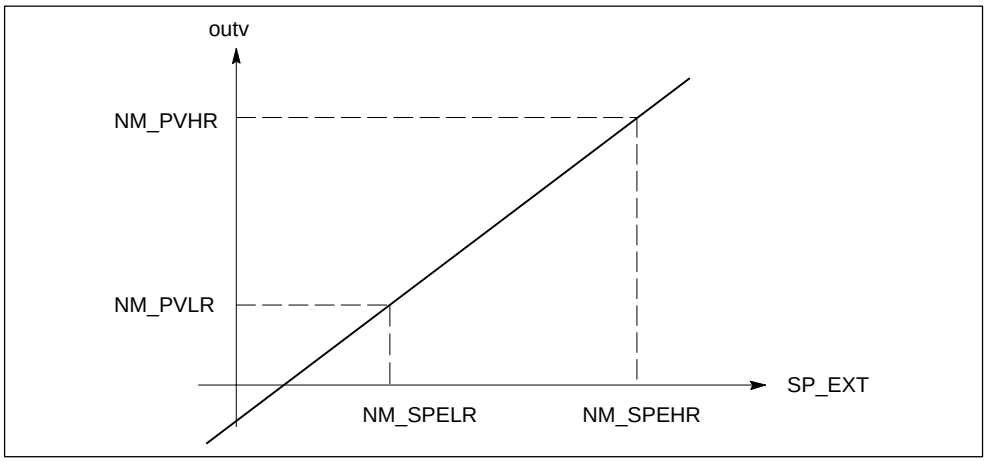
The SP_NORM Function

The SP_NORM function normalizes an analog input value. The analog external setpoint is transferred to the outv output variable using the normalization curve (straight line). The output value OUTV is accessible with the configuration tool at measuring point MP2 (Figure 2-12).

The output value of the function is effective when the control input SPEXT_ON = TRUE.

To specify the straight line normalization curve the following parameters must be defined:

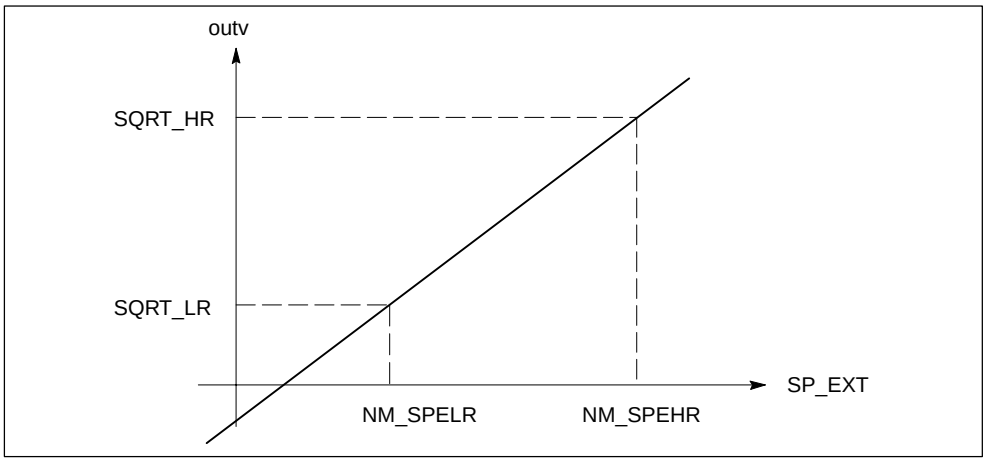
- The upper limit of the input value SP_EXT: NM_SPEHR
- The lower limit of the input value SP_EXT: NM_SPELR
- The upper limit of the output value outv: NM_PVHR (this value is specified in the normalization function of the process variable.)
- The lower limit of the output value outv: NM_PVLR (this value is specified in the normalization function of the process variable.)



The output value outv is calculated from the respective input value SP_EXT in accordance with the following formula:

$$outv = (SP\_EXT - NM\_SPELR) \times (NM\_PVHR - NM\_PVLR) / (NM\_SPEHR - NM\_SPELR) + NM\_PVLR$$

In the special case of an activated square-root function in the process-variable branch, the normalization values of the square-root function (SQRT_HR und SQRT_LR) are used as the upper and lower limits of the output value.



In this case the output value of the normalization function is calculated from the input value SP_EXT in accordance with the following formula:

outv = (SP_EXT – NM_SPELR) x (SQRT_HR – SQRT_LR) / (NM_SPEHR – NM_SPELR) + SQRT_LR

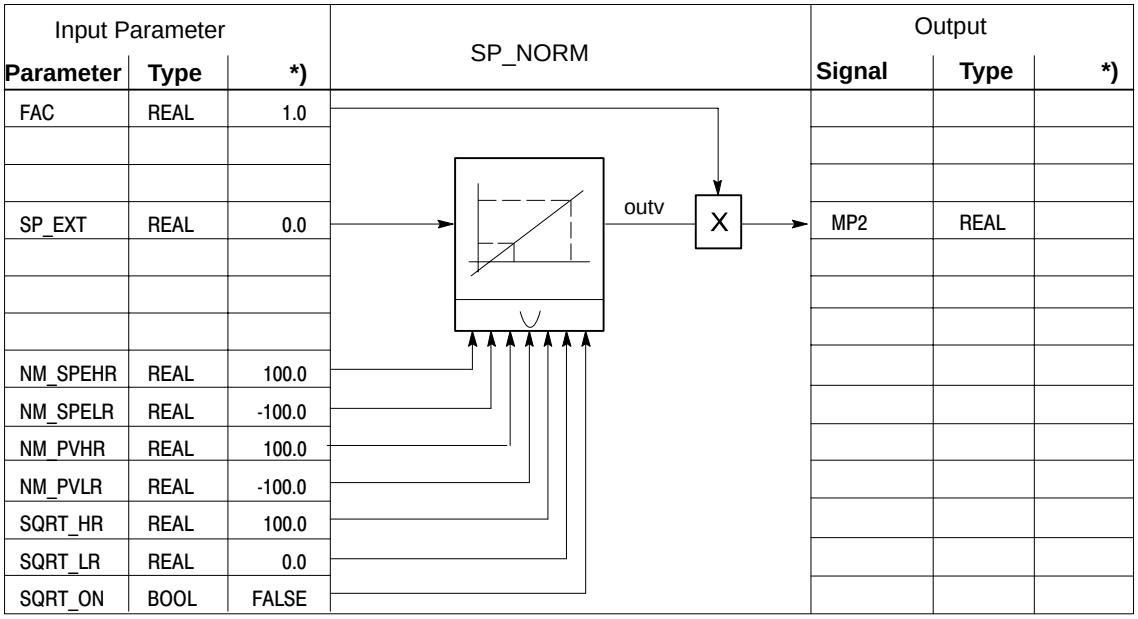
Internally, the function does not limit any values and the parameters are not checked. If you enter the same value for NM_SPEHR and NM_SPELR, division by zero can occur in the above formula. The function block does not rectify this fault!

Parameters of the SP_NORM Function

The output parameter outv is an implicit parameter and is accessible with the configuration tool only at measuring point MP2.

Parameter	Meaning	Permitted range of values
SP_EXT	External setpoint	Technical range of values (physical value)
NM_SPEHR	Upper limit of the input value SP_EXT	Technical range of values (physical unit of SP_EXT)
NM_SPELR	Lower limit of the input value SP_EXT	Technical range of values (physical unit of SP_EXT)
NM_PVHR	Upper limit of the output value outv	Technical range of values (physical unit of the process variable or no dimension, if the square-root function is activated)
NM_PVLR	Lower limit of the output value outv	Technical range of values (physical unit of the process variable or no dimension, if the square-root function is activated)
SQRT_HR	Upper limit of the output value outv, if a square-root function is activated in the process variable branch	Technical range of values

Parameter	Meaning	Permitted range of values
SQRT_LR	Lower limit of the output value outv, if a square-root function is activated in the process variable branch	Technical range of values
SQRT_ON	Activate square-root function	TRUE, FALSE



*) Default when the instance DB is created

Figure 4-9 Functions and Parameters for Normalizing the External Setpoint

4.1.4 FC Call in the Setpoint Branch (SPFC)

Application

By inserting a user-specific FC block in the setpoint branch it is possible to process a setpoint set externally before it is connected to the controller (for example a signal delay or linearization) (Figure 2-12).

The SPFC Function

If you activate the SPFC function with SPFC_ON = TRUE, a user-defined FC block is called. The number of the FC block is entered using the SPFC_NBR parameter.

The controller calls the user FC. Input/otput parameterts of the user FC are not supplied with values. You must must therefore program the data transfer using S7 STL in the user FC. A programming example is shown below.

STL	Explanation

```
FUNCTION “User FC”
VAR_TEMP
INV:REAL;
OUTV:REAL;
END_VAR
BEGIN
L      “Controller DB”.SPFC_IN
T      #INV
       //User function OUTV=f(INV)
L      #OUTV
T      “Controller DB”.SPFC_OUT
END_FUNCTION
```

The value of SPFC_ON then determines whether a user-programmed function in the form of a standard FC (for example a characteristic curve) is inserted at this point in the setpoint channel or whether the setpoint is processed further without any such influence.

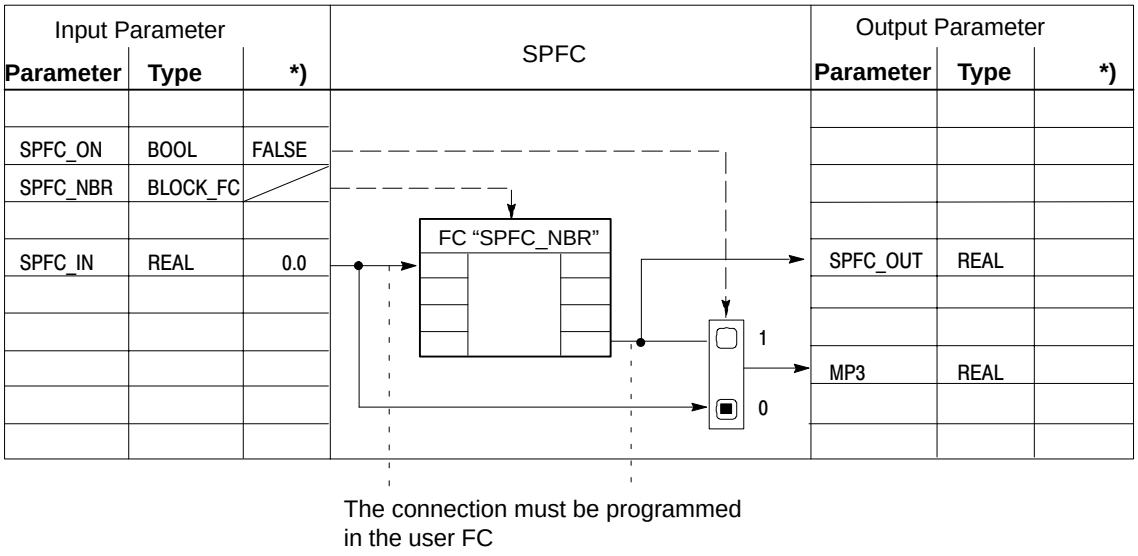
Caution

The block does not check whether an FC exists. If the FC does not exist, the CPU changes to STOP with an internal system error.

Parameters of the SPFC Function

The input value SPFC_IN is an implicit parameter. This can be monitored using the configuration tool either at measuring point MP1 (setpoint = SP_INT) or at measuring point MP2 (setpoint = SP_EXT). The output value is accessible at measuring point MP3.

The SPFC_IN input is switched through to the setpoint branch when SPFC_ON = FALSE is set (default).



*) Default when the instance DB is created

Figure 4-10 Calling an FC Block in the Manipulated Variable Branch

4.1.5 Limiting the Rate of Change of the Setpoint (SP_ROC)

Application

Ramp functions are used in the setpoint branch when step-shaped changes in the actuating signal are not acceptable for the process since a step change in the setpoint normally means a step change in the manipulated variable of the controller. Such abrupt changes in the manipulated variable must, for example, be avoided when there is gearing between a motor and the load and when a fast increase in the speed of the motor would overload the gear unit.

The SP_ROC Function

The SP_ROC function limits the rate of change of the setpoints processed in the controller separately for the rate of change up and rate of change down and also separately for the positive and negative ranges.

The limits for the rate of change of the ramp function in the positive and negative range of the reference variable are entered at the four inputs SPURLM_P, SPDRLM_P, SPURLM_N and SPDRLM_N. The rate of change is an up or down rate per second. Faster rates of change in the setpoint are delayed by these limits.

If, for example, **SPURLM_P** is configured to 10.0 [technical range of values/s], the following values are added to the 'old value' of outv in each sampling cycle as long as inv > outv:

Sample time	1 s	$\rightarrow outv_{old} + 10$
	100 ms	$\rightarrow outv_{old} + 1$
	10 ms	$\rightarrow outv_{old} + 0.1$

How signals are handled by the function is illustrated by the following figure based on an example. Step functions at the input inv(t) become ramp functions at output outv(t).

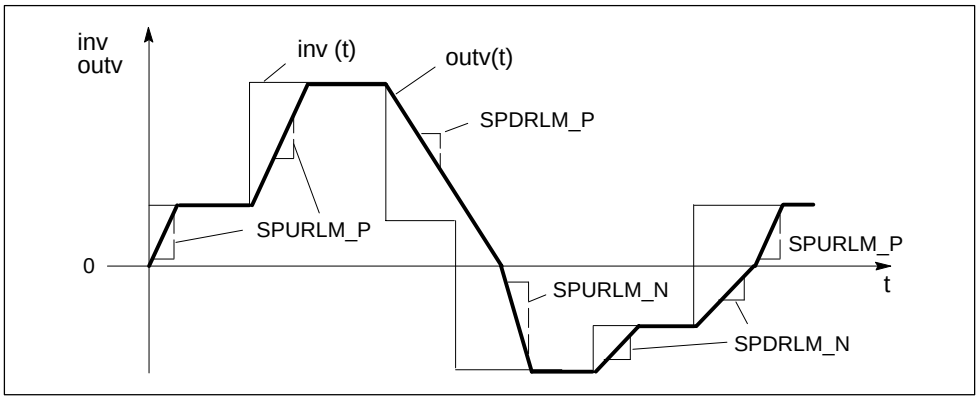


Figure 4-11 Limiting the Rate of Change of the Setpoint SP(t)

No signal is output when the rate of change limits are reached.

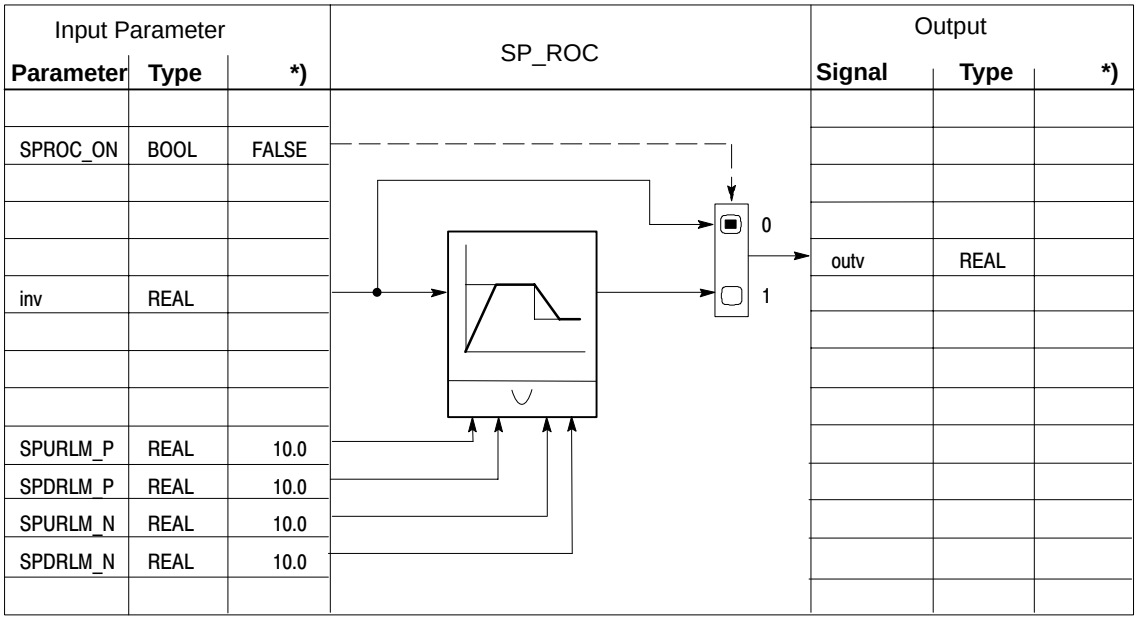
Parameters of the SP_ROC Function

The inv input value is an implicit parameter and is accessible to the configuration tool only at measuring point MP3 (Figure 2-12).

The output value outv is not accessible at the configuration tool (see Figure 2-12).

The rates of change (per second) are always entered as a positive value.

Parameter	Ramp	Meaning	Permitted range of values
SPURLM_P	OUTV > 0 and |OUTV| rising	SP rise limit in the positive range	≥ 0 [technical range of values/s]
SPDRLM_P	OUTV > 0 and |OUTV| falling	SP fall limit in the positive range	≥ 0 [technical range of values/s]
SPURLM_N	OUTV < 0 and |OUTV| rising	SP rise limit in the neg. range	≥ 0 [technical range of values/s]
SPDRLM_N	OUTV < 0 and |OUTV| falling	SP fall limit in the neg. range	≥ 0 [technical range of values/s]



*) Default when the instance DB is created

Figure 4-12 Functions and Parameters for Limiting the Rate of Setpoint Change

4.1.6 Limiting the Absolute Value of the Setpoint (SP_LIMIT)

Application

The range of values of the setpoint determines the range within which the process variable can fluctuate, in other words, the range of values permitted for the process.

In order to avoid critical or illegal process states, the setting range of the reference variable has upper and lower limits in the setpoint branch by the Standard PID Control.

The SP_LIMIT Function

The SP_LIMIT function limits the setpoint SP to the selectable upper and lower limits SP_LLM and SP_HLM as long as the input value INV is outside these limits. Since the function cannot be disabled, an upper and lower limit must always be assigned during the configuration.

The numerical values of the limits are set at the input parameters for the upper and lower limits. If the input value inv(t) exceeds these limits, this is indicated at the corresponding signal outputs (Figure 4-14).

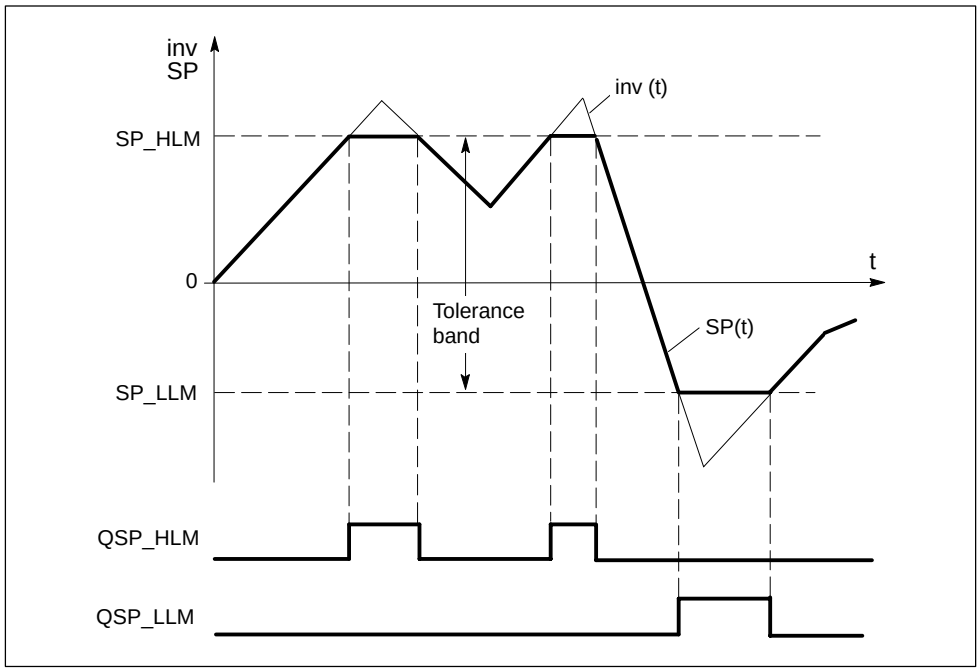


Figure 4-13 Limits for the Absolute Values of the Setpoint SP (t)

Start Up and Mode of Operation

- In case of a complete restart all the signal outputs are set to zero.
- The limitation operates as shown in the following table:

SP =	QSP_HLM =	QSP_LLM =	when:
SP_HLM	TRUE	FALSE	inv $\geq$ SP_HLM
SP_LLM	FALSE	TRUE	inv $\leq$ SP_LLM
INV	FALSE	FALSE	SP_HLM < inv < SP_LLM

The effective setpoint of the Standard PID Control is indicated at the output, i.e. at the parameter SP.

Parameters of the the SP_LIMIT Function

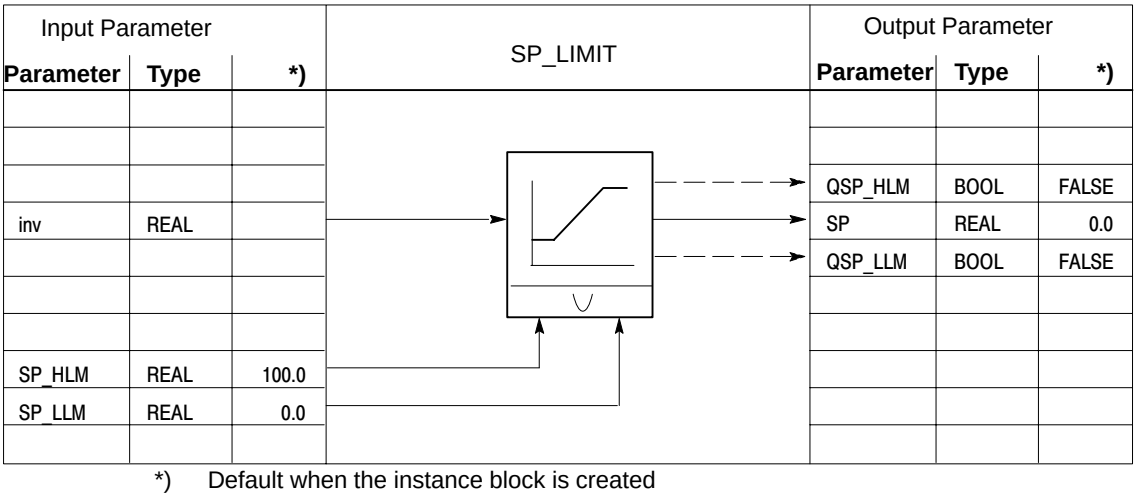
The inv input value is an implicit parameter and can only be monitored with the configuration tool at measuring point MP3.

For the limitation function to operate properly, the following must apply:

SP_HLM > SP_LLM

Parameter	Meaning	Permitted Values
SP_HLM	Upper limit of the setpoint	SP_LLM ... Upper limit of the technical range of values
SP_LLM	Lower limit of the setpoint	Lower limit of the technical range of values ... SP_HLM

The input parameter inv is an implicit input parameter and is not accessible at the configuration tool (see Figure 2-12).



*) Default when the instance block is created

Figure 4-14 Functions and Parameters of the Absolute Value Limits of the Setpoint

4.1.7 Setpoint Adjustment Using the Configuration Tool

SP Display and Setting in the Loop Monitor

The configuration tool has its own interface to the controller FB. It is therefore possible at any time to interrupt the setpoint branch and to specify your own setpoint SP_OP, for example for test purposes when working on a programming device/personal computer on which the configuration tool is loaded (Figure 4-15).

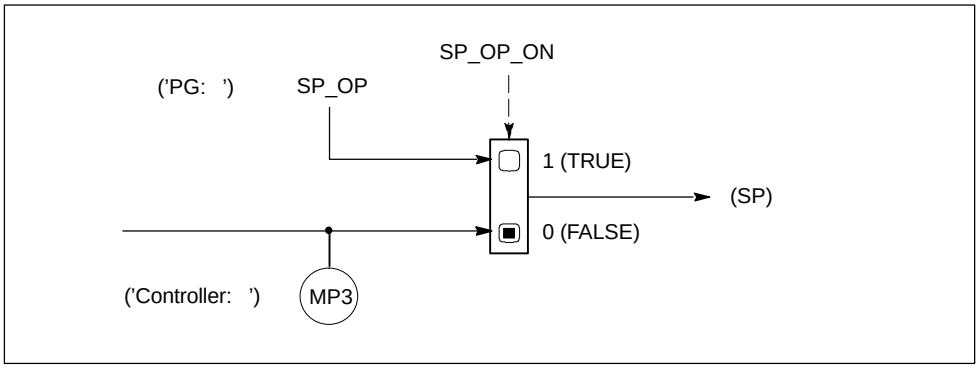


Figure 4-15 Intervention in the Setpoint Branch Using the Configuration Tool

One of the three fields (labeled **setpoint**) in the **loop monitor** is available for this. Here the setpoint currently applied to measuring point MP3 is displayed below ("Controller:"). The field above this (PG:) is used to display and change the parameter SP_OP.

Changeover to the Setpoint Specification by the Configuration Tool

If the switch in the configuration tool is set to 'PG: ', the switching signal of the structure switch SPOP_ON is set to TRUE and SP_OP is enabled to the setpoint SP value in the controller FB.

If the rate of change limitation SP_ROC is activated in the setpoint branch, you can switch over between the "PG" ' and 'Controller: ' settings without a sudden change occurring in the setpoint. The value adopted with the changeover (MP3) can be viewed in the "Controller: " display field of the **loop monitor**. The SP then approaches this value using the ramp set at SP_ROC.

These interventions, however, only affect the process when you transfer them to the programmable logic controller by clicking the "Send" button in the **loop monitor**.

4.2 Signal Processing in the Process Variable Branch

4.2.1 Normalizing the Process Variable Input

Application

The ”Normalization in the process variable” function is used to normalize the input value PV_PER or PV_IN to the physical unit of the process variable.

The PV_NORM Function

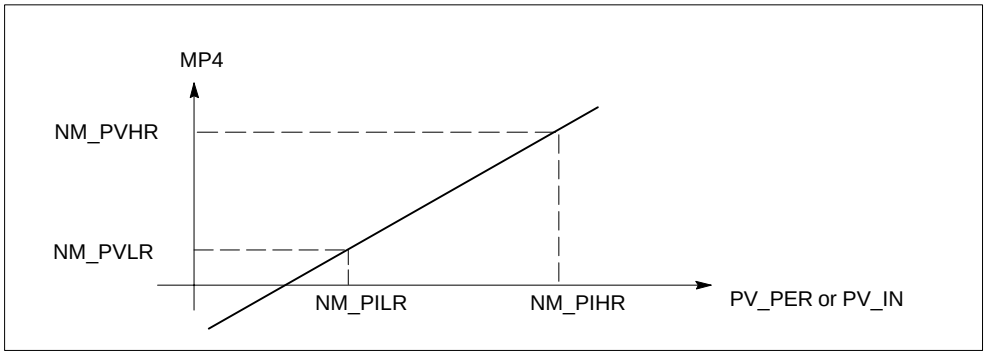
The PV_NORM function normalizes an analog input value. The switch PVPER_ON is used to determine the input variable to be normalized:

- PVPER_ON = TRUE: Input variable is the process variable I/O PV_PER
- PVPER_ON = FALSE: Input variable is the internal process variable PV_IN

The input variable is transferred to the output variable MP4 by using the normalization curve (straight line). The measuring point MP4 is accessible at the configuration tool (see Figure 2-13).

To specify the straight line normalization curve the following four parameters must be defined:

- The upper limit of the input value PV_PER or PV_IN: NM_PIHR
- The lower limit of the input value PV_PER or PV_IN: NM_PILR
- The upper limit of the output value MP4: NM_PVHR
- The lower limit of the output value MP4: NM_PVLR



The output value MP4 is calculated from the respective input value PV_PER or PV_IN in accordance with the following formula:

MP4 = (**PV_PER** – NM_PILR) x (NM_PVHR – NM_PVLR) / (NM_PIHR – NM_PILR) + NM_PVLR

or

MP4 = (**PV_IN** – NM_PILR) x (NM_PVHR – NM_PVLR) / (NM_PIHR – NM_PILR) + NM_PVLR

Internally, the function does not limit any values and the parameters are not checked. If you enter the same value for NM_PIHR and NM_PILR, division by zero can occur in the above formula. The function block does not rectify this fault!

Normalization of the I/O Process Variable

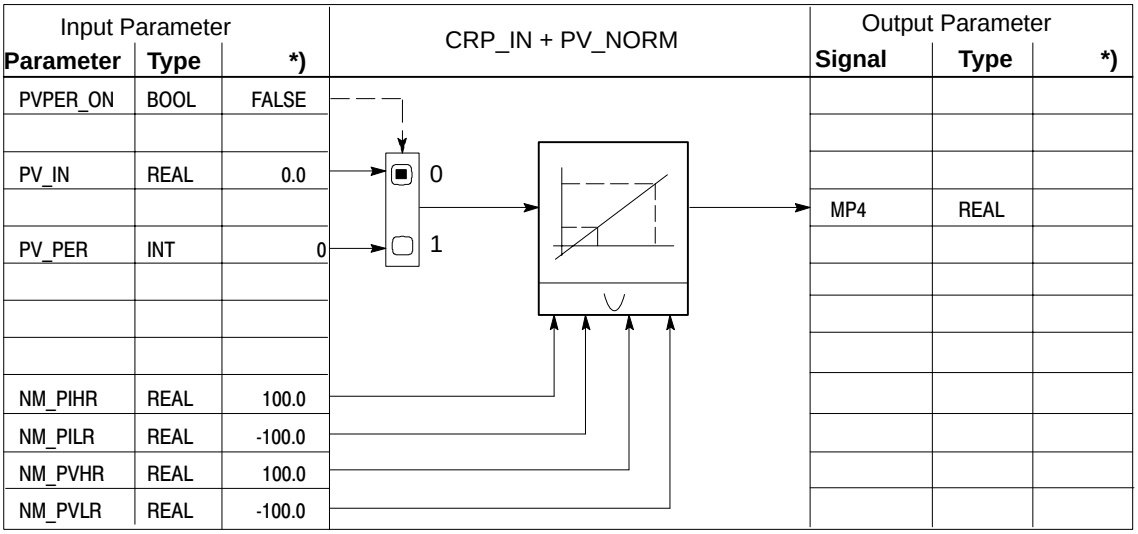
Input of the upper and lower limit of the input value is supported by the configuration user interface.

The rated value upper limit for voltage, current and resistance measuring ranges of the parameter PV_PER (I/O input) ialways lies at decimal 27648, the rated value lower limit to 0 or -27648. With temperature modules, the nominal range upper limit is variable. It is specified in the respective module description.

Parameters of the CRP_IN and PV_NORM Functions

The PV_PER peripheral input is switched to the process variable branch when PVPER_ON = TRUE is set. The normalized peripheral process variable can be monitored at measuring point MP4 (Figure 2-13).

Parameter	Meaning	Permitted Values
PV_PER	Process variable in the peripheral format	
NM_PIHR	Upper limit of the input value	Technical range of values
NM_PILR	Lower limit of the input value	Technical range of values
NM_PVHR	Upper limit of the output value MP4:	Technical range of values (physical unit of the process variable or no dimension, if the square-root function is activated)
NM_PVLR	Lower limit of the output value MP4:	Technical range of values (physical unit of the process variable or no dimension, if the square-root function is activated)



*) Default when the instance DB is created

Figure 4-16 Functions and Parameters for Normalizing Physical Process Variables

4.2.2 Damping the Process Variable (LAG1ST)

Application

The LAG1ST function is used as a delay element for the process variable. This can be used to suppress disturbances.

The LAG1ST Function

By incorporating a time delay, higher frequency fluctuations in the process variable signal can be damped so that they are excluded from the processing in the control algorithm in particular to avoid affecting the derivative action. The amount of signal damping is determined by the time constant PV_TMLAG.

The damping effect is achieved by a first order time lag algorithm.

The transfer function in the Laplace transform is as follows:

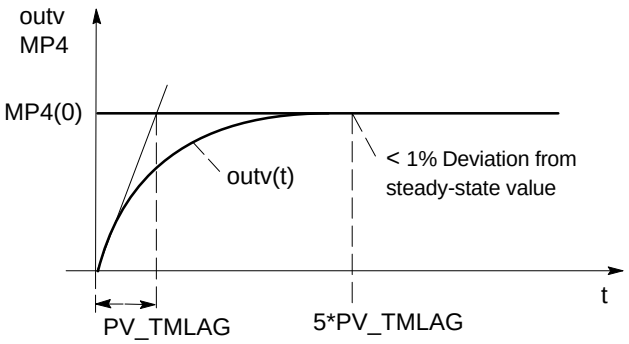
$$\frac{\text{outv(s)}}{\text{MP4(s)}} = \frac{1}{(1 + \text{PV\_TMLAG} * s)}$$ where s = Laplace variable

The step response in the time domain is as follows:

$$\text{outv(t)} = \text{MP4(0)}\ (1 - e^{-t/\text{PV\_TMLAG}})$$

Legend:

MP4(0)	the size of the process variable jump at the input
outv(t)	the output value
PV_TMLAG	the delay time constant
t	time



Conditions for Parameter Assignment

If PV_TMLAG ≤0.5 * CYCLE, there is no lag in effect.

A sampling time (CYCLE) of less than a fifth of the time lag is necessary to achieve a time lag approaching the analog response.

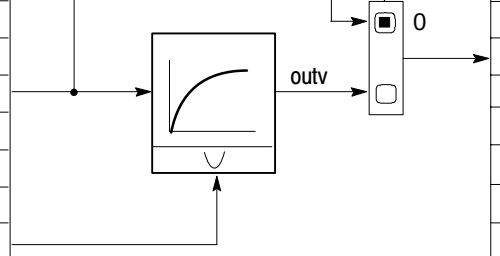
Parameters of the LAG1ST Function

The outv output value is an implicit parameter and can only be monitored with the configuration tool at measuring point MP5 (Figure 2-13).

If LAG1STON = FALSE, the peripheral input PV_PER or the internal input PV_IN is switched to the process variable branch without a time lag (default).

Parameter	Meaning	Permitted Values
PV_TMLAG	Process variable time lag	Entire range of values

Input Parameter			LAG1ST	Output Parameter		
Parameter	**Type**	***)**		**Signal**	**Type**	***)**
LAG1STON	BOOL	FALSE		MP5	REAL	0.0
MP4	REAL	0.0				
PV_TMLAG	TIME	T#5s				



*) Default when the instance block is created

Figure 4-17 Smoothing the Process Variable

4.2.3 Extracting the Square Root (SQRT)

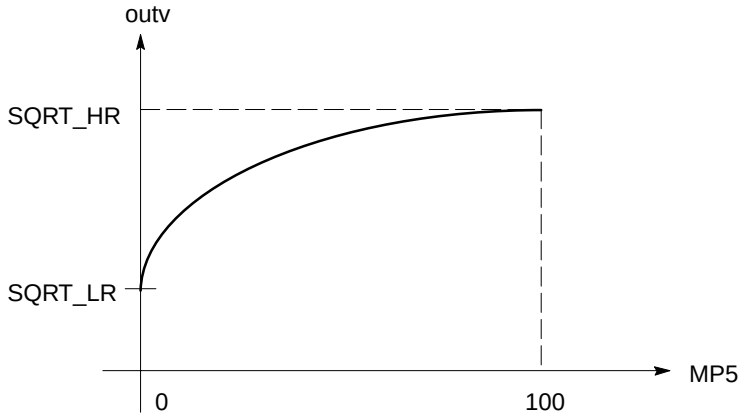
Application

If the process variable supplied by a sensor is a physical value that is in a quadratic relationship to the measured process variable, the changes in the process variable must first be linearized before they can be processed further in the controller. This task is performed by the SQRT function in the process variable branch of the Standard PID Control. The measured signal must always be linearized by extracting the square root when flow measurements are performed with orifice plates or venturi tubes. The measured differential pressure (effective pressure) is then proportional to the square of the flow.

If the SQRT_ON input signal is set to TRUE, the square root function is activated in the process variable branch. The algorithm for the square root function is as follows:

outv = SQRT (MP5) x (SQRT_HR – SQRT_LR) / 10.0 + SQRT_LR

This formula requires that the input value of the square-root be normalized to the numerical range of 0 .. 100. The parameters NM_PVHR and NM_PVLR of the normalization in the process-variable branch must therefore be configured to 100.0 and 0.0.
The square root from this value results in a numerical range of 0 ... 10. The normalization values SQRT_HR and SQRT_LR are used to normalize this numerical range to the physical measuring range (SQRT_LR to SQRT_HR).



Example of Normalization

Let the input value PV_IN of the controller be the differential pressure in mbar:

Measuring-range beginning NM_PILR	Measuring-range end NM_PIHR	Value example for PV_IN
20.0 mbar	200.0 mbar	150.0 mbar

The normalization function PV_NORM is used to calculate the normalized differential pressure, whereby NM_PVHR = 100.0 and NM_PVLR = 0.0:

MP4 = (PV_IN – NM_PILR) * (NM_PVHR – NM_PVLR) / (NM_PIHR – NM_PILR) + NM_PVLR
= (PV_IN – 20.0 mbar) * (100.0 – 0.0) / (200.0 mbar – 20.0 mbar) + 0.0
= (PV_IN – 20.0 mbar) * 100 / 180.0 mbar

Initial value of MP4	End value of MP4	Value example for MP4
0.0	100.0	72.222

No smoothing is used in this example, so that the following applies: MP5 = MP4.

This results in the following values for the square roots from the normalized differential pressure MP5:

Initial value after the square root	End value after the square root	Value example for SQRT(MP5)
0.0	10.0	8.498

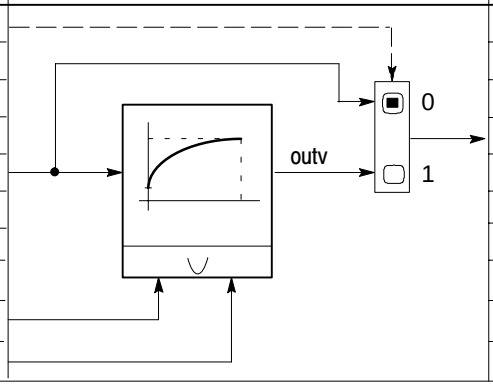
This results in the following equation for the normalized output value outv of the square-root function (physical flow) with SQRT_HR = 20000.0 m^3/h and SQRT_LR = 0.0 m^3/h:

outv = SQRT(MP5) * (SQRT_HR – SQRT_LR) / 10.0 + SQRT_LR
= SQRT(MP5) * (20000.0 m^3/h – 0.0 m^3/h) / 10.0 + 0.0 m^3/h
= 2000.0 m^3/h * SQRT(MP5)

Measuring-range beginning outv	Measuring-range end outv	Value example for outv
0.0 m^3/h	20000.0 m^3/h	16996.732 m^3/h

The output parameter outv is an implicit parameter and is not accessible at the configuration tool (see Figure 2-13).

Input Parameter			SQRT	Output		
Parameter	**Type**	***)**		**Signal**	**Type**	***)**
SQRT_ON	BOOL	FALSE				
				(PVFC_IN)	REAL	0.0
(MP5)	REAL	0.0				
SQRT_HR	REAL	100.0				
SQRT_LR	REAL	0.0				



*) Default when the instance DB is created

Figure 4-18 Functions and Parameters for Extracting the Square Root of the Process Variable Signals

4.2.4 FC Call in the Process Variable Branch (PVFC)

Application

By including a user-defined FC block in the process variable branch, the process variable signal can be pre-processed (for example signal delay or linearization) before further processing in the controller (Figure 2-13).

The PVFC Function

By activating the PVFC function with PVFC_ON = TRUE, a user-specific function (FC) is called. The number of the FC to be used is entered with the PVFC_NBR parameter.

The controller calls the user FC. The existing input/output parameters of the user FC are not supplied. You must therefore program the data transfer using S7 STL in the user FC. A programming example is shown below.

STL	Explanation

```
FUNCTION “User FC”
VAR_TEMP
INV:REAL;
OUTV:REAL;
END_VAR
BEGIN
L      “Controller DB”.PVFC_IN
T      #INV
       //User function OUTV=f(INV)
L      #OUTV
T      “Controller DB”.PVFC_OUT
END_FUNCTION
```

The value of PVFC_ON then determines whether a user-programmed function in the form of a standard FC (for example a characteristic curve) is inserted at this point in the process variable channel or whether the process variable is processed further without any such influence.

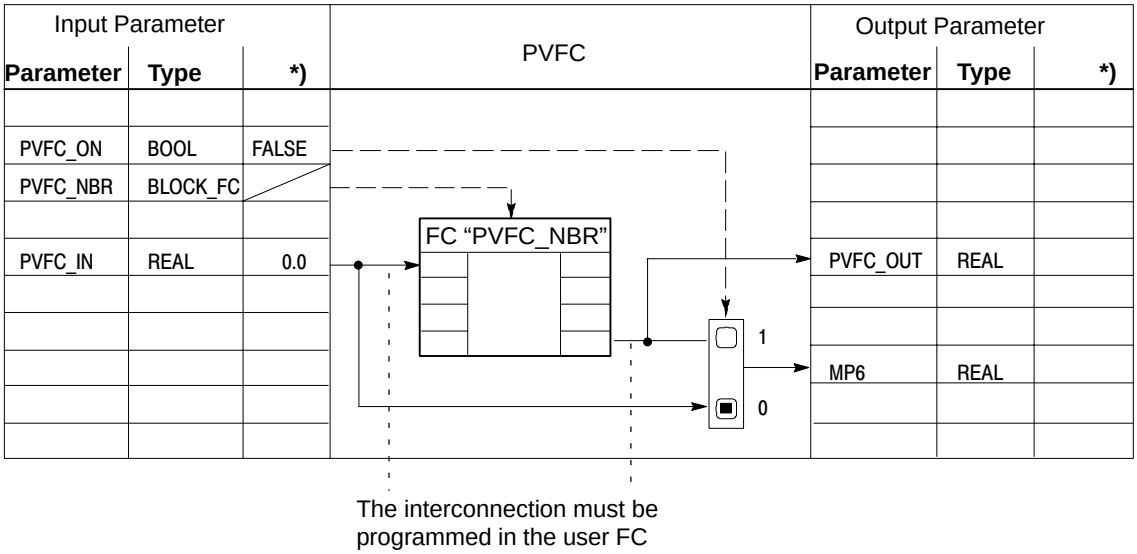
Caution

The block does not check whether an FC exists. If the FC does not exist, the CPU changes to STOP with an internal system error.

Parameters of the PVFC Function

The input value PVFC_IN is an implicit parameter and cannot be monitored at the configuration tool. The output value is accessible at measuring point MP6 (Figure 2-13).

If PVFC_ON = FALSE, (default) the PVFC_IN input is switched through to the process variable branch.



*) Default when the instance DB is created

Figure 4-19 Calling an FC Block in the Process Variable Branch

4.2.5 Monitoring the Process Variable Limits (PV_ALARM)

Application

Illegal or dangerous states can occur in a system if process values (for example motor speed, pressure, level, temperature etc.) exceed or fall below critical values. In such situations, the PV_ALARM function is used to monitor the permitted operating range. Limit violations are detected and signaled to allow a suitable reaction.

The PV_ALARM Function

The PV_ALARM function monitors four selectable limits in two tolerance bands for the process variable PV(t). If the limits are reached or exceeded, the function signals a warning at the first limit and an alarm at the second limit.

The numerical values of the limits are set in the input parameters for “Warning” and “Alarm” (Figure 4-20). If the process variable (PV) exceeds or falls below these limits, the corresponding output bits QPVH_ALM, QPVH_WRN, QPVL_WRN and QPVL_ALM are set (Figure 4-20).

To prevent the signal bits “flickering” due to slight changes in the input value or due to rounding errors, a hysteresis PV_HYS is set. The hysteresis must pass the process variable before the messages are reset.

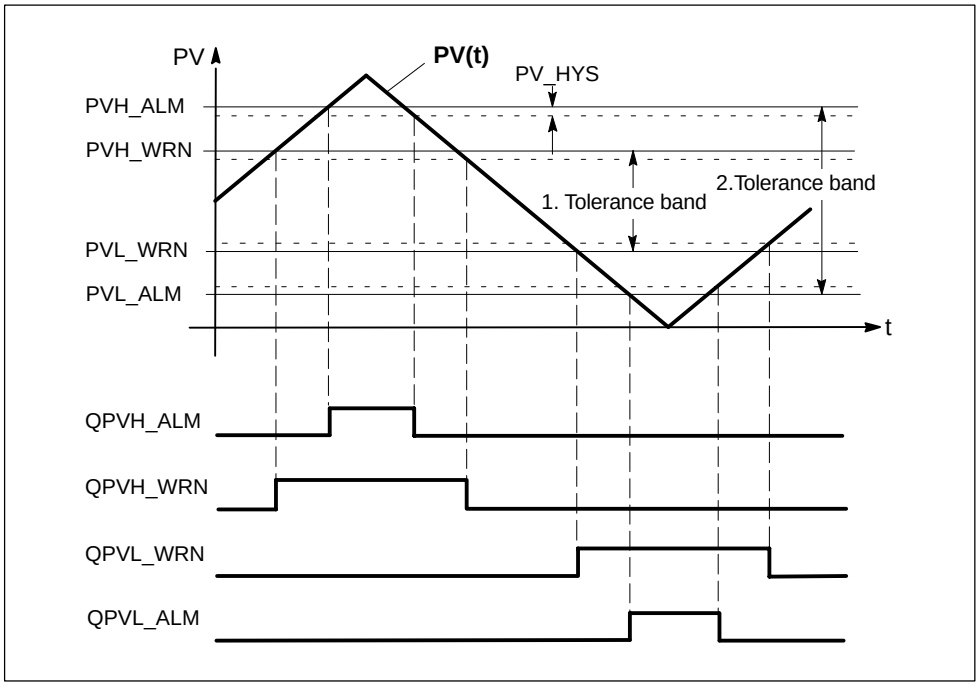


Figure 4-20 Process Variable PV – Monitoring the Limit Values

Startup and Mode of Operation

- In case of a complete restart all the signal outputs are set to zero.
- The limit value indication operates according to the following functions:

QPVH _ALM	QPVH _WRN	QPVL _WRN	QPVL _ALM	when	and
TRUE	TRUE	FALSE	FALSE	PV ↗ PV↘	PV ≥ PVH_ALM PV ≥ PVH_ALM – PV_HYS
FALSE	TRUE	FALSE	FALSE	PV ↗ PV↘	PV ≥ PVH_WRN PV ≥ PVH_WRN – PV_HYS
FALSE	FALSE	TRUE	FALSE	PV ↘ PV↗	PV ≤ PVL_WRN PV ≤ PVL_WRN + PV_HYS
FALSE	FALSE	TRUE	TRUE	PV ↘ PV ↗	PV ≤ PVL_ALM PV ≤ PVL_ALM + PV_HYS

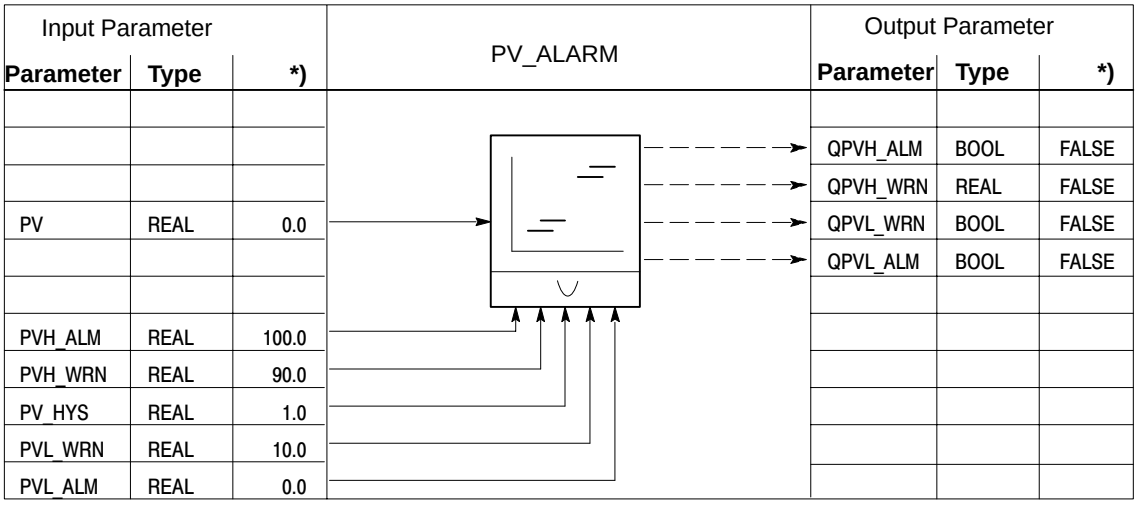
For the block to function correctly, the following must apply:

PVL_ALM < PVL_WRN < PVH_WRN < PVH_ALM

Parameters of the PV_ALARM Function

You cannot disable the PV_ALARM function. When you configure a Standard PID Control, you should therefore make sure that you set suitable limit values. Otherwise, limit value violations will be indicated using the default parameters.

Parameter	Meaning	Permitted Values
PVH_ALM	Upper PV limit 'alarm'	Techn. range of values
PVH_WRN	Upper PV limit 'warning'	Techn. range of values
PVL_ALM	Lower PV limit 'alarm'	Techn. range of values
PVL_WRN	Lower PV limit 'warning'	Techn. range of values
PV_HYS	PV hysteresis	≥ 0 [%]



*) Default when the instance block is created

Figure 4-21 Functions and Parameters of the Process Variable Limit Value Monitoring

4.2.6 Monitoring the Rate of Change of the Process Variable (ROCALARM)

Application

If the rate of change in the process variable is too fast (for example motor speed, pressure, level, temperature etc.), illegal or dangerous situations can occur in the process or plant. Here, the ROCALARM function is used to make sure that the process variable does not exceed or fall below a permitted range of change or slope. Limit violations are detected and signaled to allow a suitable reaction.

The ROCALARM Function

The ROCALARM function monitors limits for the rate of change of the process variable PV(t).

The numerical values for the rate of change limits are set at the input parameters for “up rate” and “down rate” in the positive and negative ranges of the process variable. The rate of change is an up or down rate as a percentage per second.

If the rate of change of the process variable exceeds these limits, the output signal bits QPVURLMP to QPVDRLMN are set (Figure 4-22 and 4-23).

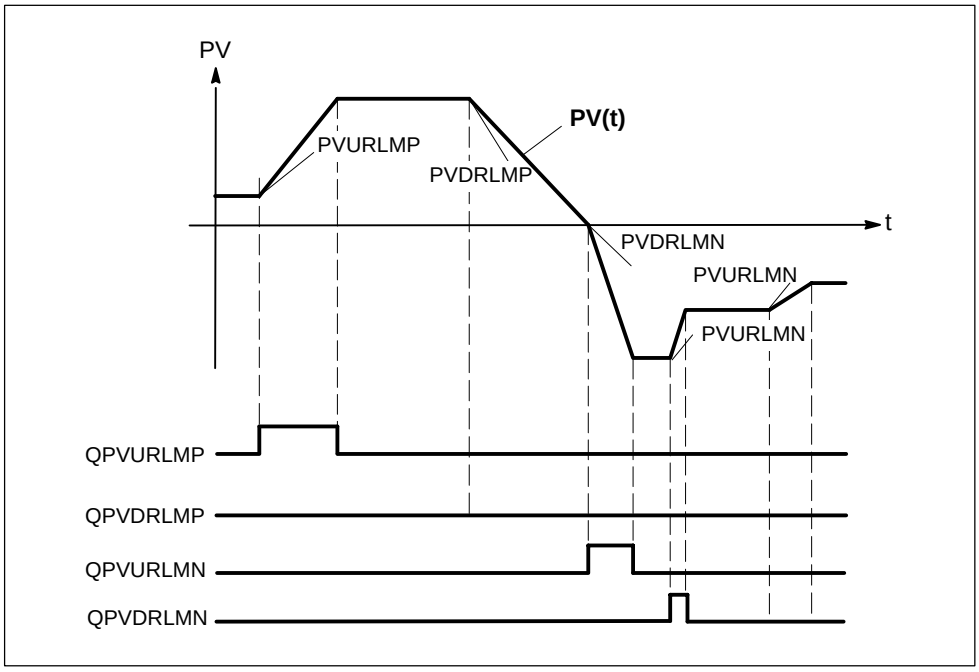


Figure 4-22 Monitoring the Rate of Change (Slope) of the Process Variable PV(t)

The ramp parameters are as follows:

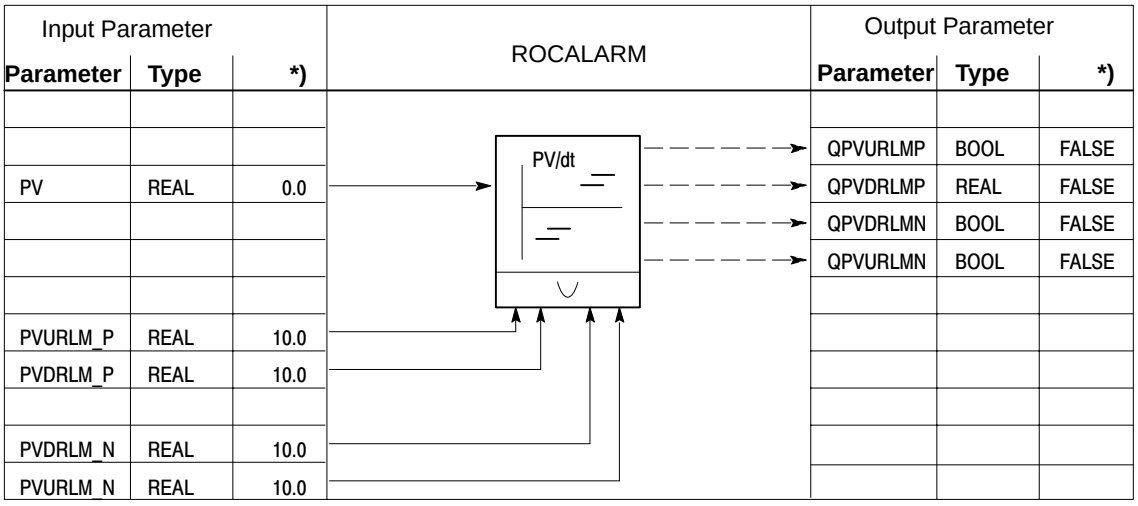
Parameter	PV Change
PVURLM_P	PV > 0 and |PV| rising
PVDRLM_P	PV > 0 and |PV| falling
PVURLM_N	PV < 0 and |PV| rising
PVDRLM_N	PV < 0 and |PV| falling

Parameters of the ROCALARM Function

You cannot disable the ROCALARM function. When you configure a Standard PID Control, you should therefore make sure that you set suitable limit values. Otherwise, limit value violations will be indicated using the default parameters (Figure 4-23).

Parameter	Meaning	Permitted range of values
PVURLM_P	PV rise limit in the positive range	$\geq$ 0 [/s]
PVDRLM_P	PV fall limit in the positive range	$\geq$ 0 [/s]
PVURLM_N	PV rise limit in the neg. range	$\geq$ 0 [/s]
PVDRLM_N	PV fall limit in the neg. range	$\geq$ 0 [/s]

The rates of change are always entered as a positive value.



*) Default when the instance block is created

Figure 4-23 Functions and Parameters of the Rate of Change Monitoring of the Process Variable PV(t)

4.2.7 Changing the Manipulated Variable Using the Configuration

Process Variable Displays and Settings in the Loop Monitor Tool

The configuration tool has its own interface to the controller FB. It is therefore possible at any time to interrupt the process variable branch and to specify your own process variable values PV_OP, for example, for test purposes when working on a PG/PC on which the configuration tool is loaded (Figure 4-24).

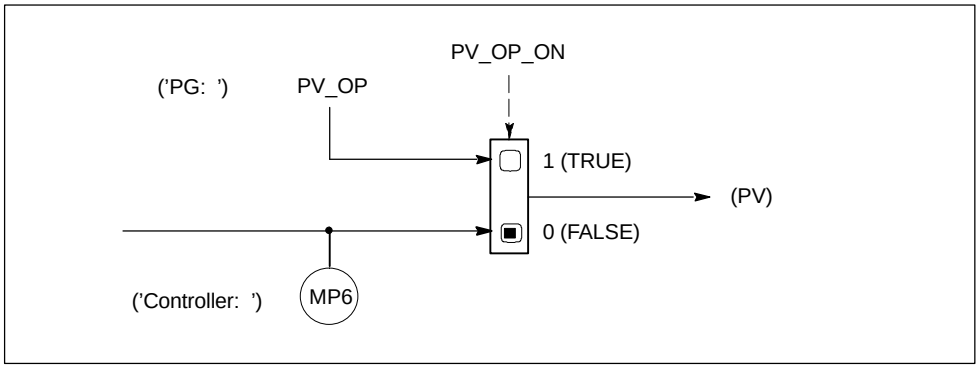


Figure 4-24 Intervening in the Process Variable Branch Using an Operator Panel

One of the three fields (labeled **process variable**) in the **loop monitor** is available for this. Here the process variable currently applied to measuring point MP6 is displayed in the “Controller:” field. The field above this (PG:) is used to display and change the parameter PV_OP.

Changeover to the Process Variable Specification by the Configuration Tool

If the switch in the configuration tool is set to ’PG: ’, the switching signal of the structure switch PVOP_ON is set to TRUE and PV_OP is enabled to the process variable PV value in the controller FB.

The value adopted with the changeover (MP6) can be viewed in the “Controller: ” display field of the **loop monitor**.

These interventions, however, only affect the process when you transfer them to the programmable logic controller by clicking the “Send” button in the **loop monitor**.

4.3 Processing the Error Signal

4.3.1 Filtering the Signal with DEADBAND Function

Application

If the process variable or the setpoint is affected by higher frequency noise and the controller is optimally set, the noise will also affect the controller output. This can, for example, lead to large fluctuations in the manipulated value at high control again when the D action is activated. Due to the increased switching frequency (step controller) this leads to faster wear and tear on the final control element.

This function reduces noise in the error signal of the controller in the settled state and thus reduces unwanted oscillation of the controller output.

The DEADBAND Function

The DEADBAND function is a selectable band in which small fluctuations in the input variable around a specified zero point are suppressed. Outside this band, the error signal ER rises or falls in proportion to the input value. You can specify the width of the DEADBAND using the parameter DEADB_W. The DEADBAND width can only have positive values.

If the input variable is within the DEADBAND, the value 0 is output (error signal = 0). The output only rises or falls by the same values as the input variable inv only when the input variable leaves this DEADBAND. This also falsifies the transferred signal when it is outside the DEADBAND. This is, however, an acceptable compromise to avoid step changes at the limits of the DEADBAND (Figure 4-25). The amount to which the signal is falsified corresponds to the value DEADB_W and can therefore be checked easily.

The DEADBAND operates according to the following functions:

(ER) = INV + DEADB_W where $inv < -DEADB\_W$

(ER) = 0 where $-DEADB\_W \leq inv \leq +DEADB\_W$

(ER) = INV + DEADB_W where $inv > +DEADB\_W$

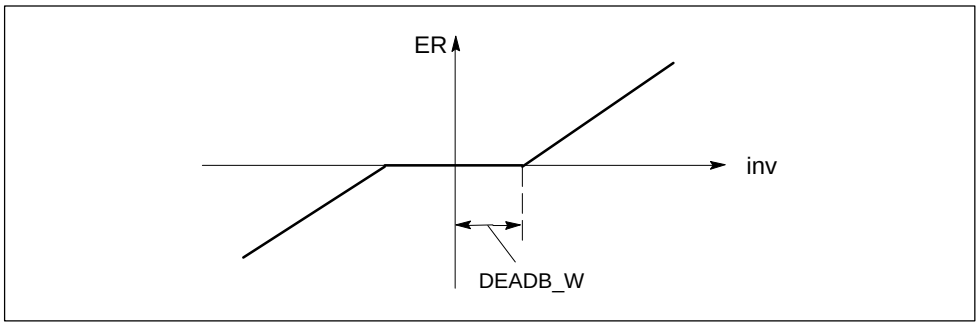


Figure 4-25 Filtering Noise Affecting the Error Signal ER using a DEADBAND

Parameters of the DEADBAND Function

The DEADBAND function can be disabled. The effects of signal filtering can be monitored at the “ER” output using the **curve recorder** (configuration tool) (Figure 2-13.

The DEADB_W parameter can be selected between 0.0 and the upper limit of the technical range of values.

Parameter	Meaning	Permitted Values
DEADB_W	Dead band width (= range zero to dead band upper limit)	0 ... Upper limit of the technical range of values

Input Parameter			DEADBAND	Output Parameter		
Parameter	**Type**	***)**		**Parameter**	**Type**	***)**
DEADB_ON	BOOL	FALSE				
				ER	REAL	0.0
inv	REAL					
DEADB_W	REAL	1.0				

*) Default when the instance block is created

Figure 4-26 Functions and Parameters of the DEADBAND Function in the Error Signal Channel

4.3.2 Monitoring the Error Signal Limit Values (ER_ALARM)

Application

If the process variable deviates from the setpoint by a large amount, undesirable states can occur in the process. The ER_ALARM function monitors the error signal and detects when it exceeds or falls below the permitted range. ER_ALARM detects and indicates any such limit violations so that a suitable reaction can be started.

The ER_ALARM Function

The ER_ALARM function monitors four selectable limits in two tolerance bands for the error signal ER(t). If the limits are reached or exceeded, the function signals a warning at the first limit and an alarm at the second limit.

The numerical values of the limits are set in the input parameters for “Warning” and “Alarm” (Figure 4-28). If the error signal (ER) exceeds or falls below these limits, the corresponding output bits QERN_ALM to QERP_ALM are set.

To prevent the signal bits “flickering” due to slight changes in the input value or due to rounding errors, a hysteresis ER_HYS is set. The error signal must pass the hysteresis before the messages are reset.

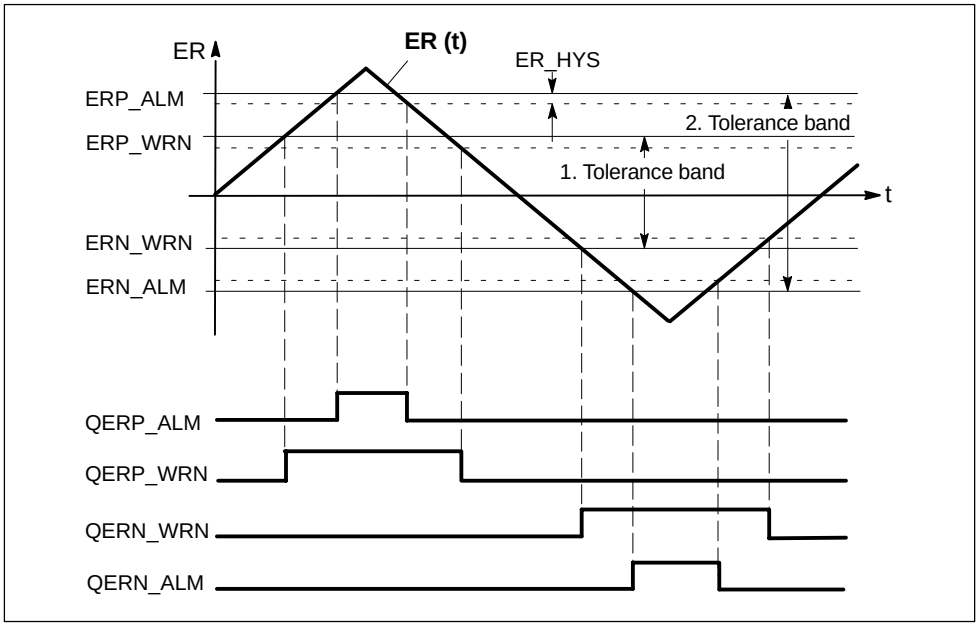


Figure 4-27 Monitoring the Limit Values of the Error Signal ER

Startup and Mode of Operation

- In case of a complete restart all the signal outputs are set to zero.
- The limit value indication operates according to the following functions:

QERP _ALM	QERP _WRN	QERN _WRN	QERN _ALM	when:	and:
				ER ↗	ER ≥ ERP_ALM
TRUE	TRUE	FALSE	FALSE	ER↘	ER ≥ ERP_ALM – ER_HYS
				ER ↗	ER ≥ ERP_WRN
FALSE	TRUE	FALSE	FALSE	ER↘	ER ≥ ERP_WRN – ER_HYS
				ER ↘	ER ≤ ERN_WRN
FALSE	FALSE	TRUE	FALSE	ER ↗	ER ≤ ERN_WRN + ER_HYS
				ER ↘	ER ≤ ER_ALM
FALSE	FALSE	TRUE	TRUE	ER ↗	ER ≤ ERN_ALM + ER_HYS

For the block to function correctly, the following must apply:

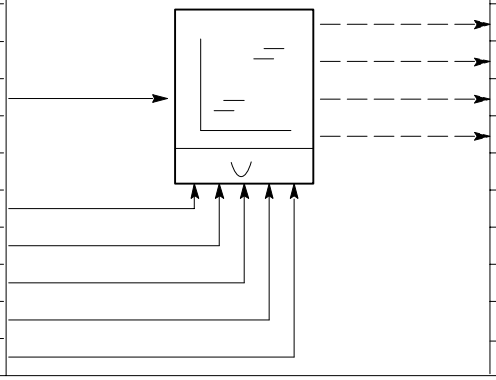
ERN_ALM < ERN_WRN < ERP_WRN < ERP_ALM

Parameters of the ER_ALARM Function

You cannot disable the ER_ALARM function. When you configure a standard controller, you should therefore make sure that you set suitable limit values. Otherwise, limit value violations will be indicated using the default parameters (Figure 4-28).

Parameter	Meaning	Permitted Values
ERP_ALM	Upper ER limit 'alarm'	≥ 0.0, technical range of values
ERP_WRN	Upper ER limit 'warning'	≥ 0.0, technical range of values
ERN_WRN	Unterer ER limit 'warning'	≤ 0.0, technical range of values
ERN_ALM	Unterer ER limit 'alarm'	≤ 0.0, technical range of values

Input Parameter			ER_ALARM	Output Parameter		
Parameter	**Type**	***)**		**Parameter**	**Type**	***)**
				QERP_ALM	BOOL	FALSE
				QERP_WRN	REAL	FALSE
ER	REAL	0.0		QERN_WRN	BOOL	FALSE
				QERN_ALM	BOOL	FALSE
ERP_ALM	REAL	100.0				
ERP_WRN	REAL	90.0				
ER_HYS	REAL	1.0				
ERN_WRN	REAL	-90.0				
ERN_ALM	REAL	-100.0				



*) Default when the instance block is created

Figure 4-28 Functions and Parameters of the Error Difference ER Limit Value Monitoring

4.4 The PID Controller Functions

Normalization of the Input Variables ER and PV

The input variables ER and PV of the PID controller are normalized before controller processing to the range of 0 to 100 in accordance with the following equations:

- If the square-root function is de-activated (SQRT_ON = FALSE):
  - $ER_{Normalized}$ = ER * 100.0 / (NM_PVHR – NM_PVLR)
  - $PV_{Normalized}$ = (PV – NM_PVLR) * 100.0 / (NM_PVHR – NM_PVLR)
- If the square-root function is activated (SQRT_ON = TRUE):
  - $ER_{Normalized}$ = ER * 100.0 / (SQRT_HR – SQRT_LR)
  - $PV_{Normalized}$ = (PV – SQRT_LR) * 100.0 / (SQRT_HR – SQRT_LR)

This normalization is carried out so that the gain factor GAIN of the PID controller can be entered without dimensions. If the upper and lower limit of the physical measuring range changes (for example from bar to mbar), the gain factor then does not have to be changed.
The normalized input variables $ER_{Normalized}$ and $PV_{Normalized}$ cannot be monitored.

Control Algorithm and Controller Structure

Within the cycle of the configured sampling time, the manipulated variable of the continuous controller is calculated from the error signal in the PID algorithm. The controller is designed as a parallel structure (Figure 4-29). The proportional, integral and derivative actions can be deactivated individually.

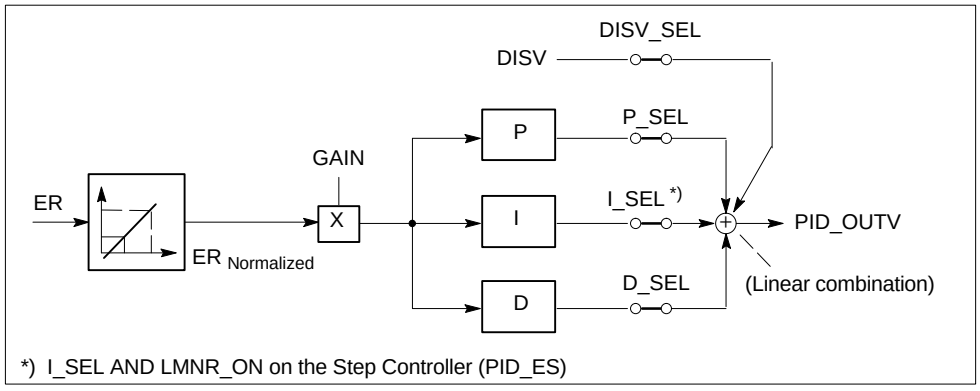


Figure 4-29 Control Algorithm of the Standard PID Control (Parallel Structure)

Feedforward control:

A disturbance **DISV** can also be connected to the PID_OUTV output signal of the controller. This function is enabled or disabled in the PID dialog box of the configuration tool using the DISV_SEL structure switch or with “Disturbance Variable On”.

PD action in the feedback path:

In the parallel structure, each action of the control algorithm receives the error signal as its input signal. In this structure, step changes in the setpoint affect the controller directly. The manipulated variable is affected immediately by step changes in the setpoint via the P and the D components.

Designing the controller differently, however, so that the P and D actions are in the feedback path, guarantees that step changes in the setpoint do not cause sudden changes in the manipulated variable (Figure 4-30). Using this structure, the I action processes the error signal as its input signal and only the **negative** error signal (factor = -1) is connected to the P and D actions.

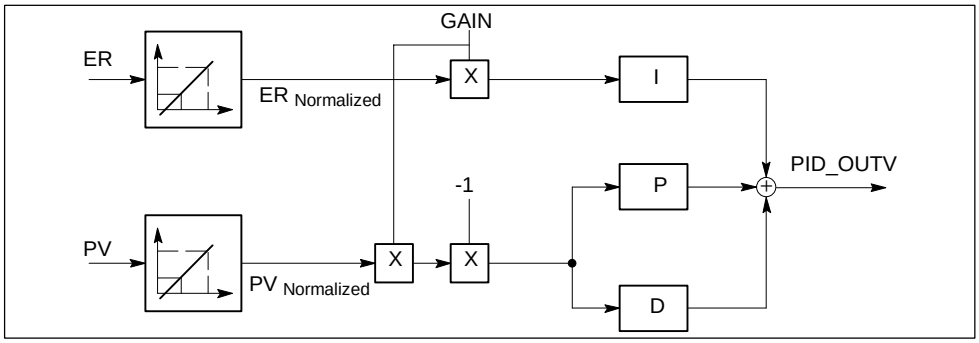


Figure 4-30 Control Algorithm with the P and D Actions in the Feedback Path

Defining the Controller Structure

To define an effective controller structure, there are five switches available (Table 4-3). The setting of theis structure switch is carried out in the configuration tool by selecting the P, I and D actions, for the P and D actions also in the feednack path. This is carried out after the PID controller block (block diagram) has been selected in the operating window ”PID”.

Table 4-3 Selecting the Controller Structure

Switch / Mode	P_SEL	I_SEL *)	D_SEL	PFDB_SEL	DFDB_SEL
P controller	TRUE	FALSE	FALSE	FALSE	FALSE
P controller (P in f. path)	TRUE	FALSE	FALSE	TRUE	FALSE
PI controller	TRUE	TRUE	FALSE	FALSE	FALSE
PI controller (P in f. path)	TRUE	TRUE	FALSE	TRUE	FALSE
PD controller	TRUE	FALSE	TRUE	FALSE	FALSE
PD controller (P in f. path)	TRUE	FALSE	TRUE	FALSE	TRUE
PID controller	TRUE	TRUE	TRUE	FALSE	FALSE
PID controller (P/D in f. path)	TRUE	TRUE	TRUE	FALSE	TRUE

*) With the step controller without position feedback signal (PID_ES with LMNR_ON = FALSE), the I action in the PID algorithm is set to zero.

Reversing the Controller Functions

You can **reverse** the controller from

- the rising process variable PV(t) → falling manipulated variable PID_OUTV(t) to the
- rising process variable PV(t) → rising manipulated variable PID_OUTV(t)

by setting a negative proportional gain for the GAIN parameter. Its sign value decides the direction of the control action of the continuous controller.

P Controller

In a P controller, the I and D actions are disabled. (I_SEL and D_SEL = FALSE). This means that if the error signal ER is 0, the output signal OUTV is also 0. If an operating point ≠ 0 is required, in other words a numerical value for the output signal when the error signal is zero, the I action must be activated (Figure 4-31).

With the I action, an operating point ≠ 0 can be specified for the P controller by setting an initialization value I_ITLVAL. To do this, set switch 'I_ITL_ON' and 'I_SEL' to TRUE.

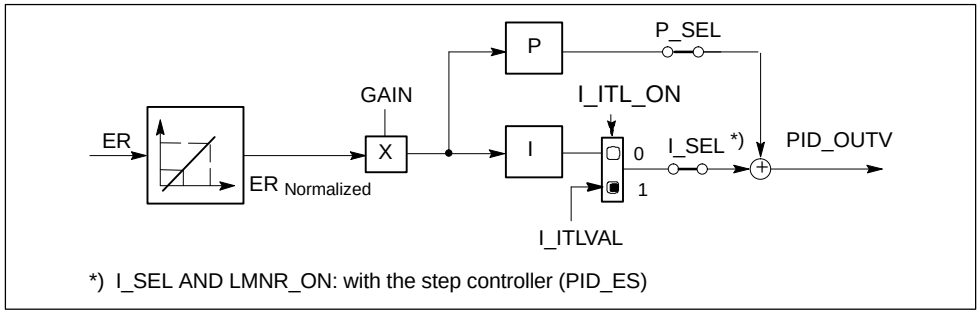


Figure 4-31 P Controller with Operating Point Setting

The step response of the P controller in the time domain is as follows:

$$PID\_OUTV(t) = I\_ITLVAL + GAIN * ER_{Normalized}(t)$$

Legend

PID_OUTV(t)	the man. variable in the automatic controller mode
I_ITLVAL	the operating point of the P controller
GAIN	the controller gain
$ER_{Normalized}(t)$	the normalized error gain

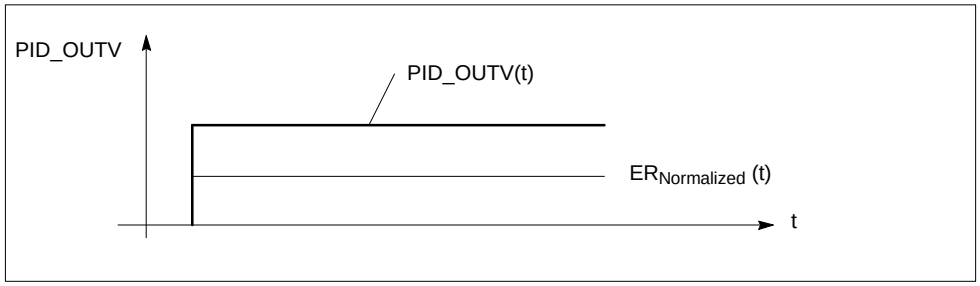


Figure 4-32 Step Response of the P Controller

PI Control

In a PI controller, the D action is disabled. (D_SEL = FALSE). A PI controller adjusts the output variable PID_OUTV using the I action until the error signal ER becomes zero. This only applies when the output variable does not exceed the limits of the manipulated value.
The step response in the time domain (Figure 4-33) is as follows:

$$PID\_OUTV(t) = GAIN * ER_{normalized}(0)\left(1 + \frac{1}{TI} * t\right)$$

Legend:

PID_OUTV(t)	the man. variable in the automatic controller mode
GAIN	the controller gain
$ER_{Normalized}(0)$	The jump height of the normalized error signal
TI	reset time

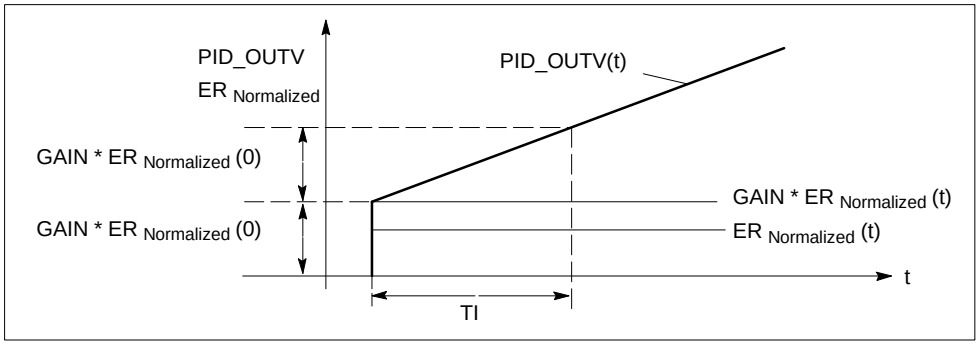


Figure 4-33 Step Response of the PI Controller

To allow a smooth changeover from the manual mode to the automatic mode of the PI controller, the output signal LMNFC_IN – LMN_P – DISV is switched to the internal memory of the integrator when the manipulated variable is being adjusteded manually (Figure 4-34). When using the step controller with position feedback signal, the integrator is corrected to the output signal LMN.

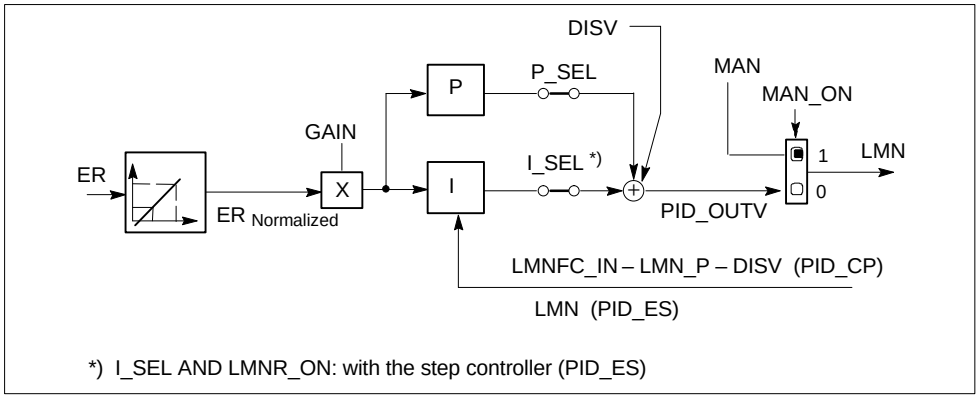


Figure 4-34 PI Controller with Smooth Switchover from Manual → Automatic Mode

To achieve a purely integrating control action disable the P action with P_SEL.

PD Controller

In the PD controller, the I action is deactivated (I-SEL = FALSE). This means that if the error signal ER is zero, the output signal OUTV is also zero. If an operating point $\neq 0$ is required, in other words a numerical value must be set for the output signal when the error signal is zero, then the I branch must be activated (Figure 4-31).

With the I action, an operating point $\neq 0$ can be specified for the P controller by setting an initialization value I_ITLVAL. To do this, set switch 'I_ITL_ON' and 'I_SEL' to TRUE.

The PD controller forms the input value ER(t) proportional to the output signal and adds the D action formed by differentiating ER(t) that is calculated with twice the accuracy according to the trapezoidal rule (Padé approximation). The time response is determined by the derivative action time TD.

To damp signals and to suppress disturbances, a first order time lag (adjustable time constant: TM_LAG) is integrated in the algorithm for forming the D action. Generally a small value is adequate for TM_LAG to achieve a successful outcome. If TM_LAG $\leq$

CYCLE/2 is configured, the time lag is disabled.

The step response in the time domain (Figure 4-35) is as follows:

$$\text{PID\_OUTV}(t) = \text{GAIN} * \text{ER}_{\text{normalized}}(0)\left(1 + \frac{\text{TD}}{\text{TM\_LAG}} * e^{-\frac{t}{\text{TM\_LAG}}}\right)$$

Legend:

PID_OUTV(t)	the man. variable in the automatic controller mode
GAIN	the controller gain
ER $_{\text{Normalized}}(0)$	The jump height of the normalized error signal
TD	derivative action time
TM_LAG	time lag

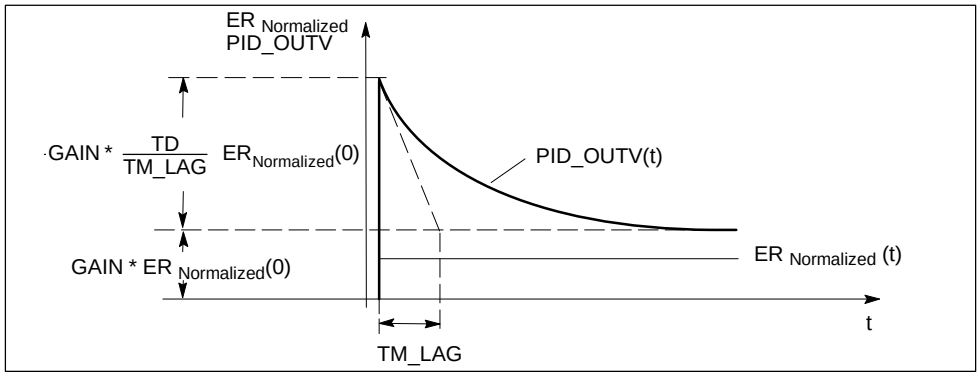


Figure 4-35 Step Response of the PD Controller

PID Controller

In a PID controller, the P, I and D actions are activated (P_SEL, I_SEL, D_SEL = TRUE). A PID controller adjusts the output variable PID_OUTV using the I action until the error signal ER becomes zero. This only applies when the output variable does not exceed the limits of the manipulated value. If the manipulated variable range limits are exceeded, the I action retains the value that was set when the limit was reached (anti reset wind-up)

The PID controller forms the normalized input value $ER_{Normalized}(t)$ proportional to the output signal and adds the actions formed by differentiating and integrating $ER_{Normalized}(t)$ that are calculated with twice the accuracy according to the trapezoidal rule (Padé approximation). The time response is determined by the derivative action time TD and the reset time TI.

To damp signals and to suppress disturbances, a first ordertime delay (adjustable time constant: TM_LAG) is integrated in the algorithm for forming the D action. Generally a small value is adequate for TM_LAG to achieve a successful outcome. If TM_LAG $\leq$ CYCLE/2 is selected, the time lag is disabled.

The step response in the time domain (Figure 4-36) is as follows:

$$PID\_OUTV(t) = GAIN * ER_{normalized}(0)\left(1 + \frac{1}{TI} * t + \frac{TD}{TM\_LAG} * e^{-\frac{t}{TM\_LAG}}\right)$$

Legend:

PID_OUTV(t)	the man. variable in the automatic controller mode
$ER_{Normalized}(0)$	The jump height of the normalized error signal
GAIN	the controller gain (= GAIN)
TI	reset time
TD	derivative action time
TM_LAG	time lag

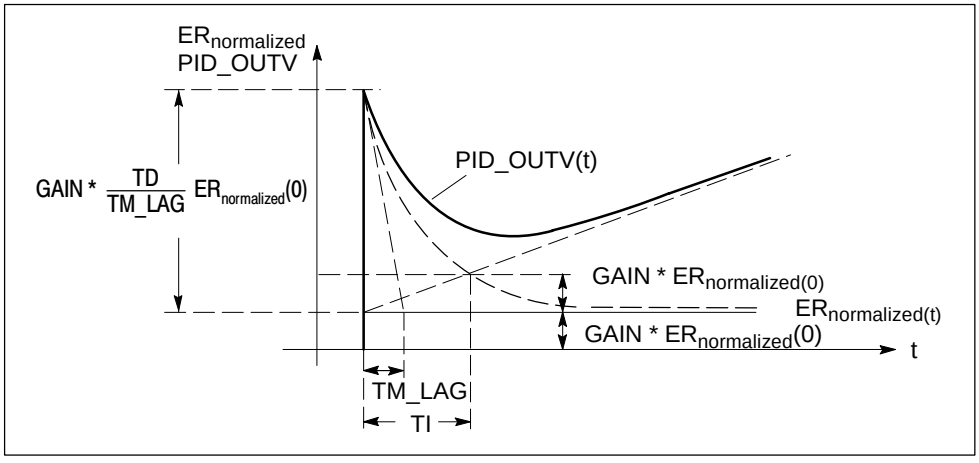


Figure 4-36 Step Response of the PID Controller

Note

You have to adjust TM_LAG if you change TD.

Recommendation: $5 \leq (TM / TM\_LAG) \leq 10$

Using and Assigning Parameters to the PID Controller

The PI/PID functions of the Standard PID Control are capable of controlling most processes in industry. Functions and methods beyond the scope of this controller are only necessary in special situations
(→ *see Section 1.2*, further S7 software packages for control tasks).

One practical problem nevertheless remains the assignment of parameters to PI/PID controllers, in other words finding the “right” settings for the controller parameters. The quality of the parameter assignment is the decisive factor in the quality of the PID control and demands either considerable practical experience, specialist knowledge or a lot of time.

These difficulties can be eliminated by using the **configuration tool**. The **process identification** function provided by this tool allows the controller parameters to be set initially using an adaptive method. The process identification creates a process model and then calculates the most suitable settings for the controller parameters. This largely automatic procedure saves the user from having to tune the installed PID controller manually using on-line techniques.

4.5 Signal Processing in the PID Controller Algorithm

4.5.1 Integrator (INT)

Application

The function of the integrator is used in standard PI and PID controllers to implement the I action. The integral action in these controllers ensures that by correcting the operating point, the error signal can become zero at any value of the manipulated variable.

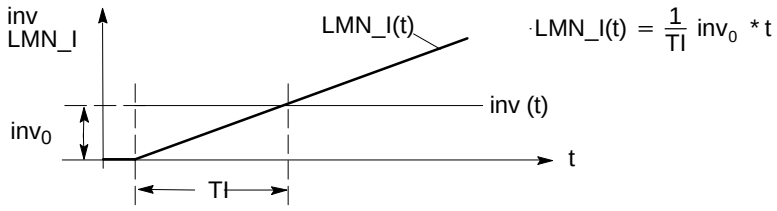
The INT Function

The integral action generates an output signal whose rate of change is proportional to the change in the absolute value of the input variable. The time response is determined by the reset time TI.

The transfer function in the time domain is as follows:

$$\cdot OUTV(t) = \frac{1}{TI} \int inv(t)\, dt$$

The step response to an input step inv_0 is as follows:



Legend:

LMN_I(t)	the output value of the integrator
inv_0	the step size at the integrator input
TI	reset time

Permitted Ranges for TI and CYCLE

Due to the limited accuracy of the REAL numbers calculated in the CPU, the following effect can occur during integration: If the sampling time CYCLE is too small compared with the reset time TI and if the input value inv of the integrator is too small compared with its output value OUTV, the integrator does not respond and remains at its current output value.

This effect can be avoided by remembering the following rule when assigning parameters:

$$CYCLE > 10^{-4} * TI$$

With this setting, the integrator reacts to changes in the input values that are in the range of ten millionths of a percent of the current output value:

$$inv > 10^{-10} * OUTV$$

To ensure that the transfer function of the integrator algorithm corresponds to the analog response, the sampling time should be less than 20% of the reset time TI, in other words TI should be five times higher than the selected sampling time:

$$CYCLE < 0.2 * TI$$

The algorithm permits values for the sampling time up to $CYCLE \leq 0.5 * TI$.

Startup and Modes

- **Initializing the I action**

  If I_ITL_ON = TRUE is activated, the initialization value I_ITLVAL is switched to the output. At the change to the normal mode when I_ITL_ON = FALSE is set, the integrator starts to integrate its input value starting at I_ITLVAL (Figure 4-37).

  – Continuous controller PID _CP

    In manual operation the integral component of the controller is tracked so that the controller begins with a sensible manipulated variable when changing over to automatic mode.The following settings can be selected:

    Smooth changeover from manual to automatic
    If SMOO_CHG = TRUE (default) the integral component in manual operation is set so that the manipulated variable remains unchanged during manual-automatic changeover. An active system deviation is compensated slowly.

    No smooth changeover from manual to automatic mode
    If SMOO_CHG = FALSE the integral component in manual operation is set so that the manipulated variable makes a jump (through the proportional and derivative components) starting from the manual manipulated value during the manual-automatic changeover. The jump height corresponds to the manipulated value change at a setpoint jump from the current process variable to the current setpoint value. The active system deviation is compensated faster. This is desirable, for exampla, for temperature processes.

    However, if the proportional component is placed in the feedback (PFDB_SEL = TRUE), only the actual value acts on the proportional component.As for a setpoint jump the manipulated variable therefore does not make a jump through the proportional component during the manual-automatic changeover. The changeover is smooth. The same applies to the derivative component, if this was also placed in the feedback (DFDB_SEL = TRUE).

  – Step controller PID_ES

    The integral component is set in manual operation so that the final controlling element is traveled by the jump height of the proportional component starting from the current position during the manual-automatic changeover.The jump height of the proportional component corresponds to the manipulated value change at a setpoint jump from the current process variable to the current setpoint value.

    However, if the proportional component is placed in the feedback (PFDB_SEL = TRUE), only the process variable acts on the proportional component. As for a setpoint jump the manipulated variable therefore does not make a jump through the proportional component during the manual-automatic changeover. The changeover is smooth. The derivative component is set during manual operation to zero ana also remains zero during the manual-automatic changeover.

- **Manual mode**

  If the actuating signal is set manually, either when MAN_ON=TRUE or LMNOP_ON = TRUE or CAS_ON = TRUE is set, the internal memory value of the integrator is corrected to the LMNFC_IN – LMN_P – DISV value (Figure 4-34). In the step controller with a position feedback signal (PID_ES) the integrator is corrected to the output signal LMN.

- **Holding the integrator**

  The binary inputs INT_HPOS and INT_HNEG can be used to block the integrator in the positive or negative direction. This can be advisable at controller cascades.If, for example, the manipulated variable of the secondary controller approaches the upper limit, a further increase in the manipulated variable of the master controller can be prevented by its integrator. This is realized by the following instruction statements:

STL	**Explanation**
U “Secondary controller”.QLMN_HLM	
= “Master controller”.INT_HPOS	
U “Secondary controller”.QLMN_LLM	
= “Master controller”.INT_HNEG	

- **Integration**

  If the switch I_SEL = TRUE is set, integration is activated starting at the I_ITLVAL value. The dynamic response of the function is determined by the reset time TI.

  If integration is deactivated (I_SEL = FALSE), the I action, in other words the internal memory and the output LMN_I of the integrator, is set to zero.

Switch / **Mode**	**I_ITL_ON**	**MAN_ON** or **LMNOP_ON**	**INT_HPOS**	**INT_HNEG**
Initialize (LMN_I)	TRUE	any	any	any
Manual mode	FALSE	TRUE	any	any
Blocking the integrator in the pos. direction	FALSE	FALSE	TRUE	FALSE
Blocking the integrator in the negative direction	FALSE	FALSE	FALSE	TRUE
Blocking the integrator in both directions	FALSE	FALSE	TRUE	TRUE
Integration	FALSE	FALSE	FALSE	FALSE

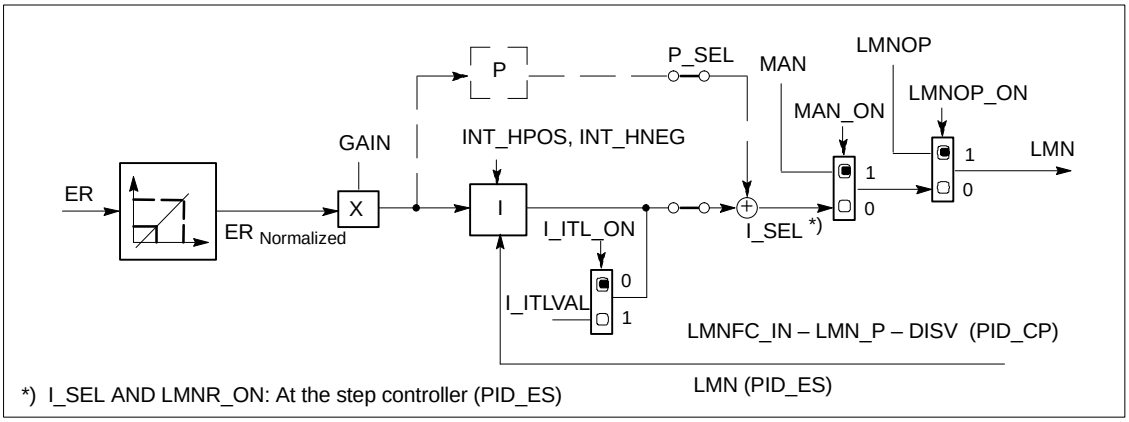


Figure 4-37 Modes of the Integrator in the PI/PID Controller

Limit behavior

The output and the memory of the integrator are limited by the upper and lower limits LMN_HLM and LMN_LLM (anti reset wind-up).

Parameters of the INT Function

The OUTV output value of the integrator can be monitored at parameter LMN_I.

Parameter	Meaning	Permitted Values
TI	Reset time	$\geq$ 5 * CYCLE
I_ITLVAL	Initialization value for I action	–100.0 to +100.0 [%]

Input Parameter			INT	Output		
Parameter	**Type**	***)**		**Signal**	**Type**	***)**
I_SEL	BOOL	FALSE				
I_ITL_ON	BOOL	FALSE				
INT_HPOS	BOOL	FALSE				
INT_HNEG	BOOL	FALSE				
				LMN_I	REAL	0.0
inv	REAL	0.0				
TI	TIME	T#20s				
I_ITLVAL	REAL	0.0				

*) Default when the instance DB is created

Figure 4-38 Functions and Parameters of the Integrator

4.5.2 Derivative Unit (DIF)

Application

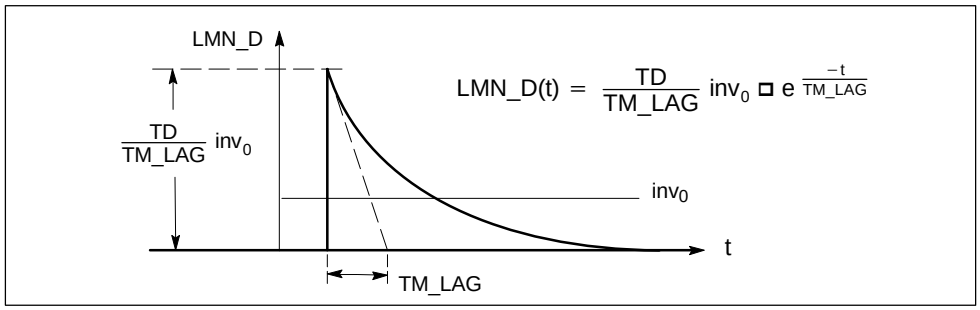
The function of the derivative unit is used to implement the D action for standard PD and PID controllers. The process variable is differentiated dynamically.

The DIF Function

The derivative action generates an output signal whose value changes proportional to the rate of change of the input value. The time response is determined by the derivative action time TD and the time lag of the derivative unit TM_LAG.

To damp signals and to suppress disturbances, a first order time delay is integrated whose time constant is set at the parameter TM_LAG.

The step response to an input step inv_0 is as follows:



Legend:

LMN_D(t)	the output value of the derivative unit
inv_0	the step value at the derivative unit input
TD	the derivative action time
TM_LAG	time lag

Permitted Ranges for TD and CYCLE

To allow the derivative unit to process its calculation algorithm correctly in the CPU, keep to the following rules when assigning the time constants:

TD $\geq$ CYCLE and

TM_LAG $\geq$ 0.5 * CYCLE

If a value less than CYCLE is set, the derivative unit operates as if TD had the same value as CYCLE.

If TM_LAG is set to a value < 0.5 * CYCLE, the derivative unit operates without a delay. The input step change is then multiplied by the factor TD/CYCLE and this value is applied to the output as a “needle pulse”. This means that in the next processing cycle, LMN_D is reset to zero.

Startup and Modes

- **Manual mode**
  If a smooth changeover from manul to automatic mode was selected (SMOO_CHG = TRUE), the derivative component is set to zero in manual mode. The changeover to automatic mode is carried out without a manipulated variable jump.

  If no smooth changeover from manual to automatic mode was selected (SMOO_CHG = FALSE), the derivative component is set in manual mode to a value which corresponds to the active error signal. The changeover to automatic mode is carried out with a manipulated variable jump which adjusts the error signal faster.

- **Differentiation**
  If the D_SEL = TRUE switch is set, the derivative action is activated. The dynamic response of the function is determined by the value of the derivative action time TD and the time lag TM_LAG.

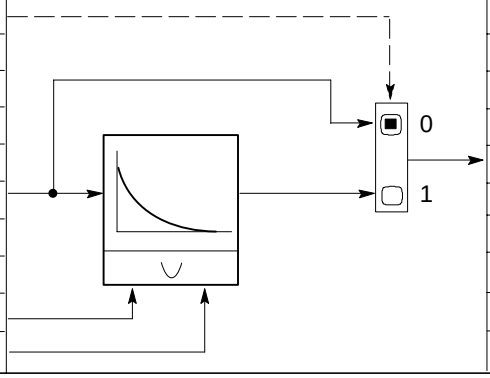
  If the derivative action is turned off (D_SEL = FALSE), the D action, in other words the internal memory and the LMN_D output of the derivative unit, is set to zero.

Switch / **Mode**	**MAN_ON** or **LMNOP_ON**
Manual mode	TRUE
Derivative action	FALSE

Parameters of the DIF Function

The output value of the derivative unit can be monitored at the parameter LMN_D.

Parameter	Meaning	Permitted Values
TD	Derivative action time	$\geq$ CYCLE
TM_LAG	Time lag of the D component	$\geq$ 0.5 * CYCLE

Input Parameter			DIF	Output		
Parameter	**Type**	***)**		**Signal**	**Type**	***)**
D_SEL	BOOL	FALSE				
				LMN_D	REAL	0.0
inv	REAL	0.0				
TD	TIME	T#10s				
TM_LAG	TIME	T#2s				

*) Default when the instance DB is created

Figure 4-39 Functions and Parameters of the Derivative Unit

The Continuous Controller (PID_CP) 5

5.1 Control Functions of the Continuous PID Controller

The PID_CP Function Block

Apart from the functions in the setpoint and process variable branch, the function block (FB) implements a complete PID controller with continuous manipulated variable output with the option of adjusting the manipulated value manually. Subfunctions can be enabled of disabled.

Using the FB, you are in a position to control technical processes and systems with continuous input and output variables on SIMATIC S7 programmable logic controllers. The controller can be used as a fixed setpoint controller either individually or in multi-loop control systems as a cascade, blending or ratio controller.

Block Diagram of the Continuous Controller

The mode of operation is based on the PID control algorithm of the sampling controller with an analog output signal, if necessary, supplemented by a pulse generator stage for generating pulse-duration modulated output signals for two or three-step controllers with proportional actuators.

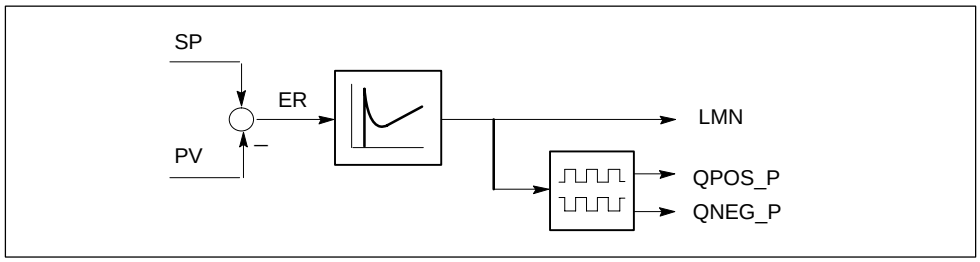


Figure 5-1 Block Diagram of the Controller with Continuous Actuating Signal ("Standard PID Control" Software Package)

Complete Restart/Restart

The PID_CP function block has an initialization routine that is run through when the input parameter COM_RST = TRUE is set.

Ramp soak (RMP_SOAK)

When the ramp soak is activated, the time slices DB_NBR PI[0 to NBR_PTS].TMV are totalled between the coordinates and indicated at the total time T_TM and total time remaining RT_TM outputs.

If PI[n].TMV is modified on-line or if TM_CONT and TM_SNBR are set, the total time and total time remaining of the ramp soak also change. Since the calculation of T_TM and RS_TM greatly increases the processing time of the RMP_SOAK function when a large number of time slices are involved, this calculation is only performed after a complete restart or when TUPDT_ON = TRUE is set.

Integral action (INT)

When the controller starts up, the integrator is set to the initialization value I_ITLVAL and the integral action is output at the LMN_I output. When it is called by a cyclic interrupt, it starts at this value.

All other outputs are set to their default values.

5.2 Processing the Manipulated Variable Signal

5.2.1 Modes Affecting the Manipulated Variable Signal

Manual Mode and Changing Modes

In addition to the “automatic” mode with the output switched to the output of the PID algorithm (PID_OUTV), the Standard PID Control also has two modes in which the manipulated variable can be influenced manually: “Manual mode without generator” and “Manual mode with up/down generator” (MAN_GEN).

Using the parameter MAN the manipulated variable can be adjusted externally either setting the value manually or by the user program setting the value. The input value MAN is limited to the manipulated variables LMN_HLM upper) and LMN_LLM (lower).

The structure of the manual value function and how it is connected can be seen in the following diagram (Figure 5-2). If MAN_GEN is activated when the controller is in a different mode, the manipulated value currently active at the output of MAN_GEN is used. The changeover to the manual value generator is therefore always smooth.

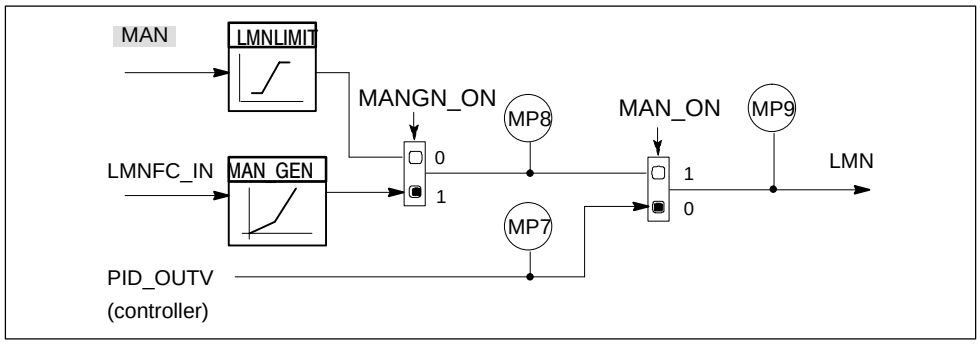


Figure 5-2 Manual Value Generation at the Standard PID Control

Automatic Mode

If MAN_ON = FALSE (block diagram in the configuration tool) is selected, the manipulated value of the PID algorithm is connected to the output. In manual mode (MAN_ON = TRUE) the integral component of the controller is tracked so that the conteoller begins with a sensible manipulated variable when changing over to automatic mode (refer to “Startup and operating mode” in Section 4.5.1). The output of the PID algorithm is applied to measuring point MP7.

In automatic mode the manual value MAN is tracked to the manipulated variable (minus the derivative component). When you change over to manual mode the manipulated variable therefore remains at the value last calculated. It can only be changed by operator control.

Manual Mode Without Generator

In this mode (MANGN_ON = FALSE and MAN_ON = TRUE) the manual value is entered as an absolute value at the MAN input. The manual manipulated value is indicated at measuring point MP8.

Manual Mode With Generator

In this mode (MANGN_ON = TRUE and MAN_ON = TRUE), the current manipulated value is increased or decreased using the MAN_GEN switch within the limits of the manipulated variable.

Switch Settings for the Modes

The following table illustrates the possible modes of the continuous controller with the required values for the structure switches.

Table 5-1 Modes of the Continuous Controller

Switch / Mode	MANGN_ON	MAN_ON
Automatic mode	any	FALSE
Manual mode with absolute value	FALSE	TRUE
Manual mode with up/down switch	TRUE	TRUE

5.2.2 Manual Value Generator (MAN_GEN)

Application

This function influences the manipulated value manually with the aid of an up/down switch. The selected value is indicated simultaneously at MP8.

The MAN_GEN Function

The MAN_GEN function generates a manipulated value that can be set and modified using a switch. The output variable outv can be increased or decreased in steps at the binary inputs MANUP and MANDN.

The range through which the manipulated value can be adjusted is limited by the upper/lower limits LMN_HLM/LMN_LLM that can be set with the limit function LMNLIMIT. The numerical values of the limits (as percentages) are set using the corresponding input parameters. To allow small changes to be made, the controller should not have a sampling time of more than 100 ms.

The rate of change of the output variable depends on the length of time that MANUP or MANDN is activated and on the currently selected limits, as follows: During the first 3 seconds after setting MANUP or MANDN:

$$\text{The increase in outv} = \frac{\text{LMN\_HLM} - \text{LMN\_LLM}}{100\ \text{s}}$$

$$\text{afterwards:} = \frac{\text{LMN\_HLM} - \text{LMN\_LLM}}{10\ \text{s}}$$

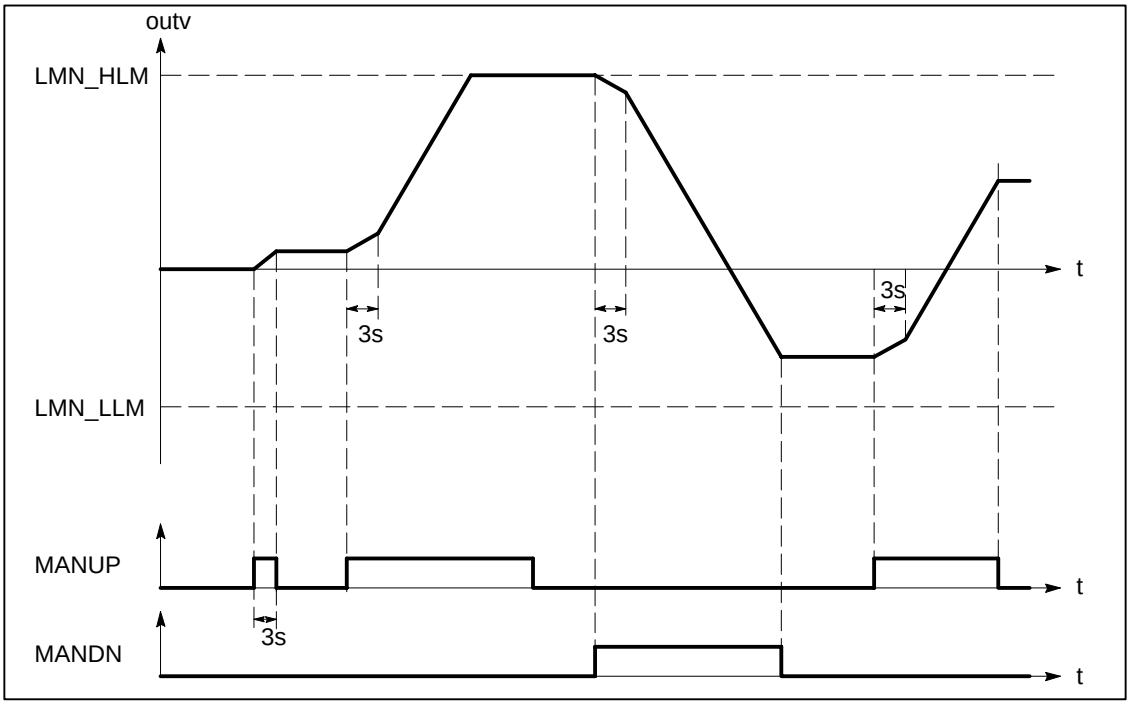


Figure 5-3 Changing the Manipulated Value in Accordance with the Switches MANUP and MANDN

At a sampling time of 100 ms and a manipulated variable range of –100.0 to 100.0%, the manipulated value changes during the first three seconds by 0.2% per cycle. If the time for which MANUP is activated is increased, the rate of change is then increased ten fold, in this case to 2% per cycle (Figure 5-3).

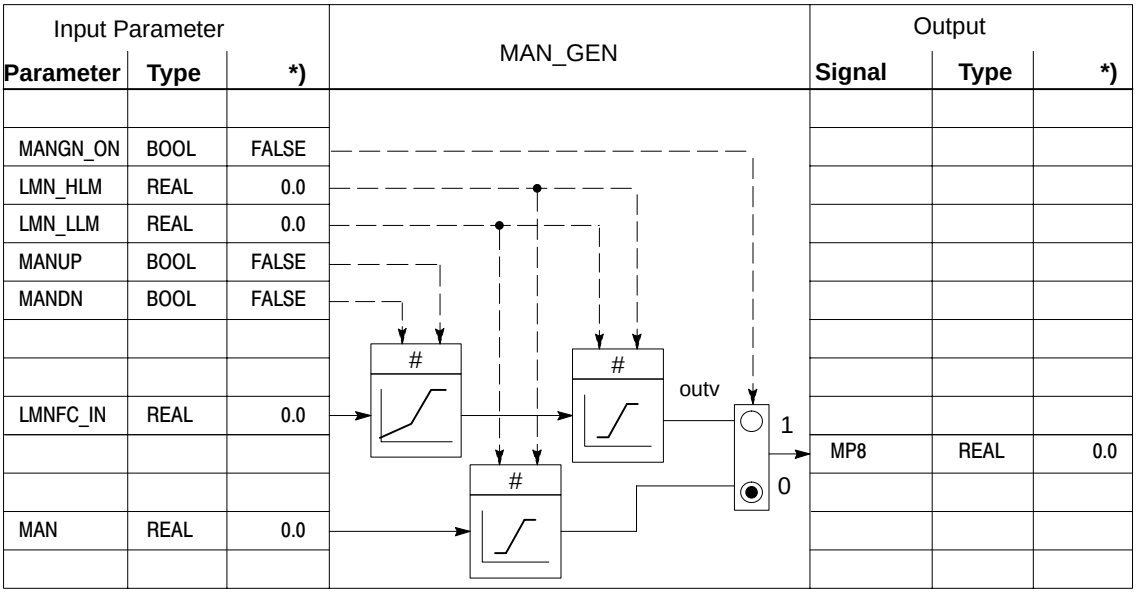
Startup and Mode of Operation of the Setpoint Generator

- During a complete restart, the outv output is reset to 0.0.
- If you then turn on the manipulated value generator (MANGN_ON = TRUE) the signal value LMNFC_IN is first output at the outv output. This means that the changeover to the manipulated value generator from a different mode is always smooth. Providing MANUP or MANDN (up/down switches of the configuration tool) are not activated, LMNFC_IN remains set at the output.

Parameters of the MAN_GEN Function

The output parameter outv is an implicit parameter. It is accessible at measuring point MP8 using the configuration tool.

Parameter	Meaning	Permitted Values
MAN	Manual manipulated value	–100.0 to 100.0 [%]



*) Default when the instance DB is created

Figure 5-4 Functions and Parameters of the Manipulated Value Generator

5.2.3 FC Call in the Manipulated Variable Branch (LMNFC)

Application

If you include a user-defined function (FC) in the manipulated variable branch, you can process the signal of the manipulated variable PID_OUTV generated in the controller (for example setting a signal time lag) before it is connected to the output of the controller.

The LMNFC Function

If you activate the LMNFC function with LMNFC_ON = TRUE, a user-defined function (FC) is called. The number of the FC to be called is entered using the LMNFCNBR parameter.

The controller calls the user FC. Input/output parameters of the user FC are not supplied with values. You must therefore program the data transfer with S7 STL. A programming example is shown below.

STL	Explanation

```
FUNCTION “User FC”
VAR_TEMP
INV:REAL;
OUTV:REAL;
END_VAR
BEGIN
L      “Controller_DB”.LMNFC_IN
T      #INV
       //User function OUTV=f(INV)
L      #OUTV
T      “Controller DB”.LMNFC_OUT
END_FUNCTION
```

The value of LMNFC_ON decides whether a freely programmed function in the form of a standard FC (for example a PT element) is included in the manipulated variable branch at this point or whether the manipulated value is further processed without this form of preprocessing (Figure 2-15).

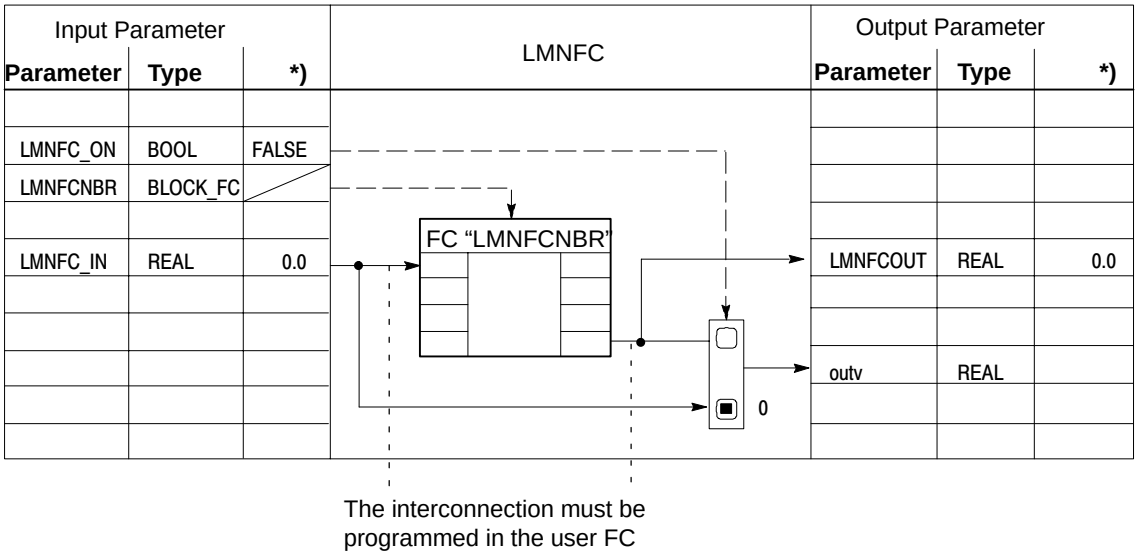
Danger

The block does not check whether an FC exists. If the FC does not exist, the CPU changes to STOP with an internal system error.

Parameters of the LMNFC Function

The LMNFC_IN input value is an implicit parameter. This can be monitored at LMNFC_IN or at measuring point **MP9** using the configuration tool.

The initial value outv ist also an implicit parameter and cannot be monitored via the configuration tool (see Figure 2-15).



*) Default when the instance DB is created

Figure 5-5 FC Call in the Manipulated Value Branch

5.2.4 Limiting the Rate of Change of the Manipulated Value (LMN_ROC)

Application

Ramp functions are used in the manipulated variable branch when step changes in the process input signal are not acceptable for the process. Abrupt changes in the manipulated value must, for example, be avoided when there is gearing between a motor and the load and when a fast rate of change in the speed of the motor would cause overload in the gearing.

The LMN_ROC Function

The LMN_ROC limits the up and down rate of change of the manipulated value at the output of the controller. Starting from zero, two ramps one with ascending and one with descending values can be selected for the entire range of values. The function is activated when LMNRC_ON = TRUE is set.

The limit values for the rate of change of the ramp functions in the positive and negative range of the manipulated variable are entered at the two inputs LMN_URLM and LMN_DRLM. The rate of change is an up or down rate as a percentage per second. Faster rates of change are reduced to these limit rates.

If, for example, 'LMN_URLM' is selected as 10.0 [%/s], the following values are added to the "old" value of outv in each sampling cycle as long as |inv| > |outv|:

Sample time	1 s	$\rightarrow outv_{old}$ + 10 %
	100 ms	$\rightarrow outv_{old}$ + 1 %
	10 ms	$\rightarrow outv_{old}$ + 0.1 %

The following diagram illustrates the way in which the signals are processed (Figure 5-6). The step functions at the inv(t) input become ramp functions at the outv(t) output.

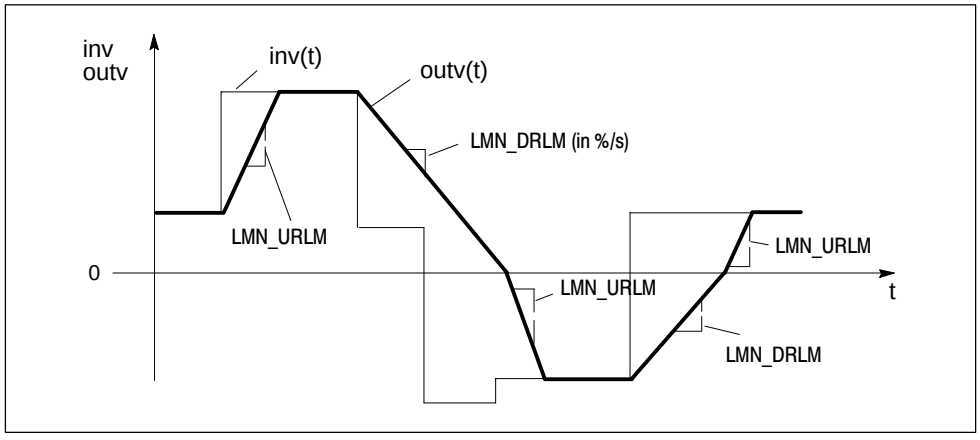


Figure 5-6 Limitation of the Rate of Change of the Manipulated Variable LMN(t)

No signal is output when the rate of change limits are reached.

The ramp parameters are as follows:

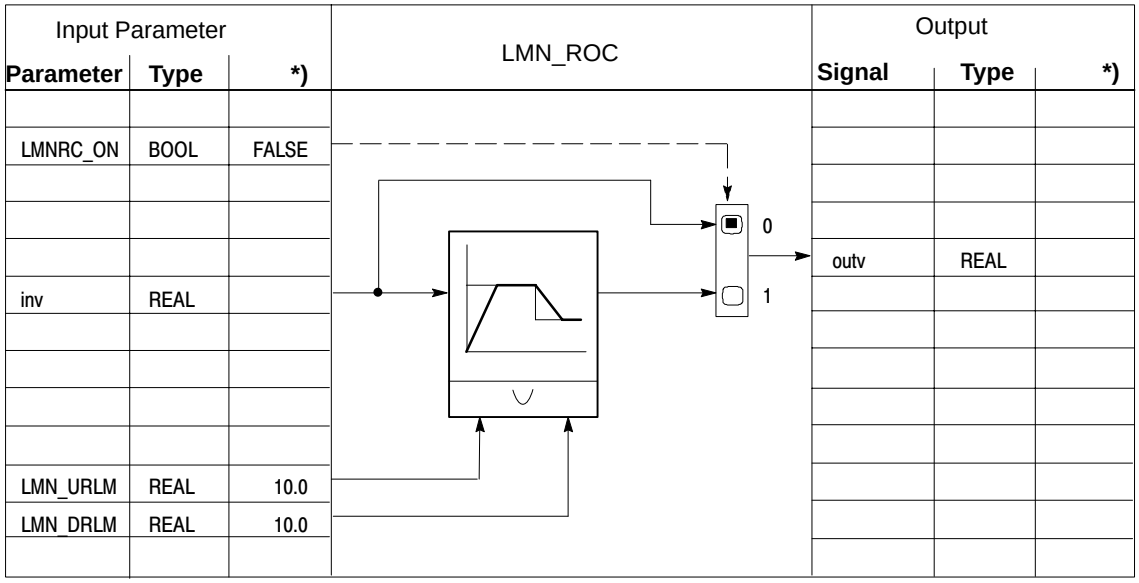
Parameter	Ramp
LMN_URLM	|outv| rising
LMN_DRLM	|outv| falling

Parameters of the LMN_ROC Function

The input value inv and the output value outv are implicit parameters.They cannot be monitored at the configuration tool (Figure 2-15).

Parameter	Meaning	Permitted Values
LMN_URLM	Manipulated value up rate limit	≥ 0 [%/s]
LMN_DRLM	Manipulated value down rate limit	≥ 0 [%/s]

The rates of change (as a percentage per second) are always entered as a positive value.



*) Default when the instance DB is created

Figure 5-7 Functions and Parameters of the Manipulated Value Rate of Change Limits

5.2.5 Limiting the Absolute Value of the Manipulated Variable (LMNLIMIT)

Application

The operating range, in other words the range through which the actuator can move within the permitted range of values, is determined by the range of the manipulated variable. Since the limits for permitted manipulated values do not normally match the 0% or 100% limit of the manipulated value range, it is often necessary to further restrict the range.

To avoid illegal statuses occurring in the process, the range for the manipulated variable has an upper and lower limit in the manipulated variable branch.

The LMNLIMIT Function

The 'LMNLIMIT' function limits the LMN(t) to selected upper and lower values LMN_HLM and LMN_LLM. The input variable inv must, however, be outside these limits. Since the function cannot be disabled, an upper and lower limit must always be assigned during the configuration.

The numerical values of the limits (as percentages) are set at the input parameters for the upper and lower limits. If these limits are violated by the input variable inv(t), this is indicated at the signaling outputs (Figure 2-15).

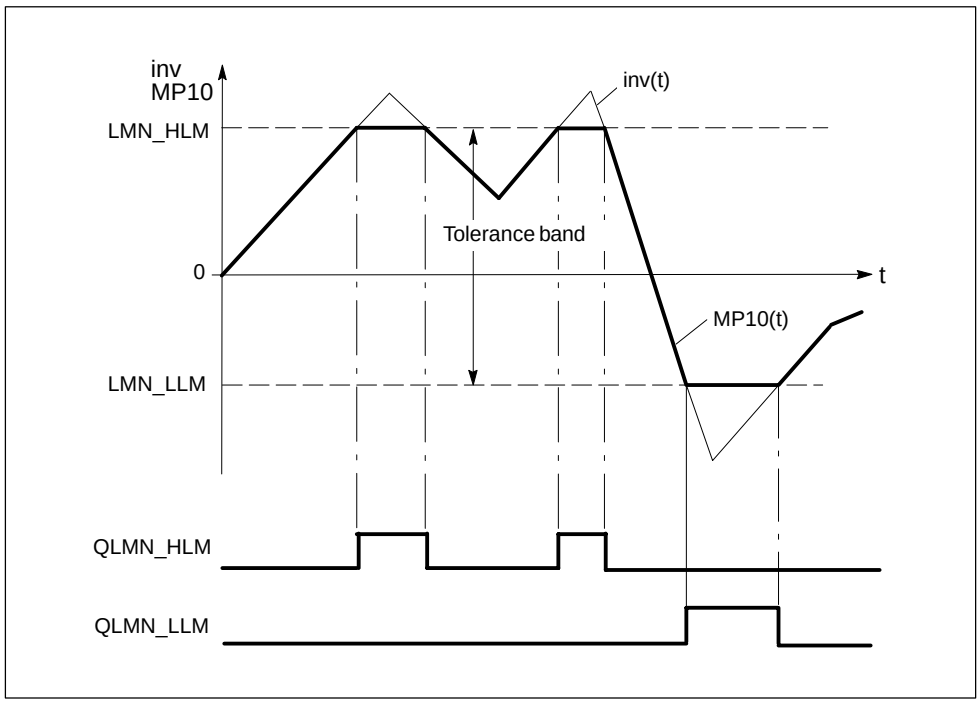


Figure 5-8 Absolute Value Limits of the Manipulated Variable LMN(t) = MP10 (t)

Start Up and Mode of Operation

- In case of a complete restart all the signal outputs are set to zero.
- The limitation operates as shown in the following table:

LMN =	QLMN_HLM	QLMN_LLM	when:
LMN_HLM	TRUE	FALSE	inv ≥ LMN_HLM
LMN_LLM	FALSE	TRUE	inv ≤ LMN_LLM
inv	FALSE	FALSE	LMN_HLM < inv < LMN_LLM

The effective manipulated value of the controller is indicated at the output (parameter LMN) and at measuring MP10.

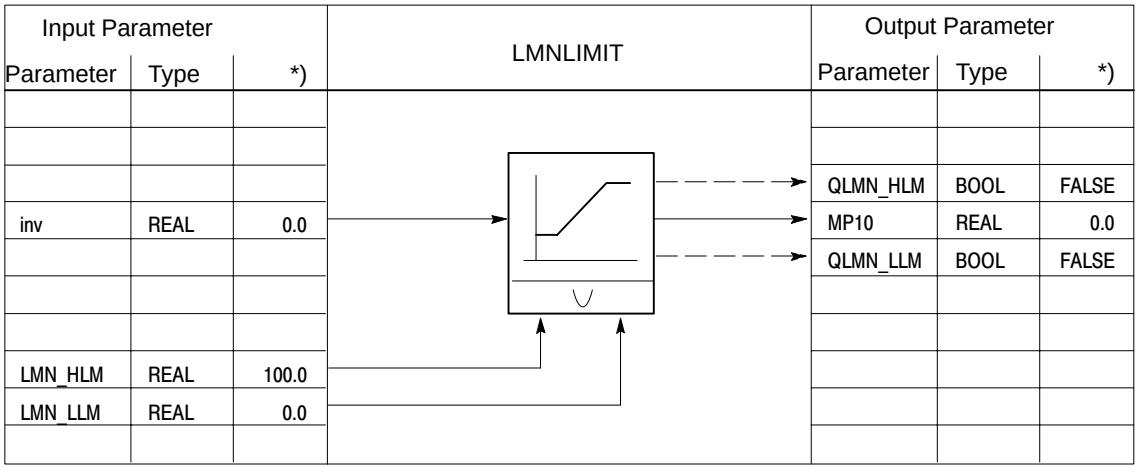
Parameters of the LMNLIMIT Function

The input value inv is an implicit parameter. It is only accessible at the parameter LMNFC_IN or at measuring point **MP9** using the configuration tool.

For the limitation function to operate properly, the following must apply:

LMN_HLM > LMN_LLM

Parameter	Meaning	Permitted Values
LMN_HLM	Upper limit of the man. variable	LMN_LLL ... 100.0 [%]
LMN_LLM	Lower limit of the man. variable	–100.0 ... LMN_HLM [%]



*) Default when the instance block is created

Figure 5-9 Functions and Parameters of the Absolute Value Limitation of the Manipulated Value

5.2.6 Normalization of the Manipulated Variable to the Format of a Physical Variable (LMN_NORM)

Application

If the manipulated variable applied to the input of the process must be a physical dimension, the floating point values in the range 0 to 100% must be converted to the physical range (for example 150 to 3000 rpm) of the manipulated variable.

The LMN_NORM Function

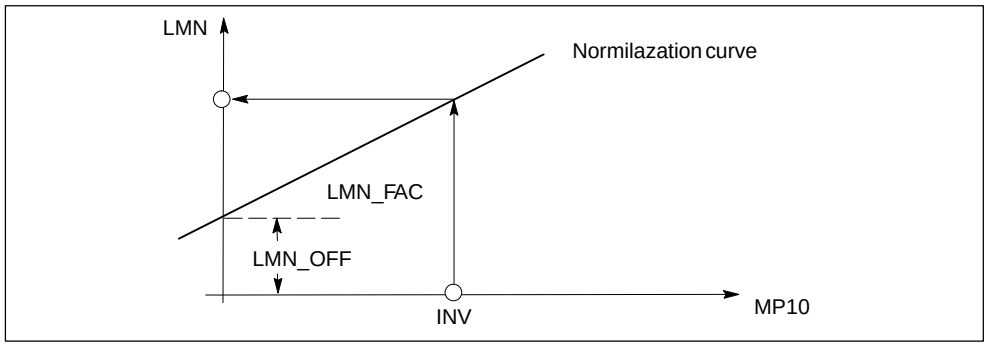
The LMN_NORM function converts the analog output variable of the controller. The analog manipulated value is converted to the output value LMN using the normalization curve. The output value can be monitored at parameter **LMN** using the configuration tool.

To obtain the normalization curve:

internal percentage value (in REAL-Format) ⇒ External physical values

two parameters must be defined:

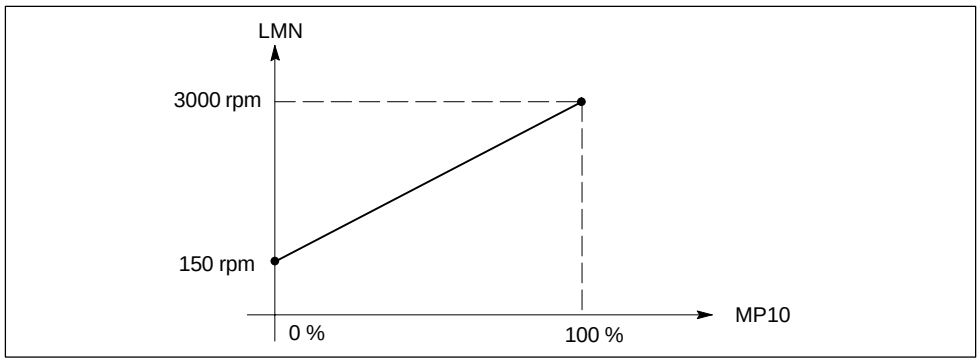
- the factor (for the slope): **LMN_FAC**
- the offset of the normalization curve from zero: **LMN_OFF**



The normalization value is calculated from the respective input value MP10:

LMN = MP10 * LMN_FAC + LMN_OFF

The following applies for the above example:



$$\text{LMNFAC} = \frac{(3000 - 150)\text{rpm}}{(100 - 0)\%} = 28,5\frac{\text{rpm}}{\%}$$

LMNOFF = 150 rpm

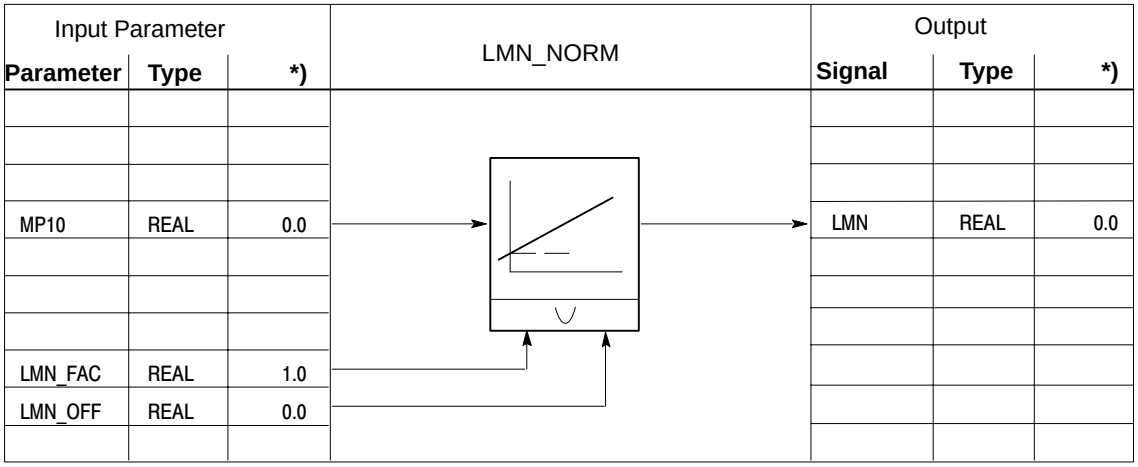
Internally, the function does not limit any values and the parameters are not checked.

Parameters of the LMN_NORM Function

The output is an implicit parameter and can be monitored at **LMN** using the configuration tool (Figure 2-15).

To define the slope used to convert the value to the physical variable at the LMN output, the parameter LMN_FAC can be selected throughout the entire technical range of values.

Parameter	Meaning	Permitted Values
LMN_FAC	Manipulated value factor (slope of the normalization curve)	Entire range of values (no dimension)
LMN_OFF	Manipulated value offset	Techn. range of values (physical value)



*) Default when the instance DB is created

Figure 5-10 Functions and Parameters for Manipulated Value Normalization to a Physical Value

5.2.7 Manipulated Value Output in the Peripheral Format (CRP_OUT)

Application

If the manipulated value is transferred to an analog output module, the numerical value of the internal manipulated variable in floating point format (as a percentage) must be converted to the numerical value of the data word connected to the output LMN_PER. This task is performed by the **CRP_OUT** function.

The CRP_OUT Function

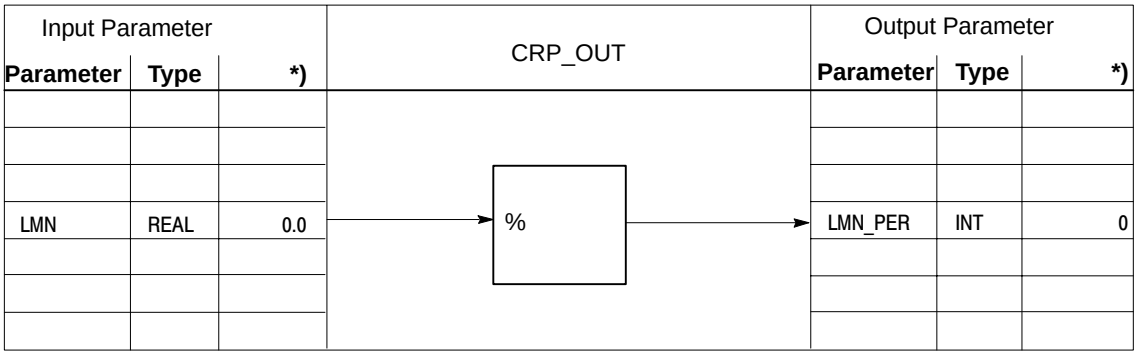
The CRP_OUT function sets the floating point value of the manipulated variable at input LMN to a value converted to the peripheral format. There is no check for positive or negative overflow or over/underdrive. Module types are not taken into account.

The following table provides an overview of the ranges and numerical values before and after processing by the normalization algorithm of the CRP_OUT function.

Manipulated Value LMN in %	Peripheral value LMN_PER
118,515	32767
100,000	27648
0,003617	1
0,000	0
–0,003617	–1
–100,000	–27648
–118,519	–32768

Parameters of the CRP_OUT Function

The input value is an implicit parameter in the floating point format. This can be monitored at output LMN using the configuration tool.



*) Default when the instance block is created

Figure 5-11 Functions and Parameters of Manipulated Variable Conversion to the Peripheral Format

5.2.8 Influencing the Manipulated Value With the Configuration Tool

LMN Display and Setting in the Loop Monitor

The configuration tool has its own interface to the controller FB. It is therefore always possible to interrupt the manipulated variable branch and to specify a manipulated value LMN_OP (for example for test purposes when working on a PG/PC on which the configuration tool is loaded) (Figure 5-12).

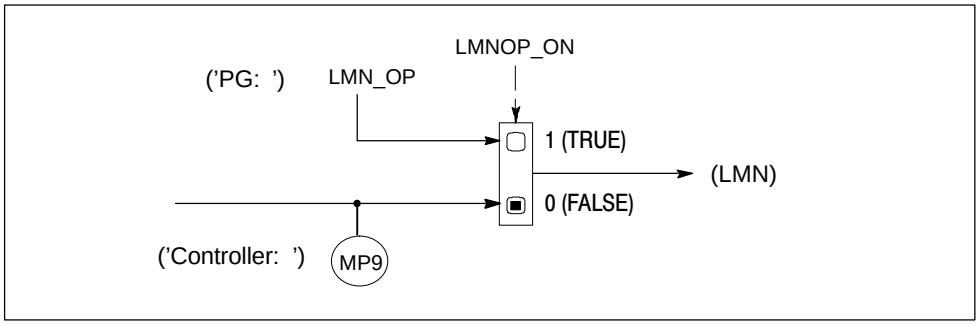


Figure 5-12 Interventions in the Manipulated Variable Branch Using the Configuration Tool

One of the identical three panels in the window of the **loop monitor** is available for this purpose and is labeled **manipulated value**. Here the manipulated valuecurrently applied to measuring point MP9 is displayed in the “Controller:” field. The field below this ('PG: is used to display and change the parameter LMN_OP.

Changeover to the Manipulated Value Specification by 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 and LMN_OP is enabled to the manipulated value in the controller FB.

If the rate of change limitation LMN_ROC is active in the manipulated variable branch, the change from switch settings “PG: ” and ”Controller:” is smooth without any step change The value adopted with the changeover (MP9) can be seen in the “Controller:” display field of the **loop display**. LMN is returned to this value at a speed dictated by the rate of change limit LMN_ROC.

These interventions only affect the process when they are sent to the programmable logic controller by clicking the “Send” button in the **loop monitor**.

5.3 Continuous Controller in Cascade Control

Interrupting the Controller Cascade

In a cascade, several controllers are directly dependent on each other. You must therefore make sure that if the cascade structure is interrupted at any point, the cascade operation can be resumed without causing any problems.

In the secondary or slave controllers of a cascade control system, a QCAS signal is formed by ORing the status signals of the switches in the setpoint and manipulated variable branches. This signal operates a switch in the secondary controllers that changes the controller to the correction mode. The correction variable is always the process variable PV of the secondary loop (Figure 5-13).

The switch from correction mode to automatic mode smoothy just as it is done when switching from manual mode to automatic mode.

The continuous controller (PID_CP) can be used as the primary controller in cascade control systems or as the secondary controller in slave loops.

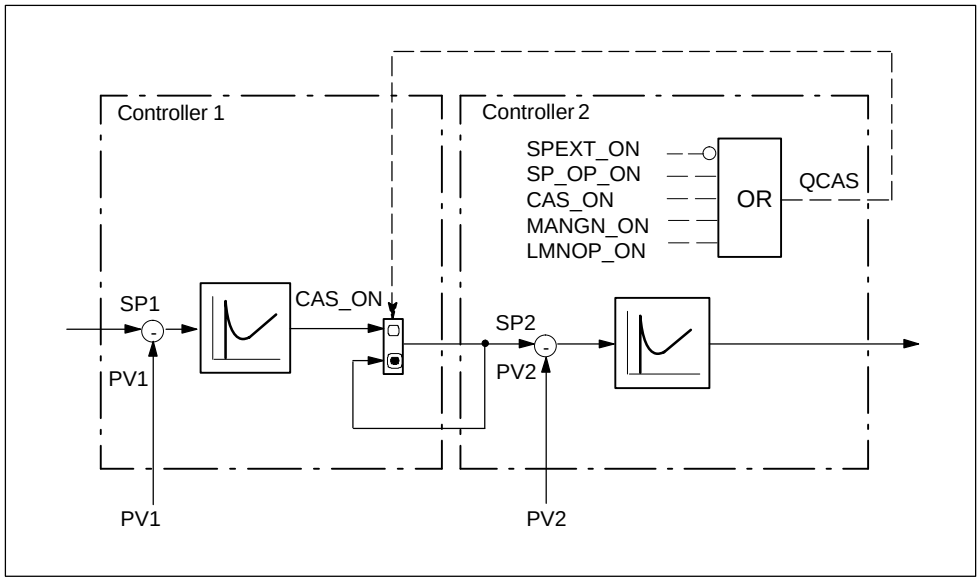


Figure 5-13 Two-Loop Cascade Control System

Note

The interconnection of the manipulated value of the master controller LMN must always go to the external setpoint value SP_EXT of the secondary controller.

Block Connections

The following diagram illustrates the principle of the controller or block connections in multi-loop cascades.

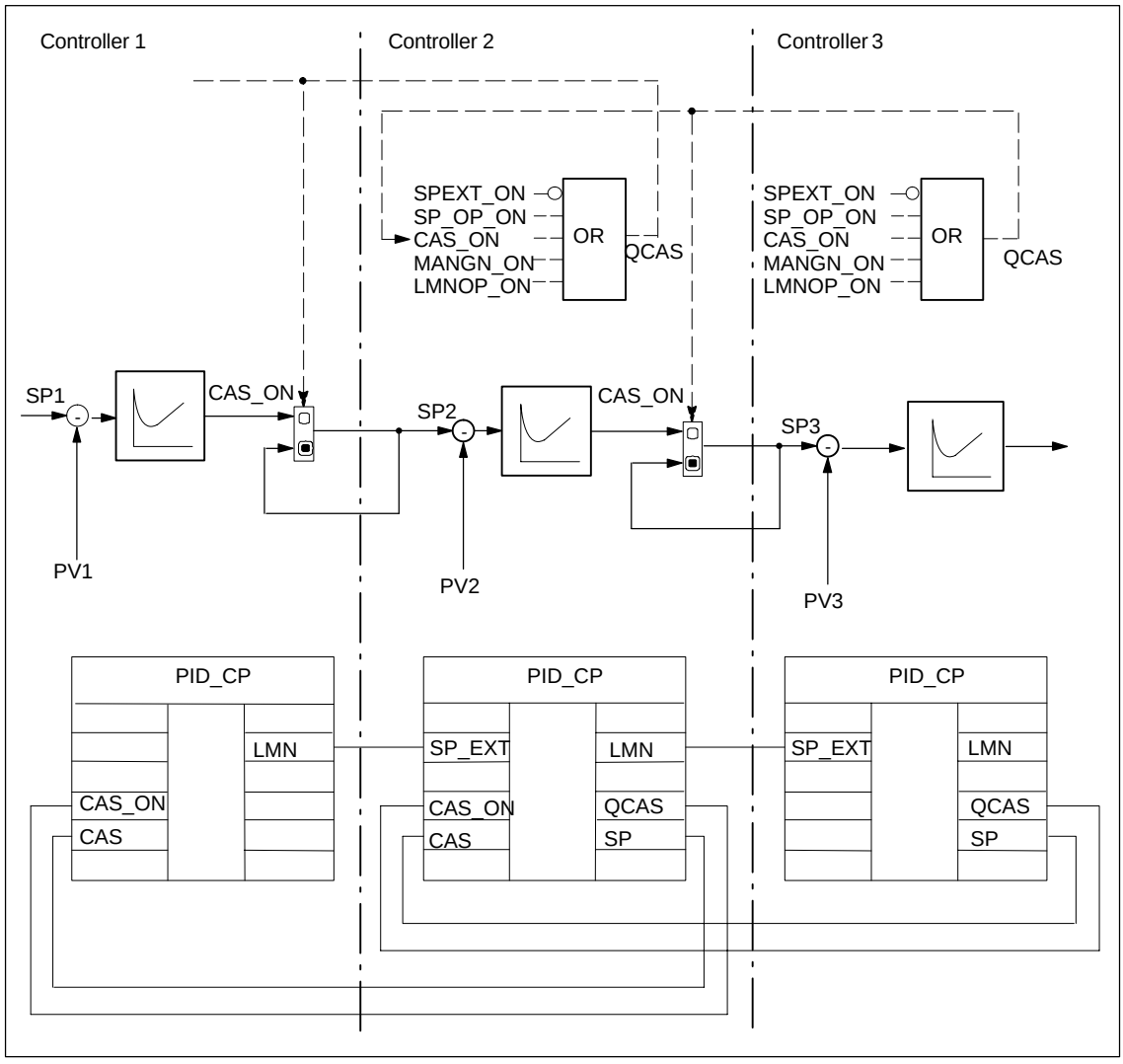


Figure 5-14 Connecting a Cascade With Two Slave Control Loops

5.4 Pulse Generator Module (PULSEGEN)

Application

The pulse genmeration function generates the pulse output of a continuous controller so that proportional actuators can be controlled by pulses using the Standard PID Control. This allows PID two-step and three-step controllers to be implemented with pulse duration modulation.

The Pulse Generator

The pulse generator module of the standard FB “PIC_CP” transforms the input variable “setpoint of the PID controller at the measuring point MP 10” by modulating the pulse width into a pulse sequence with a constant period time, which has to be configured in PER_TM.

The duration of a pulse per period is proportional to the input value. The cycle set by PER_TM is not identical to the processing cycle of the pulse generator. A PER_TM cycle consists of several processing cycles of the pulse generator and the number of pulse generator calls per PER_TM cycle is a measure of the accuracy of the pulse duration.

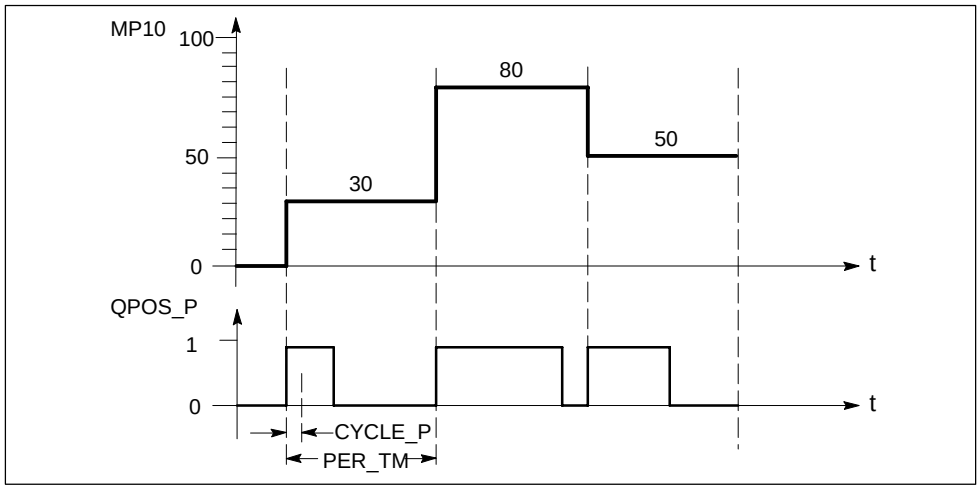


Figure 5-15 Pulse Duration Modulation

An input variable of 30% and ten calls of the puzlse generator every PER_TM cycle mean:

- ”One” at the output QPOS for the first three calls of the pulse generator (30 % of 10 calls),
- ”Zero” at the output QPOS for seven further calls of the pulse generator (70 % of 10 calls).

Controller Sampling Time CYCLE and Pulse Code Width CYCLE_P

If you have activated the pulse generation module (PULSE_ON = TRUE), you must first specify the clockec ontrol of the calling watchdog-interrupt OB at the input CYCLE_P. The duration of the generated pulse always amounts to a integer multiple of this value.

Specify the sampling time for the remaining control functions of the PID_CP at the CYCLE input. The function block PID_CP determines the time pulse scaling and processes the controlling functions with the sampling time CYCLE.

You must ensure that CYCLE is an integer multiple of CYCLE_P. If you do not observe this condition, the function block PID_CP rounds the sampling time for the controller functions to an integer multiple of CYCLE_P. The time-dependent functions (for examle. smoothing, integration, differentiation) are not executed properly then.

CYCLE can be selected smaller than the period time PER_TM_P or PER_TM_N. This is advisable if on the one hand a large period time is desired in order not to wear the final controlling element unnecessarily and if on the other hand the sampling time has to be small due to a rapid process.

An advisable value for the sampling time CYCLE is, as for the continuous controller without a pulse generator mode, that CYCLE may not be smaller than 10% of the dominating process constant of the controlled system.

Example for the effect of the parameter CYCLE_P, CYCLE and PER_TM_P or PER_TM_N:
PER_TM_P = 10 s, CYCLE = 1 s, CYCLE_P = 100 ms.
A new manipulated value is calculated every second. The comparison of the manipulated value with the currently output pulse length or break length is carried out every 100 ms.

If a pulse is output and the calculated manipulated value is greater than the currently output pulse length / PER_TM_P, the pulse is extended. Otherwise no further pulse signal is output. If no pulse is output and (100% – the calculated manipulated value) is greater than the currently output break length / PER_TM_P, the break is extended. Otherwise a pulse signal is output.

Due to a particular process of the pulse generation an increase or decrease of the manipulated variable during the period causes an increase or decrease of the output pulse. If, in this case (CYCLE < PER_TM_P), the period time is configured so large that it would cause oscillation of the actual value, the effective period time is reduced to a sensible value by the function block PID_CP.

Accuracy of the Pulse Generation

The smaller the pulse code width CYCLE_P compared to the period time PER_TM_P (or PER_TM_N), the more accurate the pulse width modulation. If order to achieve a sufficiently accurate control, the following equation should apply

$$\text{CYCLE\_P} \leq \frac{\text{PER\_TM}}{20 \ldots 50}$$

Implementation of Very Short Pulse Code Widths

In the case of a very rapid process very small pulse code widths (for example 10 ms) are required. Due to the program execution time it does not make sense to process the controlling sections in the same watchdog-interrupt OB as the calculation of the pulse output. Move the processing of the control functions either to the OB 1 or into a slower watchdog interrupt OB (processing of the control function in the OB 1 is only advisable when the scan time of the OB 1 is clearly smaller than the sampling time CYCLE of the controller).

Use the parameter SELECT to specify which program section is to be processed. The following table provides you with an overview of the configuration of the input parameter SELECT:

SELECT	Used functionality of the block	Method on which it is based
0	Control section and pulse output	Control section and pulse output in one and the same block
1	Call on OB1 (control section)	Control section in OB1, pulse output in the rapid watchdog interrupt OB
2	Call in the watchdog-interrupt OB (pulse output)	
3	Call in the slow watchdog interrupt OB (control section)	Control section in the slow watchdog-interrupt OB, pulse output in the rapid watchdog interrupt OB
2	Call in the rapid watchdog-interrupt OB (pulse output)	

The following passages explain the methods indicated in the above table for realizing very short pulse code widths in more detail.

- Control function in OB1, pulse output in the watchdog interrupt OB

  When the FB “PID_CP” is called with SELECT = 2, the calculation of the pulse output and the check whether the sampling time configured at CYCLE has expired since the last processing of the control are carried out.If this sampling time has expired, the FB writes the value TRUE to the variable QC_ACT in the instance DB.

  When the FB “PID_CP” is called with SELECT = 1, the evaluation of the variable QC_ACT in the instance DB is carried out as follows: If QC_ACT has the value FALSE, the block is terminated immediately. It thus has only require a very brief run time. If QC_ACT has the value TRUE, the control section is processed once and then the FB resets the QC_ACT.

  This procedure has the effect that the sampling time for the control function of the FB “PID_CP” cannot be observed exactly. It fluctuates around the run time of the OB1 (including all interrupts). This process is therefore only suitable if the run time of the OB1 is small in comparison to the CYCLE sampling time.

- Control function in the slow watchdog-interrupt OB, pulse output in the rapid watchdog interrupt OB

  When the FB “PID_CP” is called with SELECT = 2, the pulse output is always calculated.

  When the FB “PID_CP” is called with SELECT = 3, the control section is always processed.

Note

It is advisable to program the call of the FB “PID_CP” with its multitude of formal operands only once in an FC, and not twice completely, once in an FC which also has the parameter SELECT as its input parameter. This input parameter is then interconnected to the SELECT input of the FB “PID_CP”. Only this FC is then called in the OB1 or in the watchdog-interrupt OB.

This procedure is advisable and economizes your program memory.

Modes of the Controller With Pulse Output

Depending on the assignment of parameters for the pulse generator, PID controllers with three-step, with a bipolar or monopolar two-step output can be configured. The following table shows the settings of 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 Controller

In the “three-step controller” mode, the actuating signal can have three states, 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 status values of the binary output signals QPOS_P and QNEG_P are assigned to the respective operating states of the final controlling element. The table shows two examples.

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

Suitably dimensioning the minimum pulse or minimum break time P_B_TM can prevent extremely short on and off times that can greatly reduce the working life of actuators and control elements (Figure 5-16). To achieve this, a response threshold is set for pulse output.

Note

Small absolute values in the input variable “setpoint of the PID controller at the measuring point MP 10” that would generate a pulse duration less than P_B_TM_P are suppressed. For large input values that would generate a pulse duration greater than PER_TM_P – P_B_TM_P, a pulse duration of 100% or –100% is set.

A setting of P_B_TM_P $\leq$ 0.1 * PER_TM_P is recommended.

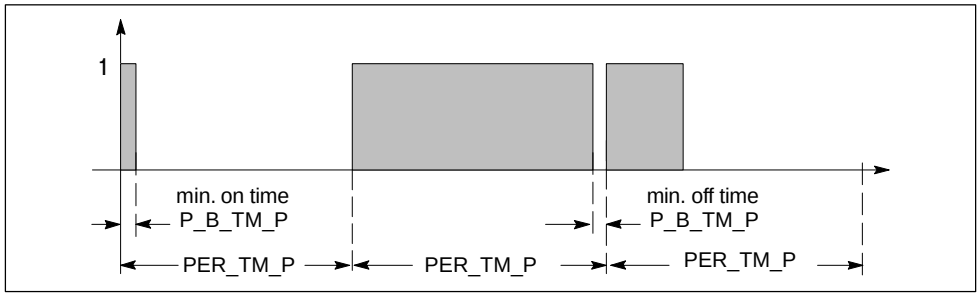


Figure 5-16 How the Pulse Output Switches On and Off

The duration of the positive or negative pulses can be calculated from the input variable “setpoint of the PID controller at the measuring point MP 10” (as a percentage) multiplied by the period:

$$\text{Pulse duration} = \frac{MP10}{100} * \text{PER\_TM\_P[s]}$$

If the minimum pulse or break time is suppressed, the conversion characteristic curve develops “dog legs” at the start and end of the range (Figure 5-17).

The statements above apply for P_B_TM_N and PER_TM_N (see Figure 5-17).

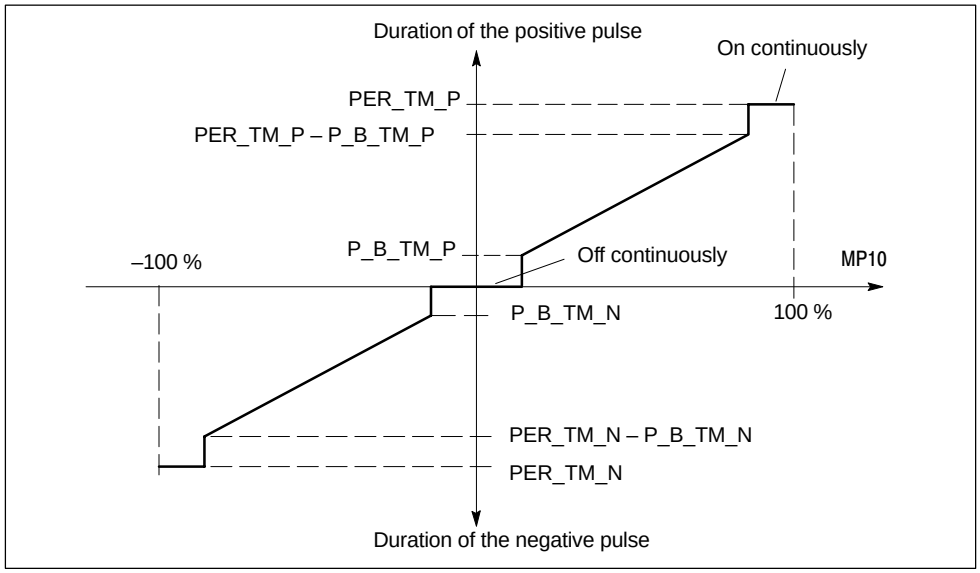


Figure 5-17 Symmetrical Curve of the Three-Step Controller (Ratio Factor = 1)

Asymmetrical Three-step Controller

Using the ratio factor RATIOFAC, the ratio of the duration of positive and negative pulses can be changed. In a thermal process, this, for example, allows different system time constants to be taken into account for heating and cooling.

If, at the same absolute value for the input variable “setpoint of the PID controller at the measuring point MP 10”, the pulse duration of the negative pulse output must be shorter than the positive pulse, a ratio factor less than 1 must be set (Figure 5-18):

pos. pulse > neg. pulse: RATIOFAC < 1

Pulse duration negative: $\frac{MP10}{100} * PER\_TM\_N * RATIOFAC$

Pulse duration positive: $\frac{MP10}{100} * PER\_TM\_P$

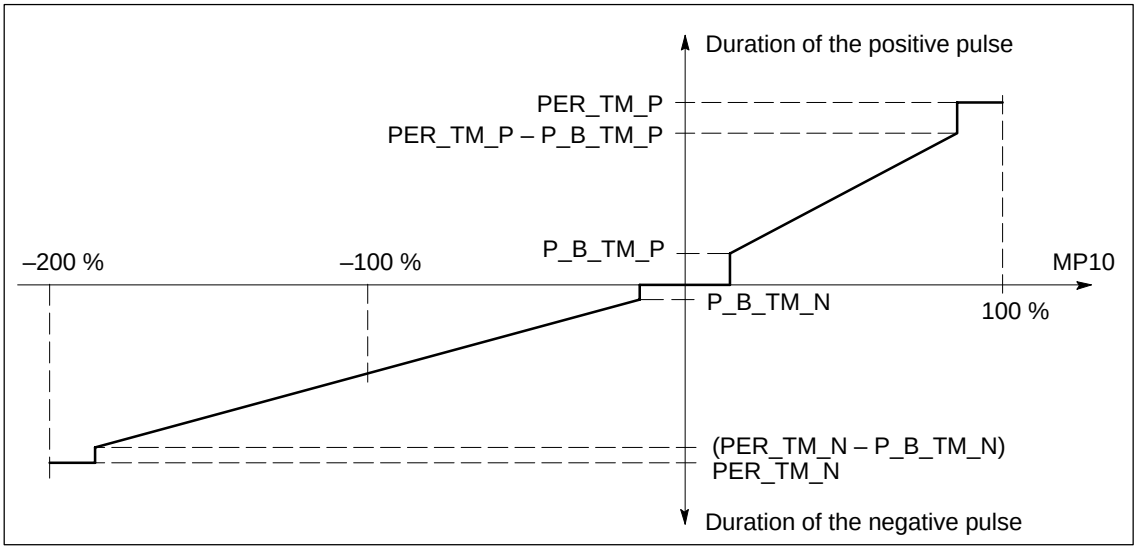


Figure 5-18 Asymmetrical Curve of the Three-Step Controller (Ratio Factor = 0.5)

If, on the other hand, with the same absolute value |MP10|, the pulse duration at the positive pulse output must be shorter than that of the negative pulse, a ratio factor greater than 1 must be set:

pos. pulse < neg. pulse: RATIOFAC > 1

Pulse duration negative: $\frac{MP10}{100} * PER\_TM\_N$

Pulse duration positive: $\frac{MP10 * PER\_TM\_P}{100 * RATIOFAC}$

Mathematically, this means that at RATIOFAC < 1, the response value for negative pulses is multiplied by the ratio factor and at RATIOFAC > 1, the response value for positive pulses is divided by the ratio factor.

Note

You have to adjust the manipulated value limits the following formulae for an assymetrical three-step controller RATIOFAC ≠ 1:

RATIOFAC < 1:

LMN_HLM = 100

LMN_LLM = –100 * (1 / RATIOFAC)

RATIOFAC > 1:

LMN_HLM = 100 * RATIOFAC

LMN_LLM = –100

Examples:

RATIOFAC = 1	RATIOFAC = 0,5	RATIOFAC = 2,0
LMN_HLM = 100	LMN_HLM = 100	LMN_HLM = 200
LMN_LLM = –100	LMN_LLM = –200	LMN_LLM = –100

Two-step Controller

In a two-step controller, only the positive pulse output QPOS_P is connected to the corresponding on/off element by PIC_CP. Depending on the range being used (MP10 = -–100.0 to 100.0% or MP10 = 0.0% to 100.0%), the two-step controller can have either a bipolar or a monopolar range (Figure 5-19 and Figure 5-20).

In the monopolar mode MP10 can only have values between 0.0 and 100%.

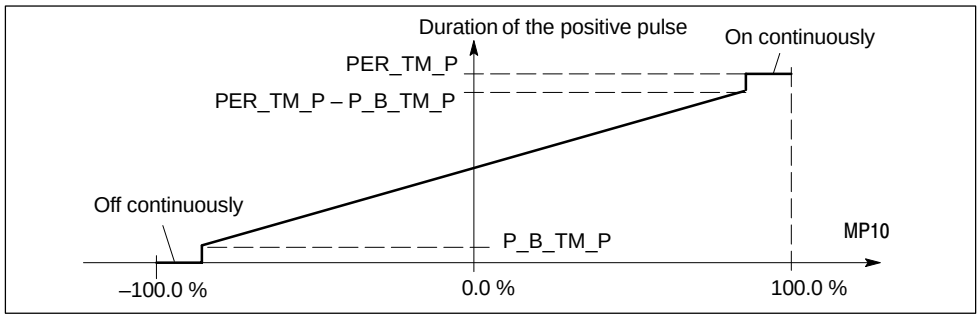


Figure 5-19 Two-Step Controller With Bipolar Range (-100% to 100%)

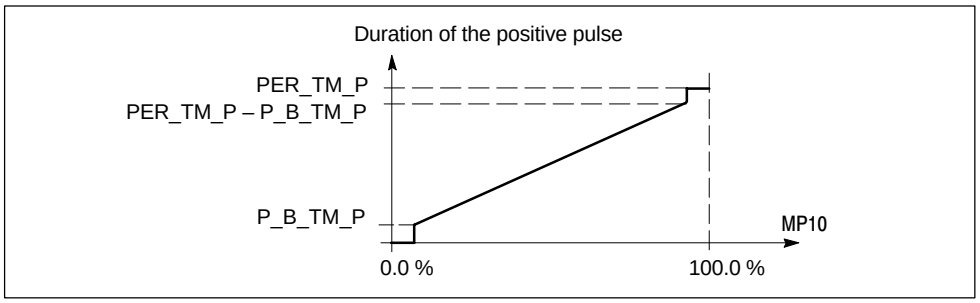


Figure 5-20 Two-Step Controller With a Monopolar Range (0% to 100%)

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 actuator pulses.

	On	Off
QPOS_P	TRUE	FALSE
QNEG_P	FALSE	TRUE

Parameters of the Pulse Generation Module

The values of the input parameters are not limited at the block ”PID_CP”. The parameters are not checked.

During a complete restart, all the parameters are set to zero.

Parameter	Meaning	Permitted values
CYCLE_P	Sampling time of the pulse generation module	≥ 1 ms
SELECT	Selection switch for the function sections to be processed in the current block call (only relevant, if PULS_ON = TRUE) 0 (default): PID and pulse generator 1: PID (block call in OB1) 2: Pulse generator (block call in watchdog-interrupt OB) 3: PID (block call in watchdog-interrupt OB)	
QC_ACT	Display whether the control part is processed at the next block call (only relevant if SELECT has the value 0 or 1)	
QPOS_P	Pulse generator Positive pulse on	
QNEG_P	Pulse generator Negative pulse on	
PULSE_ON	Pulse generator on	
STEP3_ON	Pulse generator Three-step control on	
ST2BI_ON	Pulse generator: Activate two-step control for bi-polar manipulating range (for monopolar manipulating range STEP3_ON = FALSE)	

Parameter	Meaning	Permitted values
PER_TM_P	Pulse generator Period time of the positive pulse	
PER_TM_N	Pulse generator Period time of the negative pulse	
P_B_TM_P	Pulse generator Minimum pulse or minimum break time of the positive pulse	
P_B_TM_N	Pulse generator: Minimum pulse or minimum break time of the negative pulse	
RATIOFAC	Ratio factor for asymmetrical curves	0.1 ... 10.0 (no dimension)

Input Parameter			Pulse generating module of “PID_CP”	Output Parameter		
Parameter	**Type**	***)**		**Signal**	**Type**	***)**
COM_RST	BOOL	FALSE				
CYCLE	TIME	T#10ms				
SELECT	BYTE	0				
PULSE_ON	BOOL	FALSE				
STEP3_ON	BOOL	TRUE				
ST2BI_ON	BOOL	FALSE				
			#			
				QC_ACT	BOOL	TRUE
MP10	REAL	0.0		QPOS_P	BOOL	FALSE
				QNEG_P	BOOL	FALSE
CYCLE_P	TIME	T#10ms				
PER_TM_P	TIME	T#1s				
PER_TM_N	TIME	T#1s				
P_B_TM_P	TIME	T#50ms				
P_B_TM_N	TIME	T#50ms				
RATIOFAC	REAL	1.0				

*) Default when the instance DB is created

Figure 5-21 Functions and Parameters of the Pulse Generator

The Step Controller (PID_ES) 6

6.1 Control Functions of the PID Step Controller

The PID_ES Function Block

Apart from the functions in the setpoint and process variable branch, the function block (FB2) also implements a complete PID controller with a binary manipulated value output. It is possible to adjust the manipulated value manually. Subfunctions can be enabled of disabled.

With the FB, it is possible to control technical processes and systems with integrating actuators on SIMATIC S7 programmable logic controllers. The controller can be used as a fixed setpoint controller singly or in secondary control loops in cascade, blending or ratio control systems, however it cannot be used as a primary or master controller.

The processing of the signals in the setpoint and process variable branches and the processing and monitoring of the error signal is identical to that of the continuous controller. Detailed descriptions of these functions for both controllers can be found in *Chapter* 4 of this manual.

Normalization of the Input Variables ER and PV

The input variables ER and PV of the PID controller are normalized before controller processing to the range of 0 to 100 in accordance with the following equations:

- If the square-root function is de-activated (SQRT_ON = FALSE):
  - $ER_{Normalized}$ = ER * 100.0 / (NM_PVHR – NM_PVLR)
  - $PV_{Normalized}$ = (PV – NM_PVLR) * 100.0 / (NM_PVHR – NM_PVLR)
- If the square-root function is activated (SQRT_ON = TRUE):
  - $ER_{Normalized}$ = ER * 100.0 / (SQRT_HR – SQRT_LR)
  - $PV_{Normalized}$ = (PV – SQRT_LR) * 100.0 / (SQRT_HR – SQRT_LR)

This normalization is carried out so that the gain factor GAIN of the PID controller can be entered without dimensions. If the upper and lower limit of the physical measuring range changes (for example from bar to mbar), the gain factor then does not have to be changed.

The normalized input variables $ER_{Normalized}$ and $PV_{Normalized}$ cannot be monitored.

Outline of the Functions of the Step Controller With Position Feedback Signal in the Control Loop

The mode of operation of the step controller with a position feedback signal is based on the PID control algorithm and is supplemented by the function elements for generating the binary output signals (Figure 6-1).

The three-step element changes deviations between the manipulated variable and a position feedback signal depending on the sign into positive or negative pulses for the output signal, that can then be transferred to a motorized valve drive. In practical terms, this represents a cascade control with a secondary position control loop.

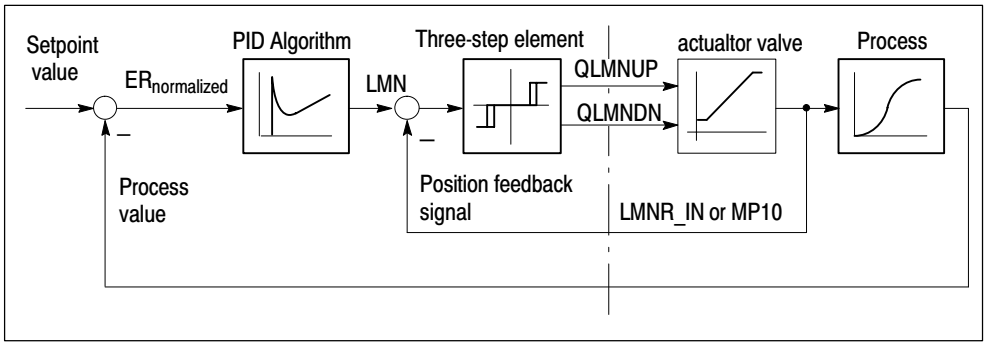


Figure 6-1 Step Controller With a Position Feedback Signal

Outline of the Functions of the Step Controller Without Position Feedback Signal

The I action of the step controller without a position feedback signal is calculated in an integrator in the feedback path. The feedback signal compared with the LMN controller output of the PD controller is derived from the indirectly acquired valve position.

- Signal elements for the simulated position feedback: $\frac{\pm 100.0}{\text{MTR\_TM}}$
- Signal elements for the I action:

  $(\text{setpoint value} - \text{process value})_{\text{normalized}} * \frac{\text{GAIN}}{\text{TI}}$

The feedback signal is thus the difference between the simulated position feedback and the I action.

The three-step element converts deviations between the manipulated variable and feedback variable depending on the sign into positive or negative pulses for the output signal, that can, for example, be transferred to a motor-driven valve.

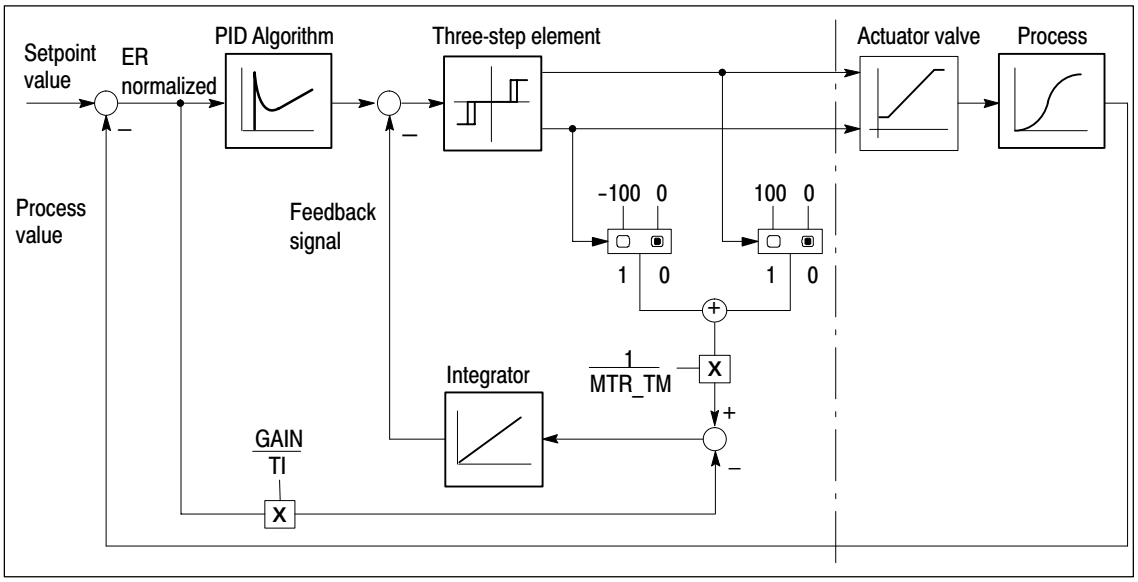


Figure 6-2 Step Controller Without Position Feedback Signal

Complete Restart/Restart

The FB PID_ES function block has an initialization routine that is run through when the input parameter COM_RST = TRUE is set.

Ramp soak (RMP_SOAK)

When the ramp soak is activated, the time slices DB_NBR PI[0 to NBR_PTS].TMV are totalled between the coordinates and indicated at the total time T_TM and total time remaining RT_TM outputs.

If PI[n].TMV is modified on-line or if TM_CONT and TM_SNBR are set, the total time and total time remaining of the ramp soak also change. Since the calculation of T_TM and RS_TM greatly increases the processing time of the RMP_SOAK function when a large number of time slices are involved, this calculation is only performed after a complete restart or when TUPDT_ON = TRUE is set.

Integral action (INT)

When the controller starts up, the integrator is set to the initialization value I_ITLVAL. When it is called by a cyclic interrupt, it starts at this value.

All other outputs are set to their default values.

6.2 Manipulated Variable Processing on the Step Controller With Position Feedback Signal

6.2.1 Modes of the Step Controller

Structure of the Step Controller

The step controller (PID_ES) with a position feedback signal consists of two parts: the controller section working with the continuous signals that is largely identical to the structure of the PID_CP function block and a second part in which the binary actuating signals are generated and in which a position control loop is formed using the position feedback signal (Figure 6-3).

The output of the PID algorithm acts as a reference input for the position controller and therefore specifies the position of the motor-driven actuator.

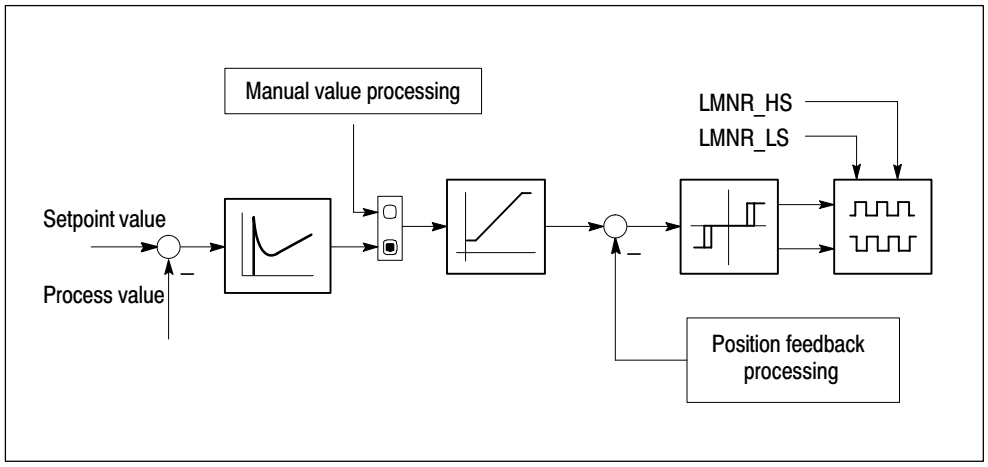


Figure 6-3 Step controller with position feedback signal

To avoid the motor being overloaded, its limit stop signals (LMNR_HS/LMNR_LS) can be used to interlock the controller outputs (Figure 2-16). If the actuator drive does not provide limit stop signals, the input parameters LMNR_HS and LMNR_LS = FALSE must be set.

Note

If no limit stop signals exist, the controller cannot detect whether or not a mechanical limit stop has been reached. It is possible that the controller then outputs signals, for example, to open the valve although it is already fully open.

Operating Modes of the Step Controller

- **Selection: Step controller with position feedback signal**

  Whenever a position feedback signal is available with the type of actuator being used, the controller structure as shown in Figure 6-3 is activated by setting LMNR_ON = TRUE.

  If no position signal can be received from the motor-driven actuator, the step controller structure without a position feedback signal must be selected by setting LMNR_ON = FALSE (see Section 6.3).

Note

The mode selector switch LMNR_ON must not be used when the controller is in the on-line mode.

- **Operating modes**

  The step controller can be operated in the same modes as the continuous controller, in other words in the “automatic” mode using a closed loop and in the “manual” mode where the actuator is driven manually in the open loop. The option of generating manual signals by entering an absolute value (MAN) or using the manipulated value generator (MAN_GEN) is extended with the step controller by the possibility of switching the output signals using LMNS_ON.

Automatic Mode

If MAN_ON = FALSE is selected, the manipulated value of the PID algorithm is switched to the three-step element. The changeover from manual to automatic produces a step change in the manipulated value LMN. This does not have a detrimental effect, however, since the process is driven using the integrating actuator (a ramp change is produced). The output of the PID algorithm is applied to measuring point MP7.

Manual Value Tracking in Automatic Mode

In automatic mode the I/O parameter MAN of the position of the actuator (LMNR_IN, if LMNRP_ON = FALSE or MP10, if LMNRP_ON = TRUE) is tracked. When you change over to manual mode the manipulated variable therefore remains at the value which corresponds to the position oif the actuator. It can onlyx be changed by operator control.

Manual Mode

Apart from the “automatic” mode, the step controller has three modes in which the actuating signal can be influenced manually:

- manual mode using the MAN signal,
- manual mode with the up/down switch (MAN_GEN),
- manual mode by direct switching of the binary outputs.

The way in which manual values can be generated and connected is illustrated in Figure 6-4. Using the MAN parameter (–100.0% to 100.0%) the variable can be influenced by connecting an absolute value or from within the user program.

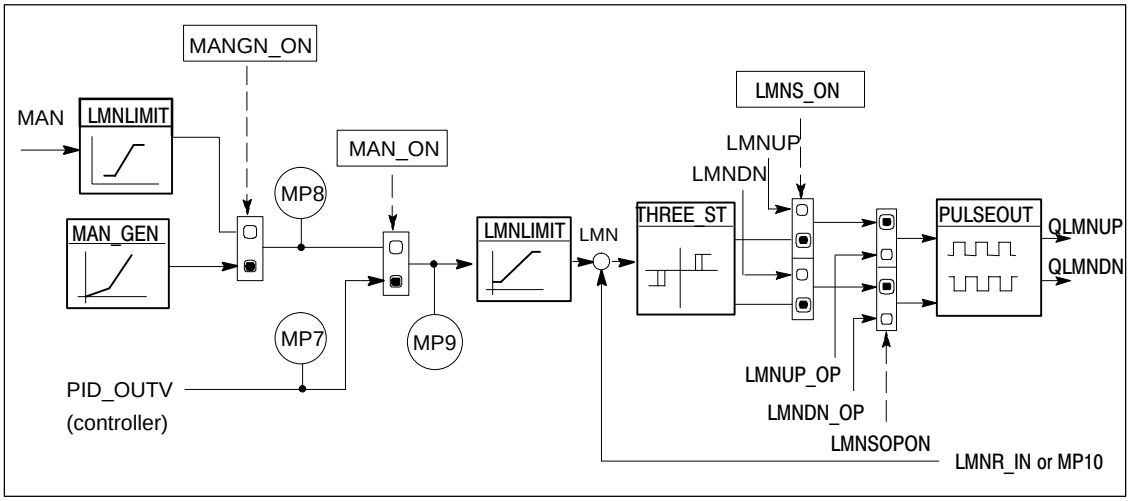


Figure 6-4 Modes and Generating Manual Values on the Step Controller With a Position Feedback Signal

If MAN_GEN is activated from within a different mode the manipulated value at the output (MP9) is adopted. The changeover to the manual value generator is therefore always smooth. The manual manipulated value can be increased or decreased within the limits LMN_HLM and LMN_LLM.

Due to the direct effect on the states of the output signals, manual switching of the actuator using LMNUP or LMNDN always has priority. When the mode is deactivated with LMNS_ON=FALSE, the next mode is always adopted without a step change.

Changing the Modes

The following table shows the possible modes of the step controller with the required values of the structure switches.

Table 6-1 Modes of the Step Controller

Switch / Mode	MANGN_ON	MAN_ON	LMNS_ON
Automatic mode	any	FALSE	FALSE
Manual mode with absolute value	FALSE	TRUE	FALSE
Manual mode with MAN_GEN	TRUE	TRUE	FALSE
Manual mode with pulse switch	any	any	TRUE

6.2.2 Influencing the Manipulated Variable With the Configuration Tool

LMN Display and Setting in the Loop Monitor

The configuration tool has its own interface to the controller FB. It is therefore possible to interrupt the manipulated variable branch using the configuration tool on a PG/PC and to set your own manipulated value (Figure 6-5).

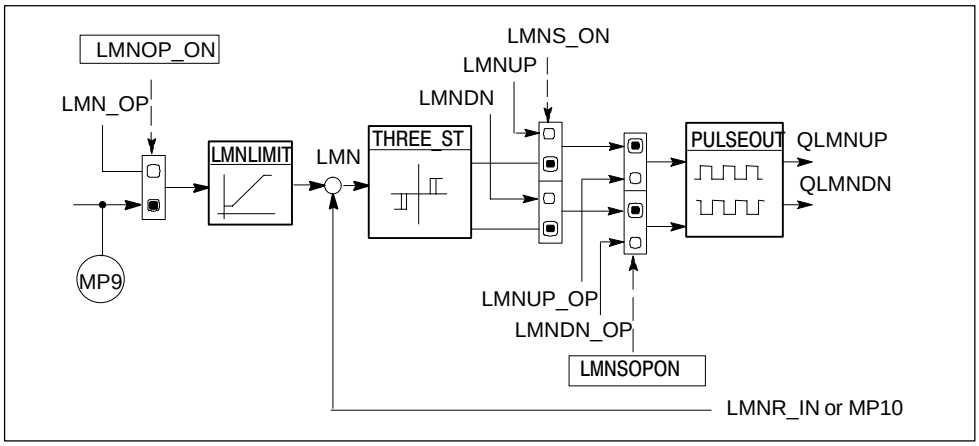


Figure 6-5 Interventions in the Manipulated Variable Branch Using the Configuration Tool

One of the three boxes in the loop monitor window is available for this purpose and is labeled manipulated variable. Here the manipulated value currently applied to measuring point MP9 is displayed in the “Controller:” field. The field below this (’PG: is used to display and change the parameter LMN_OP.

Changeover to the Manipulated Value Specification by 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 and LMN_OP is enabled to the manipulated value in the controller FB.

If the switch ”Controller/PG:” in the actuating signals field is set to “PG:”, the parameter LMNSOPON=TRUE is set and the actuating signal outputs can be operated via the parameters LMNUP_OP (high) or LMNDN_OP (low) in the control loop. This applies to the step controller with and without position feedback.

These manual interventions only affect the process after they have been transferred to the programmable logic controller by clicking the “Send” button in the loop monitor.

6.2.3 Limiting the Absolute Value of the Manipulated Variable (LMNLIMIT_IN or LMNR_PER)

Application

The range of the manipulated variable determines the operating range, in other words the range through which the actuator can move within the permitted range of values. Since the limits for permitted manipulated values do not normally match the 0% or 100% limit of the manipulated value range, it is often necessary to further restrict the range

To avoid illegal statuses occurring in the process, the range for the manipulated variable has an upper and lower limit in the manipulated variable branch.

The LMNLIMIT Function

The 'LMNLIMIT' function limits the LMN(t) to selected upper and lower values LMN_HLM and LMN_LLM. These values can be pre-defined. The input variable inv must, however, lie outside these limits. Since the function cannot be disabled, an upper and lower limit must dalways be assigned during the configuration.

The numerical values of the limits (as percentages) are set at the input parameters for the upper and lower limits. If these limits are exceeded by the input variable inv(t), this is indicated at the signaling outputs (Figure 6-7).

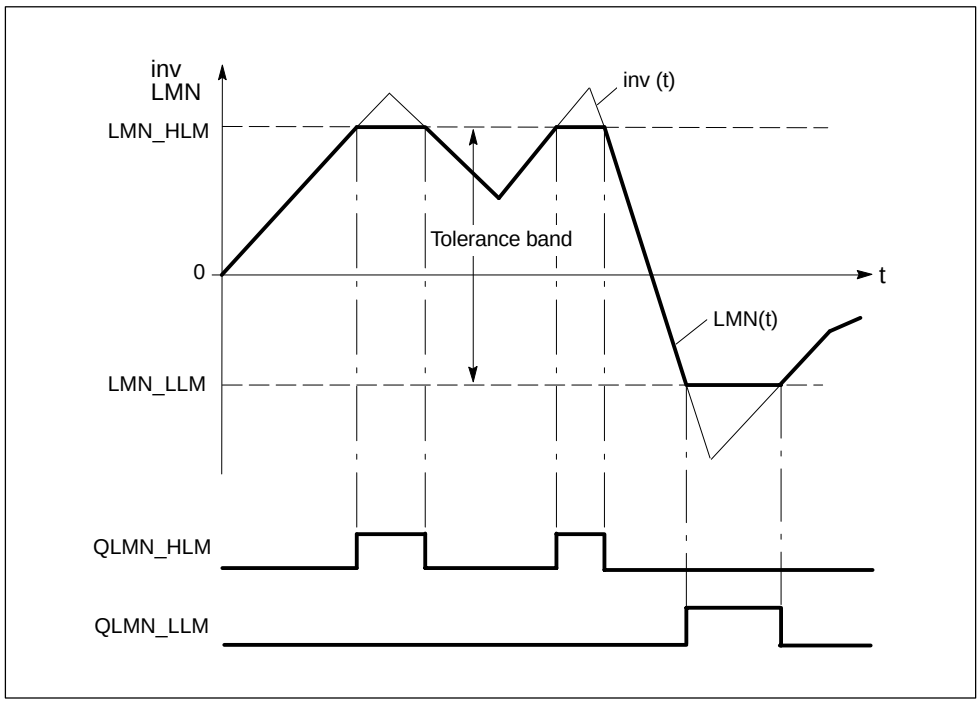


Figure 6-6 Absolute Value Limits of the Manipulated Variable LMN(t)

Start Up and Mode of Operation

- In case of a complete restart all the signal outputs are set to zero.
- The limitation operates as shown in the following table:

LMN =	QLMN_HLM	QLMN_LLM	when:
LMN_HLM	TRUE	FALSE	INV ≥ LMN_HLM
LMN_LLM	FALSE	TRUE	INV ≤ LMN_LLM
INV	FALSE	FALSE	LMN_HLM ≤ INV ≤ LMN_LLM

The effective manipulated value of the controller is indicated at the output (parameter LMN).

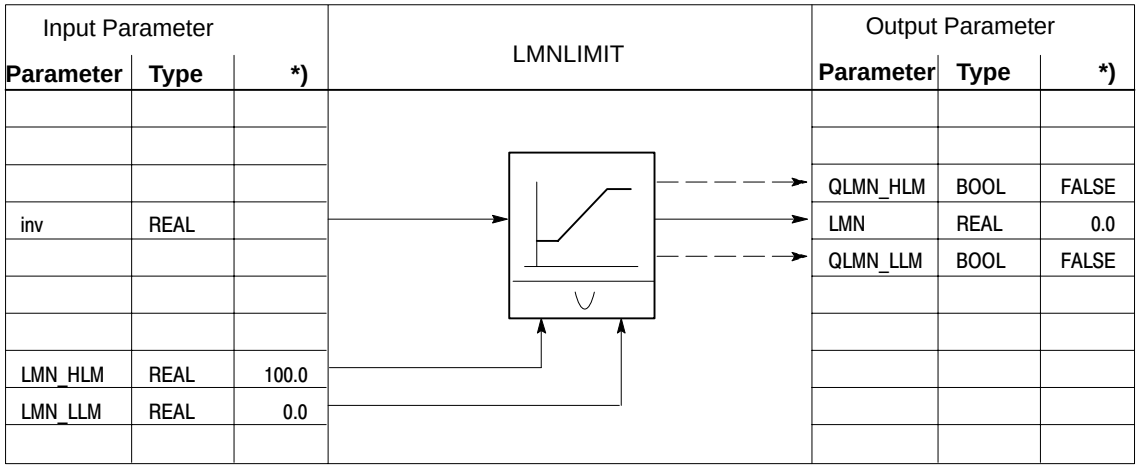
Parameters of the function LMNLIMIT

The input value INV is an implicit parameter. It is only accessible at measuring point **MP9** using the configuration tool.

For the limitation function to operate properly, the following must apply:

LMN_HLM > LMN_LLM

Parameter	Meaning	Permitted Values
LMN_HLM	Upper limit of the man. variable	LMN_LLL ... 100.0 [%]
LMN_LLM	Lower limit of the man. variable	–100.0 ... LMN_HLM [%]



*) Default when the instance DB is created

Figure 6-7 Functions and Parameters of the Absolute Value Limitation of the Manipulated Value

6.2.4 Processing the Position Feedback Signal (LMNR_IN or LMNR_PER)

Signal Adaptation

Inputs with suitable signal processing functions are available for interconnecting the position feedback signal to the comparator in the manipulated variable branch of the step controller. (Figure 6-8).

Input LMNR_PER is used to connect signals in the format of SIMATIC I/Os (peripheral format) and LMNR_IN to connect signals in floating point format.

The corresponding internal value is accessible at measuring point MP10 as a percentage.

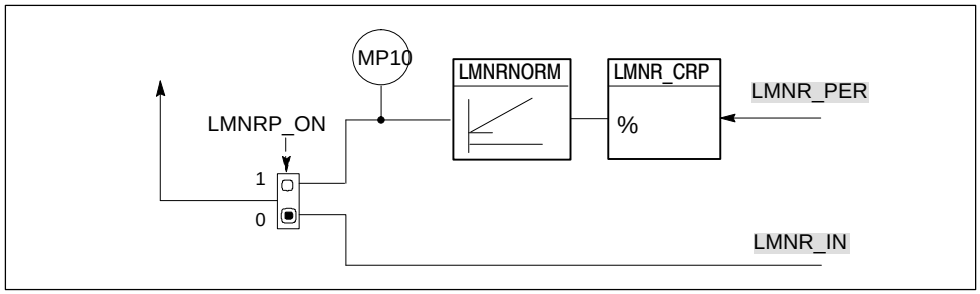


Figure 6-8 Processing the Position Feedback Signal With the Step Controller

The LMNR_CRP Function

If the value of the position feedback signal is provided by an analog input module, the numerical value of the I/O data word must be converted to a numerical value in the floating point format (as a percentage).

The LMNR_CRP function converts the numerical value of the position feedback signal at input LMNR_PER to a floating point value normalized to a percentage. There is no check for positive/negative overflow, over/underdrive or wire break.

The following table provides an overview of the ranges and numerical values before and after processing with the conversion and normalization algorithm of the LMNR_CRP function.

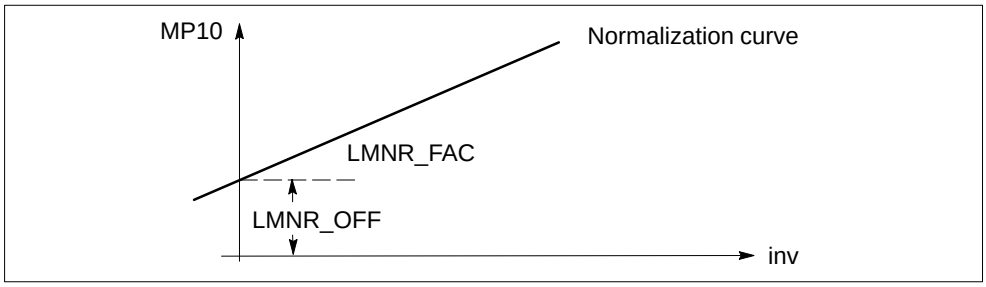
LMNR_PER Peripheral (I/O) Value	Output Value in %
32767	118,515
27648	100,000
1	0,003617
0	0,000
– 1	– 0,003617
– 27648	– 100,000
– 32768	– 118,519

The Function LMNRNORM

If the position feedback is a physical value (for example 240 ... 800 mm or 0 ... 60 ∡°), then the feedback input that has already been converted to a floating point value (as a percentage) must also be normalized to the required internal floating point value in the range between 0 and 100%.

To specify the straight line normalization curve, the following parameters must be defined:

- the factor (for the slope): **LMNR_FAC**
- the offset of the normalization curve from zero: **LMNR_OFF**



The normalization value MP10 (Figure 6-8) is calculated from the input value inv (LMNR_PER) as follows:

$$MP10 = inv * LMNR\_FAC + LMNR\_OFF$$

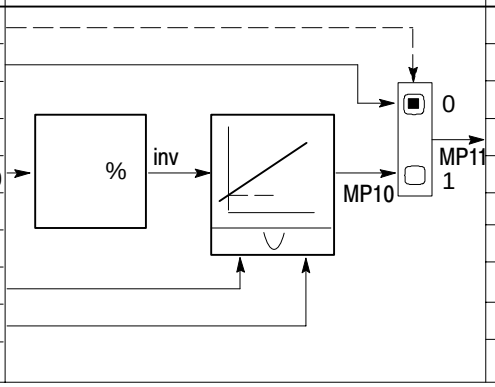
Restart

The function is effective when the control input LMNRP_ON = TRUE is set. Internally, the values are not limited. The parameters are not checked.

Parameters of the LMNR_CRP and LMNRNORM Functions

The LMNR_PER peripheral input is switched to the feedback branch when LMNRP_ON = TRUE is set. The value of LMNR_PER (in the internal format) is accessible at measuring point MP10.

Parameter	Meaning	Permitted Values
LMNR_PER	Feedback value in peripheral format	
LMNR_FAC	Slope of the curve at the input of the position feedback signal LMNR_PER	Technical range of values (no dimension)
LMNR_OFF	Zero point of the LMNR normalization curve	–100.0 ... + 100.0 [%]

Input Parameter			LMNR_CRP + LMNRNORM	Output Parameter		
Parameter	**Type**	***)**		**Signal**	**Type**	***)**
LMNRP_ON	BOOL	FALSE				
LMNR_IN	REAL	0.0				
				LMNR	REAL	
LMNR_PER	INT	0				
LMNR_FAC	REAL	1.0				
LMNR_OFF	REAL	0.0				

*) Default when the instance DB is created

Figure 6-9 Functions and Parameters of the Peripheral Value Conversion for the Position Feedback Signal

6.2.5 Generating the Actuating Signals (QLMNUP/QLMNDN)

Signal Processing

The difference between the manipulated value LMN and the position feedback signal LMNR is switched to the three-step element with hysteresis THREE_ST. The PULSEOUT pulse generator that follows this element ensures that a minimum pulse time and minimum break time are maintained to reduce wear and tear on the actuators (Figure 6-10). If the limit position switches of the actuator (LMNR_HS/LMNR_LS) are triggered, the corresponding output is disabled.

The minimum pulse time PULSE_TM and minimum break time BREAK_TM are also taken into account if the binary output signals are activated manually (LMNS_ON=TRUE or LMNSOPON=TRUE). If a limit position switch is activated, the output is also disabled in manual operation.

If both signal switches are set for actuating signal operation (LMNUP = LMNDN = TRUE or LMNUP_OP = LMNDN_OP = TRUE), the outputs PLMNUP and QLMNDN always output as FALSE.

The direct change from “Actuating signal up” (QLMNUP = TRUE, QLMNDN = FALSE) to “Actuating signal down” (QLMNUP = FALSE, QLMNDN = TRUE) is not possible. The pulse generator inserts a cycle with QLMNUP = QLMNDN = FALSE.

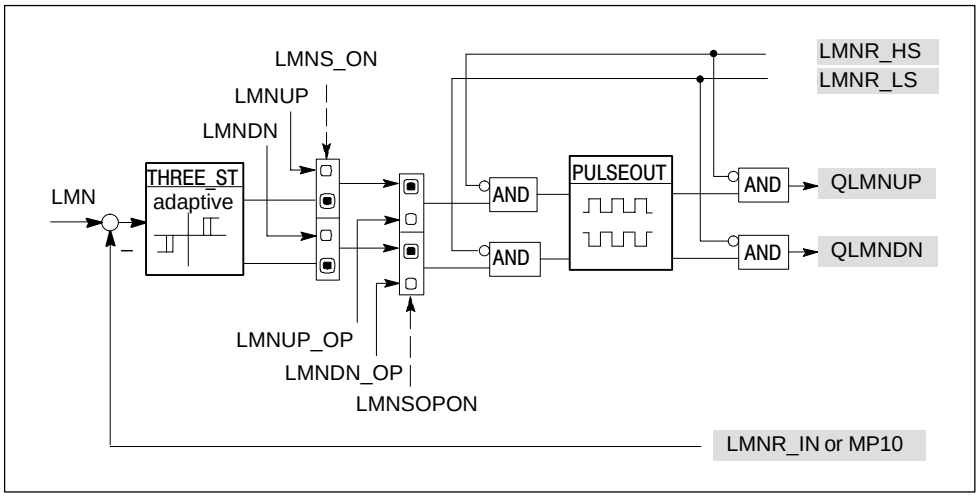


Figure 6-10 Generating the Binary Actuating Signal With Position Feedback Signal

The Three-step Element With Hysteresis THREE_ST

The deviation between the values of the actuating signal of the controller and the actual position reached by the actuator forms the input variable of the three-step element. Two binary signals are generated at its output and are either set or reset depending on the value and sign of the difference at the input.

The three-step switch THREE_ST reacts to the input signal INV as shown in the table below (ThrOn=on threshold, ThrOff=off threshold) and then adopts one of the three possible combinations of output signals UP/DOWN (Figure 6-11):

UP	DOWN	Input Combination
TRUE	FALSE	INV ≥ ThrOn or INV > (ThrOff and UP_{old} = TRUE)
FALSE	TRUE	INV ≤ –ThrOn or INV < (–ThrOff and $DOWN_{old}$ = TRUE)
FALSE	FALSE	|INV| ≤ ThrOff

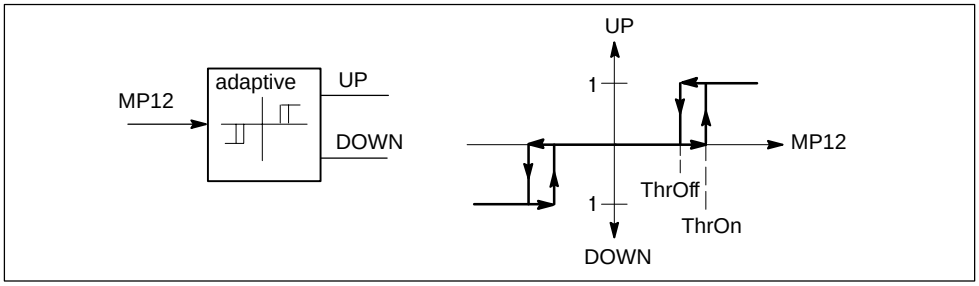


Figure 6-11 Functions of the Three-Step Element THREE_ST

The off threshold ThrOff must be higher than the change in the position feedback signal after the duration of one pulse. This value depends on the actuating time of the motor MTR_TM and is calculated as follows:

$$\text{ThrOff} = 0.5 * \frac{110}{\text{MTR\_TM}} * \text{CYCLE}$$

PULSE_TM must be a whole multiple of CYCLE.

Note

If the motor actuating time is set too high (10% above the real actuating time) the actuating signals QLMNUP and QLMNDN are switched on and off constantly.

Adapting the Response Threshold ThrOn

To reduce the switching frequency when correcting larger error signals, the response threshold ±ThrOn is adapted automatically during operation while ThrOff remains constant. ThrOn ist limited to:

$$\text{Min ThrOn} = \frac{100}{\text{MTR\_TM}} * \text{MAX (PULSE\_TM, CYCLE)}$$

$$\text{Max ThrOn} = 10$$

The adaptation of the response threshold is deactivated for pure P, D or PD controllers. Thus:

ThrOn = Min ThrOn.

The PULSEOUT Pulse Generator

The pulse generator makes sure that when the output pulses are set and reset, a minimum value is maintained for the pulse duration and pulse break.

To protect the actuator, you can therefore select a minimum pulse time PULSE_TM and a minimum break time BREAK_TM. The duration of the output pulses QLMNUP or QLMNDN is always greater than PULSE_TM and the break between two pulses is always larger than BREAK_TM. Figure 6-12 illustrates the functions of PULSEOUT based on the example of the UP signal.

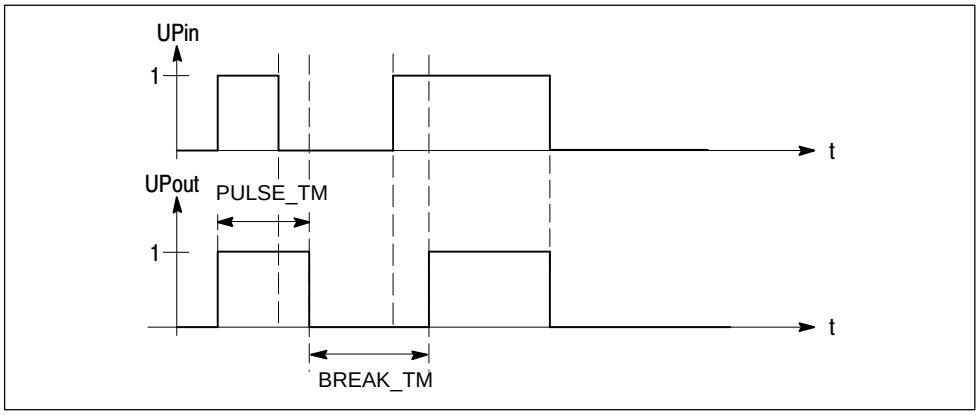
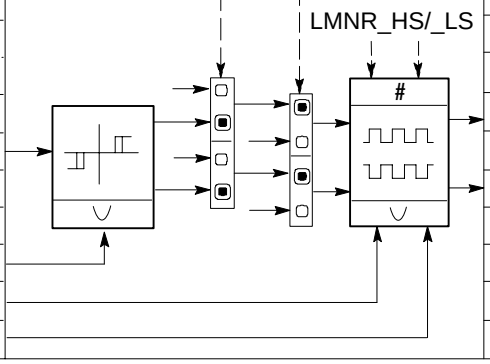


Figure 6-12 Functions of the Pulse Generator PULSEOUT

Parameters of THREE_ST and PULSEOUT

The values set for the parameters PULSE_TM and BREAK_TM must be a whole multiple of the cycle time CYCLE. If the values set are smaller than CYCLE, then the cycle time CYCLE is used for the minimum pulse and minimum break times.

Input Parameter			THREE_ST PULSEOUT	Output Parameter		
Parameter	**Type**	***)**		**Signal**	**Type**	***)**
LMNSOPON	BOOL	FALSE				
LMNUP_OP	BOOL	FALSE				
LMNDN_OP	BOOL	FALSE				
LMNS_ON	BOOL	TRUE				
LMNUP	BOOL	FALSE				
LMNDN	BOOL	FALSE				
LMNR_HS	BOOL	FALSE				
LMNR_LS	BOOL	FALSE				
				QLMNUP	BOOL	FALSE
MP12	REAL	0.0				
				QLMNDN	BOOL	FALSE
MTR_TM	TIME	T#30s				
PULSE_TM	TIME	T#3s				
BREAK_TM	TIME	T#3s				



*) Default when the instance DB is created

Figure 6-13 Functions and Parameters for Generating Actuating Signals on the Step Controller

6.3 Manipulated Variable Processing on the Step Controller Without Position Feedback Signal

Structure and Function of the Step Controller

The step controller (PID_ES) without a position feedback signal consists of two parts:the PD section that operates with continuous signals and a second section in which the binary actuating signals are generated from the difference between the PD action and feedback (Figure 6-14).

The integrator in the feedback path of this step controller totals the error signal from ± 100/MTR_TM and $ER_{Normalized}$*GAIN/TI. The difference between the assumed motor position and the I action is applied to the output of the integrator. In the settled state, the output of the integrator and the PD action become zero. Since the input of the three-step element also becomes zero, the binary actuating signals QLMNUP and QLMNDN are set to FALSE. The I action of the PID algorithm is disabled. Functions for assigning defaults to the I action or holding the I action are not implemented on the step controller without position feedback signal. A manual mode using the MAN parameter is also omitted because there is no information about the position of the actuator.

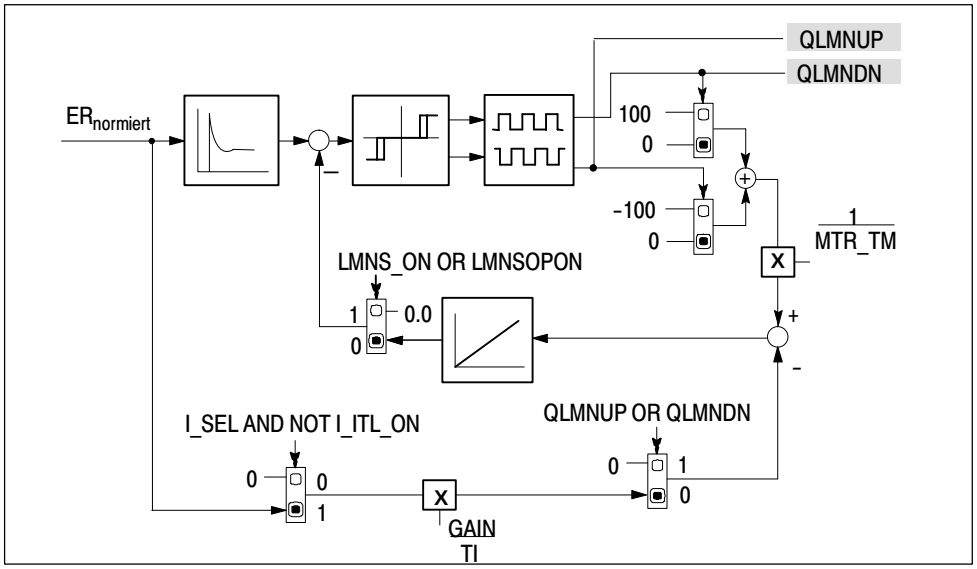


Figure 6-14 Step Controller Without Position Feedback Signal

To avoid overloading the drive, its limit stop signals (LMNR_HS/LMNR_LS) can be used to interlock the controller outputs (Figure 2-17). If the actuator drive does not provide limit stop signals, the input parameters LMNR_HS and LMNR_LS = FALSE must be set.

Note

If no limit position signals exist, the controller cannot recognize when a mechanical limit is reached. It is possible that the controller then outputs signals, for example, to open the valve although it is already fully open.

Modes of the Step Controller

- **Selection: Step controller without position feedback signal**

  If there is no position feedback signal available to indicate the position of the actuator, the control structure illustrated in Figure 6-14 is activated by setting LMNR_ON = FALSE.

- **Operating modes**

  Due to the absence of information about the position of the actuator, there is no manual mode with the MAN parameter or with the manual value generator MAN_GEN on the step controller without position feedback signal. Apart from the “automatic” mode, in other words closed loop control, the “manual” mode with direct keying of the output pulses can also be set in the open loop with LMNS_ON = TRUE.

Manual Mode

When the manual mode is active (LMNS_ON = TRUE), the binary output signal QLMNUP and QLMNDN can be set using the switches LMNUP and LMNDN (Figure 6-15). The minimum pulse time PULSE_TM and minimum pulse break are maintained.

If one of the limit position switches LMNR_HS or LMNR_LS is set, the corresponding output signal is also disabled in the manual mode.

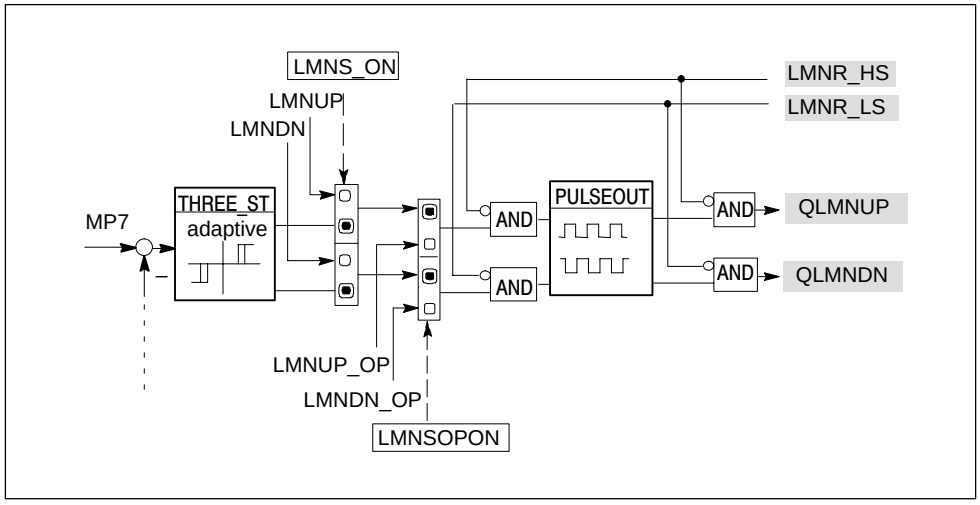


Figure 6-15 Manual Mode With the Step Controller Without Position Feedback Signal

The direct manual mode using LMNUP or LMNDN directly affects the output signals and therefore always has priority. When the controller switches back to the automatic mode with LMNS_ON = FALSE, the change is always smooth.

The following table shows the possible modes of the step controller without position feedback signal:

Table 6-2 Modes of the Step Controller Without Position Feedback Signal

Mode \ Switch	LMNS_ON
Automatic mode	FALSE
Manual mode setting binary output signals	TRUE

Generating the Actuating Signals QLMNUP/QLMNDN

The difference between the PD component of the controller and the feedback value (MP 11) is switched to the three-step element with hysteresis THREE_ST. The PULSEOUT pulse generator that follows this element ensures that a minimum pulse time and minimum break time are maintained to reduce wear and tear on the actuators (Figure 6-16). If the limit position switches of the actuator (LMNR_HS/LMNR_LS) are triggered, the corresponding output is disabled.

The minimum pulse time PULSE_TM and minimum break time BREAK_TM are also taken into account if the binary output signals are activated manually (LMNS_ON=TRUE or LMNSOPON=TRUE (Figure 6-15). If a limit position switch is activated, the output is also disabled in manual operation.

If both signal switches are set for actuating signal operation (LMNUP = LMNDN = TRUE or LMNUP_OP = LMNDN_OP = TRUE), the outputs PLMNUP and QLMNDN always output as FALSE.

The direct change from “Actuating signal up” (QLMNUP = TRUE, QLMNDN = FALSE) to “Actuating signal down” (QLMNUP = FALSE, QLMNDN = TRUE) is not possible. The pulse generator inserts a cycle with QLMNUP = QLMNDN = FALSE.

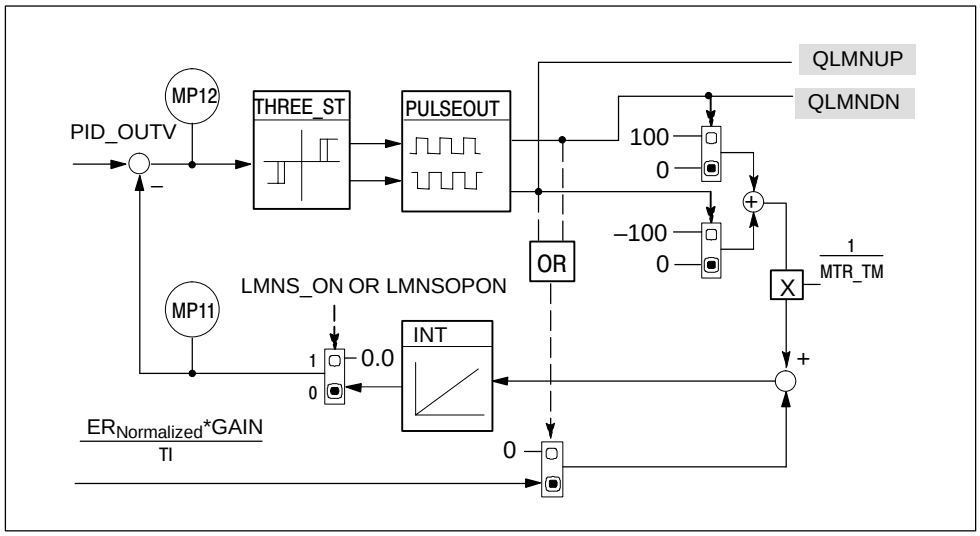


Figure 6-16 Generating the Binary Actuating Signal on the Step Controller Without Position Feedback Signal

The Three-step Element With Hysteresis THREE_ST

The difference between the values of the PD component of the controller and the feedback value forms the input variable of the three-step element. Two binary signals are generated at its output and are either set or reset depending on the value and sign of the difference at the input.

The three-step element THREE_ST reacts to the input signal MP12 (PD component feedback) in accordance with the following relationships (ThrOn = On threshold, ThrOff = Off threshold) and then adopts one of the three possible combinations of output signals UP/DOWN (Figure 6-17):

UP	DOWN	Input Combination		
TRUE	FALSE	INV ≥ ThrOn	or	INV > (ThrOff and UP_{old} = TRUE)
FALSE	TRUE	INV ≤ ThrOn	or	INV < (–ThrOff and $DOWN_{old}$ = TRUE)
FALSE	FALSE	INV ≤ ThrOff		

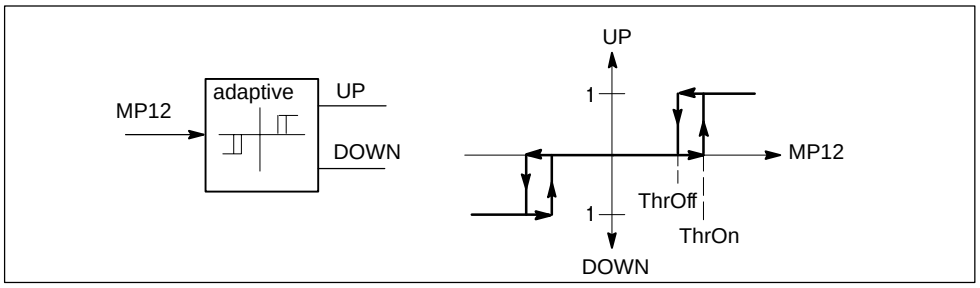


Figure 6-17 Functions of the Three-Step Element THREE_ST

The off threshold ThrOff must be higher than the change in the position feedback signal after the duration of one pulse. This value depends on the actuating time of the motor MTR_TM and is calculated as follows:

$$\text{ThrOff} = 0.5 * \frac{110}{\text{MTR\_TM}} * \text{CYCLE}$$

Adapting the Response Threshold ThrOn

To reduce the switching frequency when correcting larger error signals, the response threshold ±ThrOn is adapted automatically during operation while ThrOff remains constant. ThrOn ist limited to:

$$\text{Min ThrOn} = \frac{100}{\text{MTR\_TM}} * \text{MAX (PULSE\_TM, CYCLE)}$$

$$\text{Max ThrOn} = 10$$

The adaptation of the response threshold is deactivated for pure P, D or PD controllers. Thus:

ThrOn = Min ThrOn.

The PULSEOUT Pulse Generator

The pulse generator has the same functions as step controllers with position feedback signals (see Section 6.2.5).

Simulating the Position Feedback Signal

If no position feedback signal is available as a measurable value, this can also be simulated (LMNRS_ON = TRUE). When optimizing the PID controller parameters using the configuration tool, the position feedback signal is always required as an input variable.

The position feedback signal is simulated by an integrator using the motor actuating time MTR_TM as the reset time (Figure 6-18). In the status LMNRS_ON = FALSE, the start value of the parameter LMNRSVAL is output at the integrator output LMNR_SIM. After switching to TRUE, the simulation starts using this value.

If LMNR_HS = TRUE is set, the integration is limited upwards, is LMNR_LS = TRUE is set, it is limited downwards. There is no matching of the simulated position feedback signal to the limit positions.

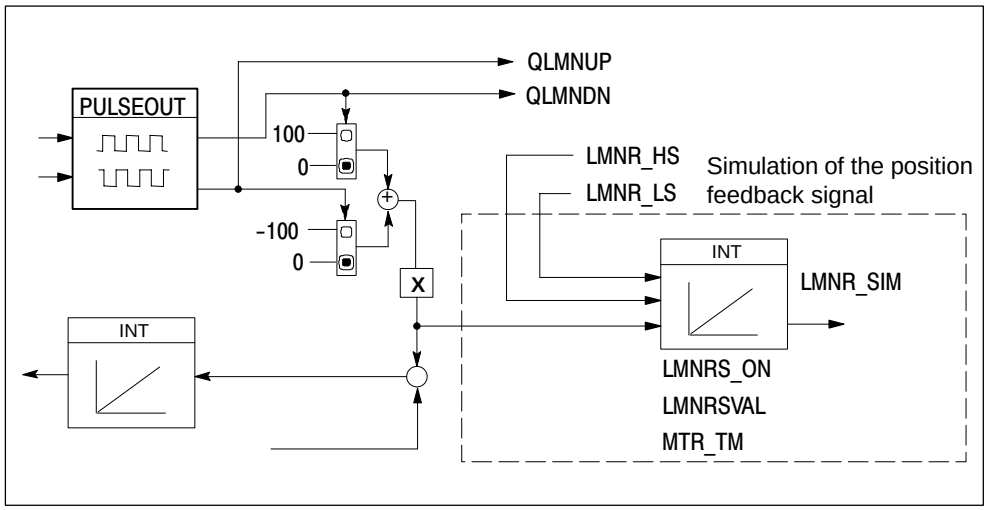


Figure 6-18 Simulation of the position feedback signal

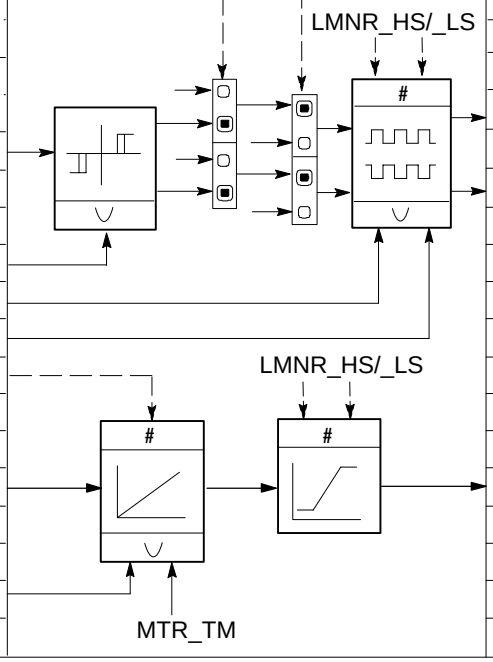
Note

The position feedback signal is only simulated. It does not necessarily match the actual position of the actuator. If a real position feedback exists, this should always be used.

Parameters for Manipulated Variable Processing

The values set for the parameters PULSE_TM and BREAK_TM must be a whole multiple of the cycle time CYCLE. If the values set are smaller than CYCLE, then the cycle time CYCLE is used for the minimum pulse and minimum break times.

Input Parameter			THREE_ST, PULSEOUT, Simulation of the position feedback signal	Output Parameter		
Parameter	**Type**	***)**		**Signal**	**Type**	***)**
LMNSOPON	BOOL	FALSE				
LMNUP_OP	BOOL	FALSE				
LMNDN_OP	BOOL	FALSE				
LMNS_ON	BOOL	TRUE				
LMNUP	BOOL	FALSE				
LMNDN	BOOL	FALSE				
LMNR_HS	BOOL	FALSE				
LMNR_LS	BOOL	FALSE		QLMNUP	BOOL	FALSE
MP12	REAL	0.0				
				QLMNDN	BOOL	FALSE
MTR_TM	TIME	T#30s				
PULSE_TM	TIME	T#3s				
BREAK_TM	TIME	T#3s				
LMNRS_ON	BOOL	FALSE				
inv	REAL			LMNR_SIM	REAL	
LMNRSVAL	REAL	0.0				



*) Default when the instance DB is created

Figure 6-19 Functions and Parameters for Generating Actuating Signals on the Step Controller Without Position Feedback Signal

6.4 Step Controllers in Cascade Controls

Interrupting the Controller Cascade

In a cascade, several controllers are directly dependent on each other. You must therefore make sure that if the cascade structure is interrupted at any point, the cascade operation can be resumed without causing any problems.

In the secondary or slave controllers of a cascade control system, a QCAS signal is formed by ORing the status signals of the switches in the setpoint and manipulated variable branches. This signal operates a switch in the secondary controllers that changes the controller to the correction mode. The correction variable is always the process variable PV of the secondary loop (Figure 6-20).

The switch from correction mode to automatic mode smoothy just as it is done when switching from manual mode to automatic mode.

Note

Step controllers (PID_ES) can only be used in cascade controls as slave controllers in secondary control loops.

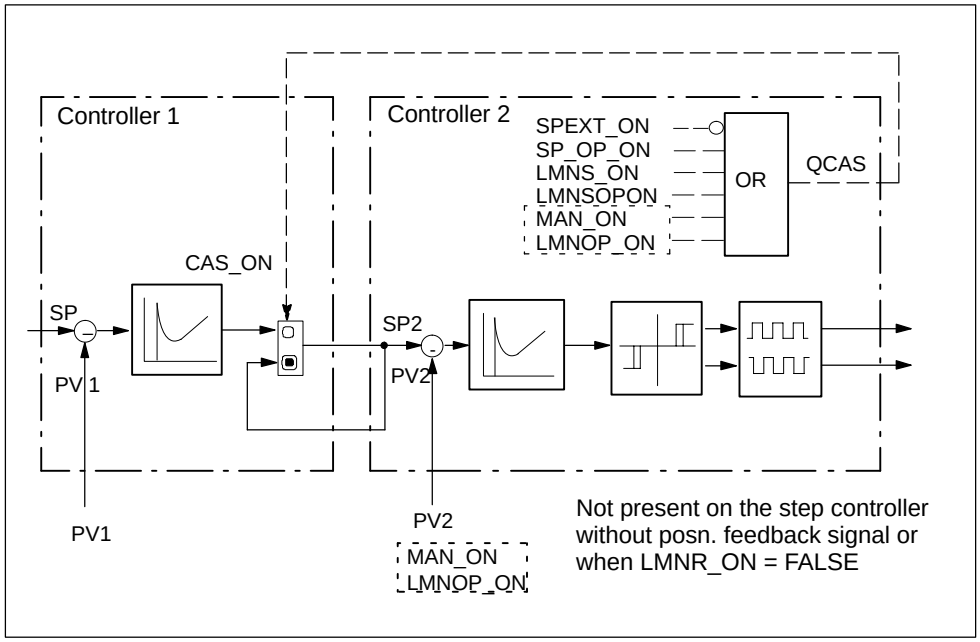


Figure 6-20 Two-Loop Cascade Control With a Step Controller

Block Connections

The following diagram illustrates the principle of the controller or block connections in multi-loop cascades.

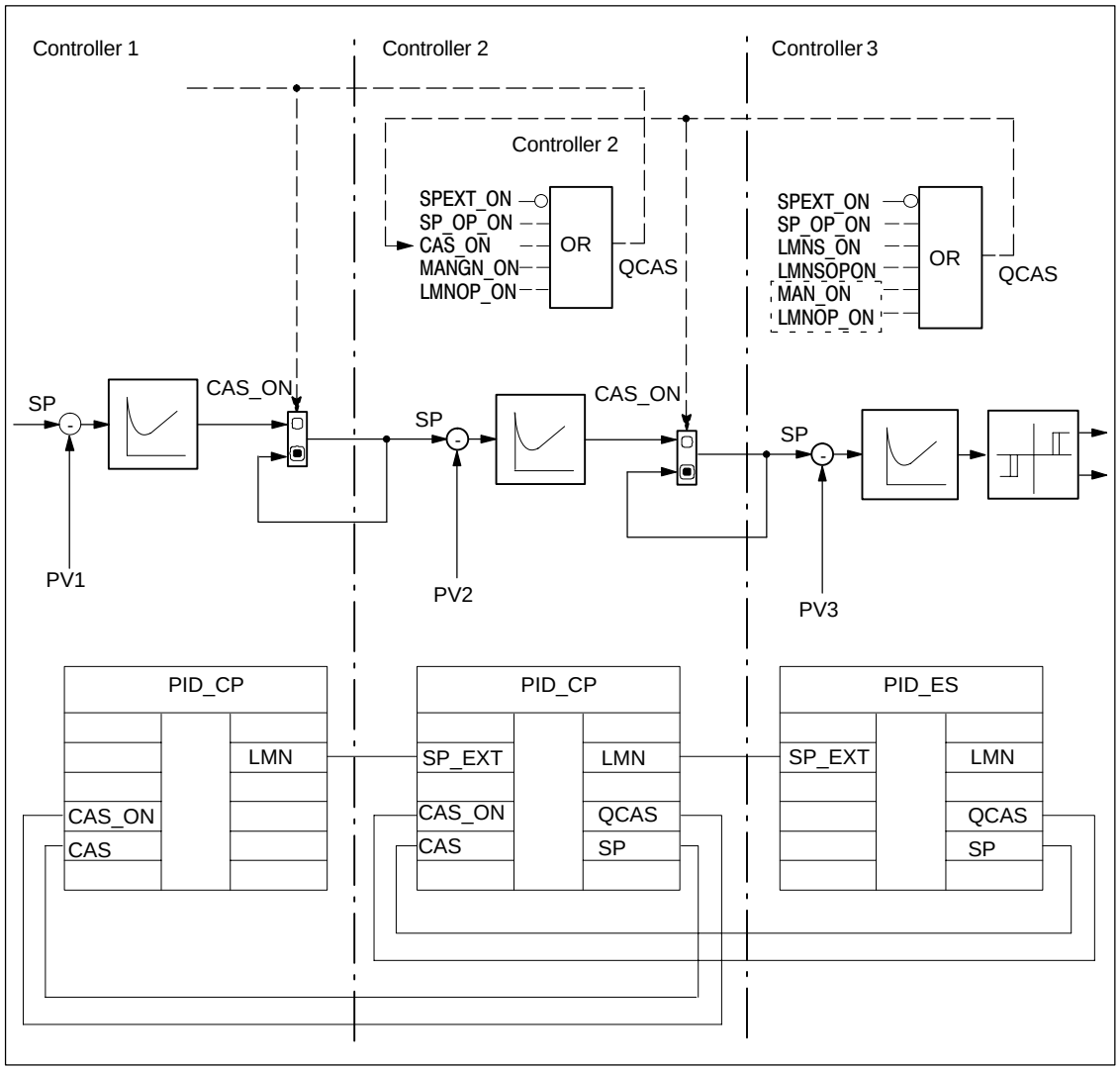


Figure 6-21 Connecting a Cascade With Two Secondary Control Loops and a Step Controller

The Loop Scheduler and Examples of Controller Configurations

7

7.1 The Loop Scheduler (LP_SCHED)

Application

The loop scheduler LP_SCHED is used when the number of watchdog-interrupts of a CPU is not enough to realize the desired (various) sampling times. It allows up to 256 control loops to be called with sampling times which amount to an integer multiple of the watchdog-interrupt cycles.

Overview

The ”LP_SCHED” function reads the parameters specified by you from the ”DB_LOOP” shared data block calculates the variables required to schedule the loops and saves these again into the ”DB_LOOP” data block.

You must call the ”LP_SCHED” FC in a watchdog-interrupt OB. Afterwards you must program a conditional call for all the corresponding control loops in the same watchdog-interrupt OB.The condition for calling the individual control loops is determined by the ”LP_SCHED” FC and is placed in the ”DB_LOOP” DB. The control-loop FBs ”PID_CP” and ”PID_ES” cannot be called through the ”LP_SCHED” FC since the input and output parameters of the FBs must have been assigned.

During operation you can disable the call of individual control loops manually and furthermore reset individual control loops.

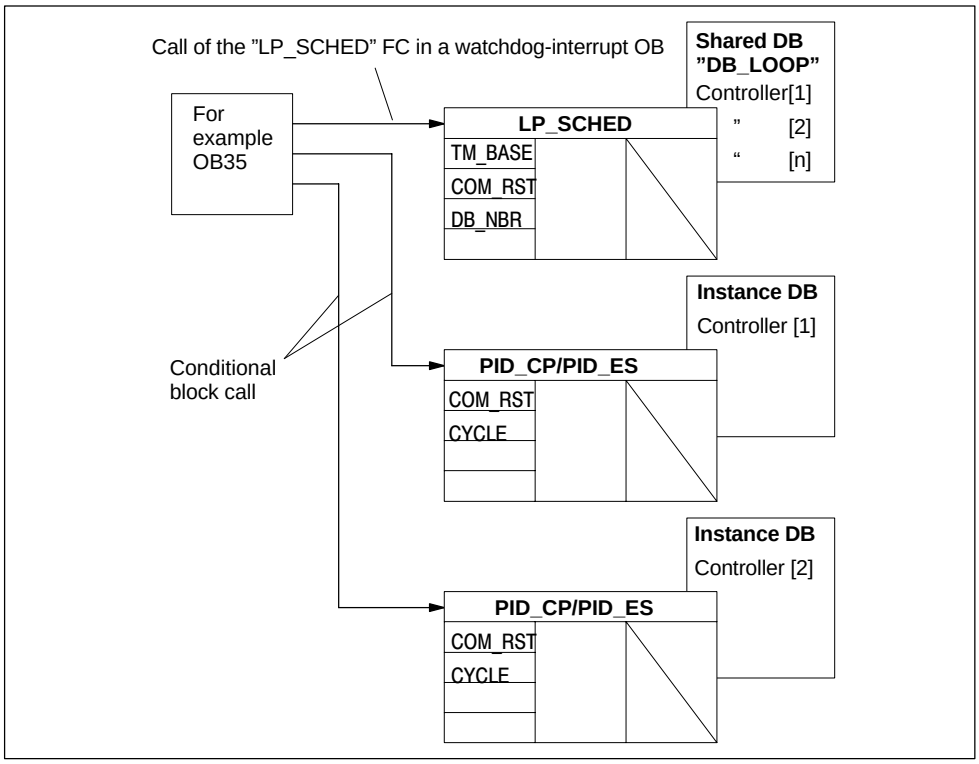


Figure 7-1 Principle of the Controller Call Based of two Control Loops

Structure of the ”DB_LOOP” DB

Parameter	Type	range of values	Description
GLP_NBR	INT	1... 256	Highest control loop number
ALP_NBR	INT	1... 256	Current control loop number
	Control loop No. 1		
MAN_CYC [1]	TIME	≥ 20ms	Sampling time specified by you
MAN_DIS [1]	BOOL		Disable controller call manually
MAN_CRST [1]	BOOL		Set the complete restart manually
ENABLE [1]	BOOL		Enable
COM_RST [1]	BOOL		Complete restart
ILP_COU [1]	INT		Internal loop counter
CYCLE [1]	TIME	≥ 20ms	Sampling time calculated by the ”LP_SCHED” FC
	Control loop No. 2		
MAN_CYC [2]	TIME	≥ 20ms	Sampling time specified by you
MAN_DIS [2]	BOOL		Disable controller call manually
...	..		. . .

Brief overview:

- You have to configure the variables GLP_NBR and MAN_CYC[x], x = 1, ... GLP_NBR parametrieren.
- MAN_DIS[x] is used to disable the call of the control loop x during operation.
- MAN_CRST[x] is used to start an initialization run for the control loop x during operation.
- The ”LP_SCHED” FC enters the call condition for the control loop x in the variable ENABLE[x].
- The variables COM_RST[x] and CYCLE[x] are written by the ”LP_SCHED” FC. They are used to interconnect the inputs COM_RST and CYCLE of the control loop FBs.
- The variables ALP_NBR and ILP_COU[x] are internal variables of the ”LP_SCHED” FC. They can be of use in monitoring the function ”LP_SCHED”.

Configuration of the loop schedules in the ”DB_LOOP” DB

You have to carry out the configuration of the controller loop scheduler without the support of the configuration tool, but you do not have to create the ”DB_LOOP” DB completely new. It is available for copying in the ”Standard PID Control” library.

You have to configure the following variables in the ”DB_LOOP” DB:

- GLP_NBR: Number of control loops (or control loop FBs) whose calls are managed by the ”LP_SCHED” FC (max. of 256)
- MAN_CYC[x], x = 1, ... GLP_NBR: The sampling time desired by you for the individual control loops.Please observe the condition specified below for MAN_CYC[x] for each control loop. Otherwise the configured sampling time cannot be guaranteed.

  If you want to change the corresponding elements of the MAN_CYC field for one or more control loops during operation, this change becomes effective when the ”LP_SCHED” FC is called the next time.

Adding Further Control Loops

If you want to insert one or more control loops into the ”DB_LOOP” DB, open this DB with the DB Editor. Select the declaration view in the ”View” menu. You can now change the ARRAY range of the variables, for example 1, ... 4 instead of 1, ... 3. (You can also remove control loops by the same method.)

After you have changed back to the ”Data view” in the ”View” menu, you now have to adapt the variable GLP_NBR and check the desired sampling time for every control loop (MAN_CYC[x], x = 1, ... GLP_NBR). The condition specified below for MAN_CYC also has to be observed.

Call of the ”LP_SCHED” FC in your Program

The ”LP_SCHED” FC must be called before all control loop FBs.

Observe the following points when assigning values to the input parameter.

- TM_BASE:At this point enter the cycle of the watchdog-interrupt OB in which the ”LP_SCHED” FC is called.
- COM_RST: When the CPU is started, you must call the ”LP_SCHED” FC once with COM_RST = TRUE. You then carry out an initialization run and carry out the pre-assignments described under ”CPU startup”. In cyclic operation (watchdog interrupt) you must call the ”LP_SCHED” FC with COM_RST = FALSE.
- DB_NBR: At this input enter the number of the ”DB_LOOP” DB which the ”LP_SCHED” FC is to access.

After you have called the ”LP_SCHED” FC, you must call up all the corresponding control loop FBs conditionally.The processing of a control loop FB is to be carried out when the respective ENABLE bit in the ”DB_LOOP” DB have the value TRUE. This bit was written beforehand by the ”LP_SCHED” FC. If the control loop FB has been processed, you must assign the ENABLE bit after the value FALSE has been processed.

When you call the control loop FBs you have to interconnect their input parameters COM_RST and CYCLE with the variables COM_RST[x] and CYCLE[x] of the ”DB_LOOP” DB. CYCLE[x] contains the actual sampling time of the control loop x and is written by the ”LP_SCHED” FC at every run. If you have observed the condition specified below for configuring the variable MAN_CYC[x], CYCLE[x] has the same value as MAN_CYC[x]. Otherwise CYCLE[x] contains the value which results when MAN_CYC[x] is rounded to the next integer multiple of TM_BASE *— GLP_NBR .

The following section gives an example for calling the ”LP_SCHED” FC and for the conditional call of a control-loop FB.

STL	Explanation
CALL ”LP_SCHED”	
TM_BASE:=	Here the cycle of the watchdog interrupt is configured.Example: T#100ms or #CYCLE with CYCLE = Input parameter of the block in which the LP_SCHED is called.
COM_RST:	Here the ”LP_SCHED” FC is told whether an initialization run of the called control loops is to take place.Example: FALSE or #COM_RST with COM_RST = Input parameter of the block in which the LP_SCHED FC is called.
DB_NBR	Here the number of the ”DB_LOOP” DB is configured which is to be processed by the ”LP_SCHED” FC. Example: ”DB_LOOP” with DB_LOOP = Name of the DB assigned in the symbol table.
U ”DB_LOOP”.LOOP_DAT[1].ENABLE	
SPBN M002	Control loop call, if ENABLE = TRUE
CALL FBx,DBy	
COM_RST:= ”DB_LOOP”.LOOP_DAT[1].COM_RST	
:	Formal operand list
:	Formal operand list
CYCLE:= ”DB_LOOP”.LOOP_DAT[1].CYCLE	
:	Formal operand list
:	Formal operand list
CLR	
= ”DB_LOOP”.LOOP_DAT[1].ENABLE	Reset ENABLE bit
M002:	Continue in the program, for example conditional call of the next control loop DB
:	

Pulse Generator in Connection with LP_SCHED

If you have activated the pulse generator at the continuous controller PID_CP, the pulse code CYCLE_P has to be written with the parameter LOOP_DAT[x].CYCLE, instead of the parameter CYCLE.

Condition for Configuring the Sampling Time

The ”LP_SCHED” FC can process a maximum of one control loop per call.The following time therefore passes

TM_BASE * GLP_NBR,

until all the control loops has been processed completely once. When configuring the desired sampling time MAN_CYC[x] you must therefore observe the following condition for each control loop:

The sampling time of control loop x must be an integer multiple of the product of the time base and the number of controllers to be processed.

$$MAN\_CYC[x] \overset{!}{=} GV\ (TM\_BASE * GLP\_NBR),\ x = 1, \ldots GLP\_NBR$$

The real sampling time CYCLE[x] of the control loop x is determined by the ”LP_SCHED” FC from MAN_CYC[x] at every run as follows:

- If you have observed the above rule, the actual sampling time CYCLE[x] is identical with the sampling time MAN_CYC[x] specified with you.
- If you have not observed the condition specified above, CYCLE[x] has the value which results when MAN_CYC[x] is rounded to the next integer multiple of TM_BASE * GLP_NBR .

Example of a Loop Scheduler

The following example shows the call sequence of four control loops in a watchdog interrupt OB.A maximum of one control loop can be processed per unit of the time base TM_BASE.The call sequence results from the sequence of the control loop data in the ”DB_LOOP” DB.

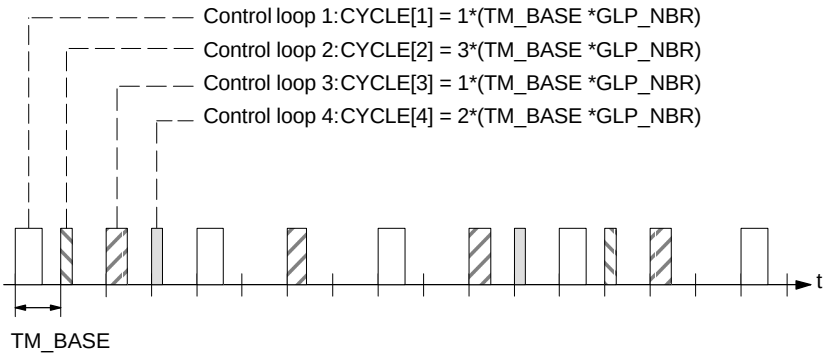


Figure 7-2 Call Sequence of Four Loops Called at Different Intervals

Call of more than one Control Loop FB per Watchdog Interrupt Time Base

If more than one control loop is to be processed in one run of a watchdog interrupt OB, the ”LP_SCHED” FC may also be called several times.All calls of this FC must be carried out before the call of the control loop FBs.You must then enter the time base of the watchdog interrupt OB divided by the number of FC calls at the input parameter TM_BASE of the ”LP_SCHED” FC.

Example: The LP_SCHED FC is called twice in the OB35. The OB35 is processed every 100 ms. The input parameter TM_BASE must therefore be configured with 50 ms.

Run Times

Please note that the sum of all the run times of the ”LP_SCHED” FC and of the control loop FBs which are processed in one run of a watchdog interrupt OB may not exceed the time base of the watchdog interrupt OB.

Note

The block does not check whether or not there is really a shared DB with the number DB_NBR nor whether the parameter GLP_NBR (highest control loop number) matches the length of the data block. If the parameters are incorrectly assigned, the CPU changes to STOP with an internal system error.

Interventions During Operation

The following changes to the ”DB_LOOP” DB are allowed during operation if only the respective parameter is changed and not the complete DB downloaded to the CPU:

- Disabling individual control loops

  If you assign the value TRUE to the variable MAN_DIS[x], processing of the control loop x is disabled during operation. The ”LP_SCHED” FC does not set the ENABLE bit of this control loop to TRUE until you assign FALSE to MAN_DIS[x].

- Initializing a control loop

  You can restart an individual control loop by assigning the value TRUE to the variable MAN_CRST[x]: In this case the ”LP_SCHED” FC assigns TRUE to the variable COM_RST[x] when the control loop x is processed again.At the next processing but one of this control loop the ”LP_SCHED” FC assigns the value FALSE to the variables MAN_CRST[x] and COM_RST[x].

- Changing the sampling time of a control loop

  The parameter MAN_CYC[x] of the ”DB_LOOP” DB may not be changed during operation.

Note

If a control loop is inserted or deleted, that is the entire ”DB_LOOP” DB is downloaded again to the CPU, without the CPU having to carry out a startup, zero must be preassigned to the internal control loop counters ILP_COU[x], x = 1, ... GLP_NBR and the parameter for the current control loop number ALP_NBR.

CPU Startup

During a startup of the CPU you must call the ”LP_SCHED” FC from the corresponding start-up OB and assign the value TRUE to the input COM_RST . You must assign the value FALSE again to this input in the watchdog interrupt OB. The ”LP_SCHED” FC disposes of an initialization routine which is started when TRUE is assigned to the input parameter COM_RST. The following preassignments have to be carried out in the ”DB_LOOP” DB during this initialization run.

- Current control loop number ALP_NBR = 0
- Enable: ENABLE[x] = NOT MAN_DIS[x], x = 1, ... GLP_NBR
- Sampling time: CYCLE[x] has the value assigned to it which results when MAN_CYC[x] is rounded to the next integer multiple of TM_BASE * GLP_NBR, x= 1, ... GLP_NBR.
- Control loop initialization:COM_RST[x] = TRUE, x = 1, ... GLP_NBR
- Internal loop counter: ILP_COU[x] = 0, x = 1, ... GLP_NBR

After the call for the ”LP_SCHED” FC in the start-up OB call the control loop conditionally there, so that it can carry out your initializations.

Monitoring the ”LP_SCHED” FC

The ”LP_SCHED” FC enter the number of the next control loop to be executed in the variable ALP_NBR of the ”DB_LOOP” DB. The number of the respective control loop results from the positioning of its call data in the sequence of entries in the DB (see Table 7-1).

The variable ILP_COU[x] is the internal control loop counter of the ”LP_SCHED” FC. It contains the time duration until the next call of the corresponding control loop.The time unit of ILP_COU is the product of the time base TM_BASE and the number of control loops GLP_NBR. If ILP_COU = 0, the ”LP_SCHED” FC sets the ENABLE bit of the respective control loop.

If Control Loops cannot be Called Unexpectedly

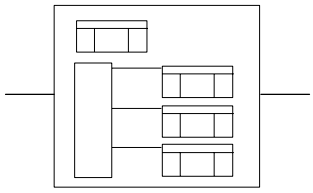
If the function ”LP_SCHED” is called, but individual control loops cannot be processed, this can have the following causes:

- If the value TRUE has been assigned to the variable MAN_DIS[x], processing of the control loop x is disabled during operation.
- The number of FBs or of control loops which are to be processed by the ”LP_SCHED” FC has been specified too low in the GLP_NBR parameter.
- The sampling times MAN_CYC[x] of the individual control loops specified by you may not be smaller than the product of the time base TM_BASE and the number of control loops GLP_NBR. Control loops which do not fulfil this condition are not processed.

Parameters of the ”LP_SCHED” FC

The function ”LP_SCHED” controls the call of individual controllers within a watchdog interrupt OB.

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

Input Parameter			LP_SCHED	Output Parameter		
Parameter	**Type**	***)**		**Parameter**	**Type**	
						
TM_BASE	TIME	100 ms				
COM_RST	BOOL	FALSE				
DB_NBR	BLOCK_DB	DB1				

*) Default when the instance DB is created

Figure 7-3 Block Diagram and Parameters of the LP_SCHED Function

7.2 Example1: Step Controller with Process Simulation

Application

Example1 encompasses a standard step controller (PID_ES) in combination with a simulated process, which consists of an integrating final controlling element and a downstream third order delay element (PT3).

Example1 is a simple example of how to generate a step controller and to configure and test it in all its properties in off-line mode with a typical process setup.

The example will help inexperienced users to understand how controllers with a discontinuous output are used and configured in commonly encountered control systems involving processes with motor-driven actuators. This example can be used as an introduction or for training purposes.

By selecting the parameters, you can change the loop to approximate a real process. Using the configuration tool, you can go through an identification run using the model process to obtain a set of suitable controller characteristic data.

Functions of Example1

Example1 essentially consists of the two combined function blocks PID_ES and PROC_S. PID_ES embodies the standard controller used and PROC_S simulates a process with the function elements ”Valve” and PT3 (Figure 7-4). Apart from the process variable, the controller also receives information about the position of the actuator and limit position signals if limit stops are reached.

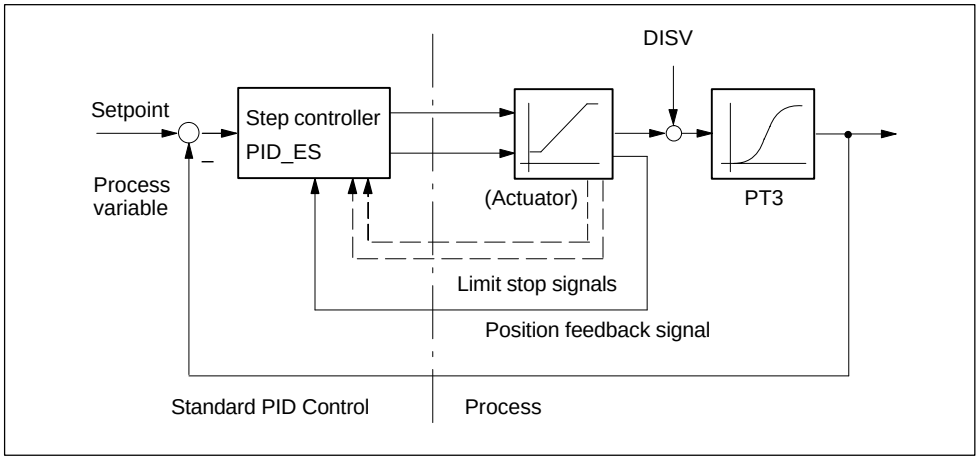


Figure 7-4 Example1, Control Loop

The function block PROC_S emulates a series connection which consists of an integrating final controlling element and three first order delay elements (Figure 7-5). The disturbance variable **DISV** is always added to the output signal of the final controlling element so that process disturbances can be feedforwarded manually here. The factor **GAIN** can be used to determine the static process gain.

The parameter for the motor actuating time **MTR_TM** defines the time which the final controlling element needs for the run from stop to stop.

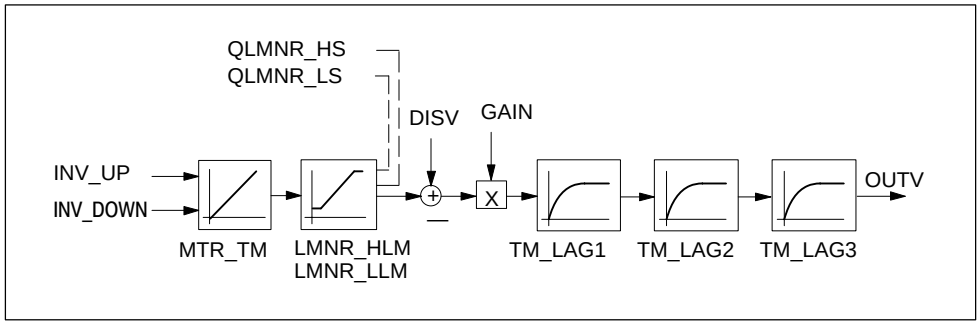


Figure 7-5 Structure and Parameters of the Process Block PROC_S

Block Structure

Example1 is put together from the function APP_1, which encompasses the blocks for the simulated process as well as the call blocks for a complete restart (OB100) and a watchdog interrupt level (OB35 with 100 ms cycle).

Table 7-1 Blocks for Example1

Block	**Name** (in the symbol bar)	**Description**
OB100		Complete restart OB
OB35		Time-driven OB: 100 ms
FC100	APP_1	Example 1
FB2	PID_ES	Step controller
FB100	PROC_S	Process for step controller
DB100	PROCESS	Instance DB for PROC_S
DB101	CONTROL	Instance DB for PID_ES

The two function blocks (Figure 7-6) are assigned the instance data blocks DB100 for the process and DB 101 for the controller.

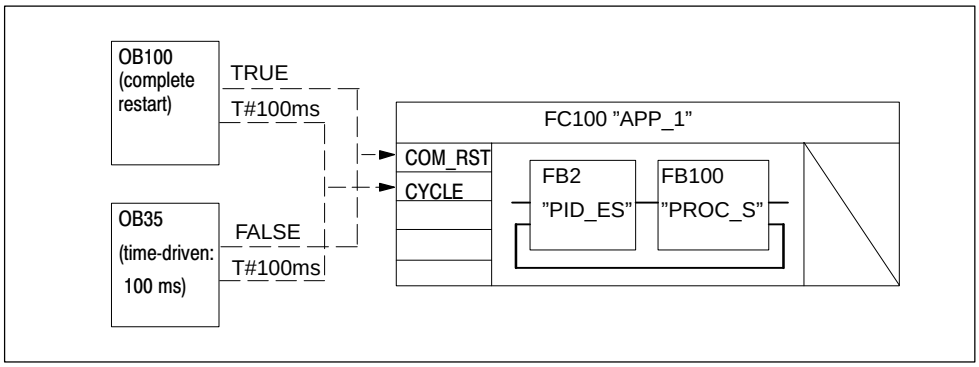


Figure 7-6 Blocks for Example 1: Interconnection and Calling

The Parameters of the Process Model

The parameters of the control block PID_ES and their meaning are described in Chapter 6. The parameters of the process block PROC_S are listed in the following table.

Table 7-2 Parameters of the Process Block "PROC_S" (DB100: FB100)

Parameter	Type	range of values	Description
INV_UP	BOOL		Input signal up (more)
INV_DOWN	BOOL		Input signal down (less)
COM_RST	BOOL		Complete restart
CYCLE	TIME	$\geq$ 1ms	Sampling time
DISV	REAL		Disturbance variable
GAIN	REAL		Loop gain
MTR_TM	TIME		Motor actuating time
LMNR_HLM	REAL	LMNR_LLM ...100.0 [%]	High limit of the position feedback signal
LMNR_LLM	REAL	–100.0...LMNR_HLM [%]	Low limit of the position feedback signal
TM_LAG1	TIME	$\geq$ CYCLE/2	Time lag 1
TM_LAG2	TIME	$\geq$ CYCLE/2	Time lag 2
TM_LAG3	TIME	$\geq$ CYCLE/2	Time lag 3
OUTV	REAL		Output variable
LMNR	REAL		Position feedback signal
QLMNR_HS	BOOL		Actuator at upper limit stop
QLMNR_LS	BOOL		Actuator at lower limit stop

After a **complete restart** the output variable OUTV as well as all the internal memory variables are set to zero.

Interconnection and Calling Example1

Figure 7-7 shows how the step controller is interconnected internally via the function FC100 with the process model to a control loop.

By opening the connection between LMNR and LMNR_IN, it is, of course, possible to implement a step controller without a position feedback signal.

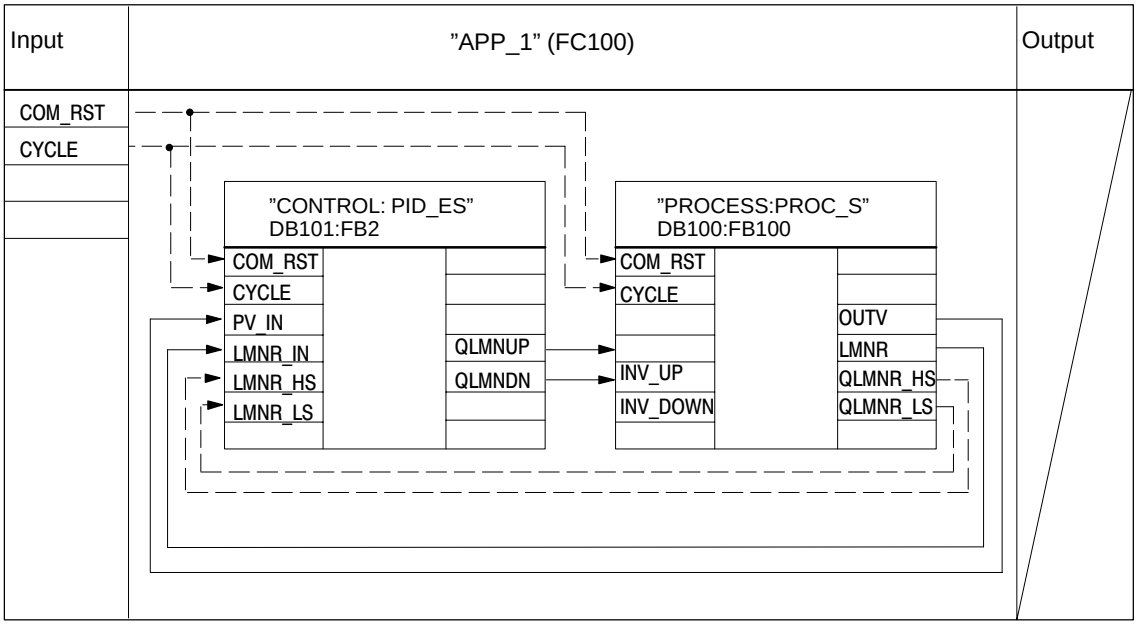


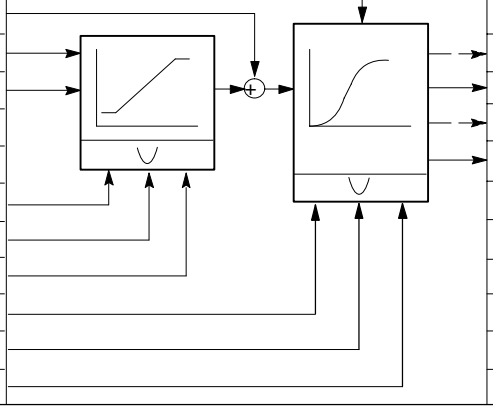
Figure 7-7 FC100 (APP_1), Connections and Call

Parameters of the Model Process for Step Controllers

Figure 7-8 shows the function scheme and the parameters of the process.

At a **complete restart** or a **warm restart** the closed-loop control behaves as described in Section 3.5.

Input Parameter			PROC_S (FB100)	Output Parameter		
Parameter	**Type**	***)**		**Signal**	**Type**	***)**
COM_RST	BOOL	FALSE				
CYCLE	TIME	T#1s				
GAIN	REAL	0.0				
DISV	REAL	0.0				
INV_UP	BOOL	FALSE		QLMNR_HS	BOOL	FALSE
INV_DOWN	BOOL	FALSE		OUTV	REAL	0.0
				QLMNR_LS	BOOL	FALSE
				LMNR	REAL	0.0
LMNR_HLM	REAL	100.0				
LMNR_LLM	REAL	0.0				
MTR_TM	TIME	T#30s				
TM_LAG1	TIME	T#10s				
TM_LAG2	TIME	T#10s				
TM_LAG3	TIME	T#10s				



*) Default when the instance DB is created

Figure 7-8 Functions and Parameters of the PROC_S Process Model

Parameters and Step Response

The reaction of a control loop with a simulated third order PT process is shown on the basis of a concrete configuration of the step controller with PI action and an activated dead band. The selected loop parameters with a 10 sec. time lag approximate the response of a fast temperature process or a level controlling system.

Setting one of the time lags TM_LAGx = 0 sec. reduces the process from third to second order.

The curve (configuration tool) shows the step and settling response of the closed loop after a setpoint change of 60% (Figure 7-9). The table contains the values set for the relevant parameters of the controller and process.

Parameter	Type	Parameter Assignment	Description
Controller:			
CYCLE	TIME	100ms	Sampling time
GAIN	REAL	0.31	Proportional gain
TI	TIME	19.190s	Reset time
MTR_TM	TIME	20s	Motor actuating time
PULSE_TM	TIME	100ms	Minimum pulse time
BREAK_TM	TIME	100ms	Minimum break time
DEADB_ON	BOOL	TRUE	Dead band on
DEADB_W	REAL	0.5	Dead band width
Process:			
GAIN	REAL	1.5	Loop gain
MTR_TM	TIME	20s	Motor actuating time
TM_LAG1	TIME	10s	Time lag 1
TM_LAG2	TIME	10s	Time lag 2
TM_LAG3	TIME	10s	Time lag 3

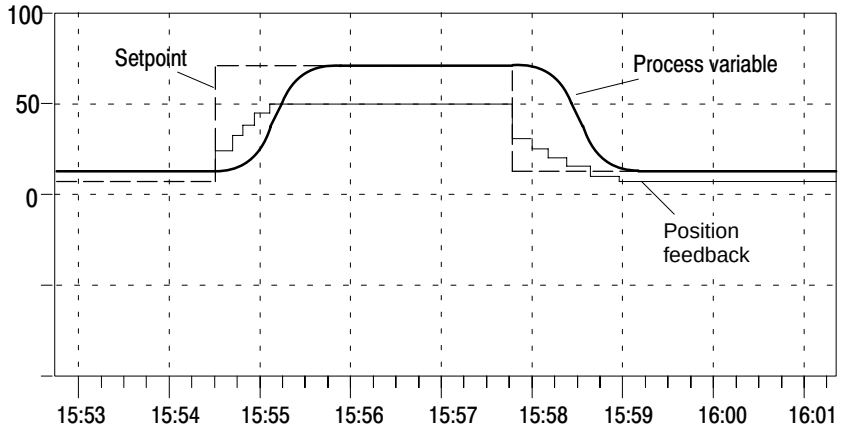


Figure 7-9 Control Loop With Step Controller Following a Step Change in the Setpoint

7.3 Example2: ContinuousController with Process Simulation

Application

Example2 encompasses a continuous standard controller (PID_CP) in combination with a simulated process which consists of a third order delay element (PT3).

Example2 is a simple example of how to generate a continuous PID controller and to configure and test it in all its properties in off-line mode with a typical process setup.

The example will help inexperienced users to understand how controllers with an analog output are used and configured in control systems involving processes with proportional actuators. This example can be used as an introduction or for training purposes.

After approximating the process to the characteristics of the real process by selecting suitable parameters, a set of controller characteristic data can be obtained by going through a process identification run using the configuration tool.

Functions of Example2

Example2 essentially consists of the two combined function blocks PID_CP (FB1) and PROC_C (FB100). PID_CP embodies the standard controller used and PROC_C simulates a third order self-regulating process (Figure 7-10).

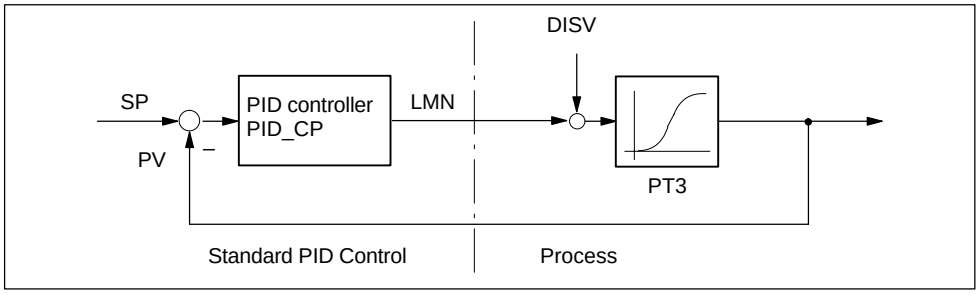


Figure 7-10 Example1, Control Loop

The function block PROC_C emulates a series connection which consists of three first order delay elements (Figure 7-11). The disturbance variable **DISV** is always added to the output signal of the final controlling element so that process disturbances can be feedforwarded manually here. The factor **GAIN** can be used to determine the static process gain.

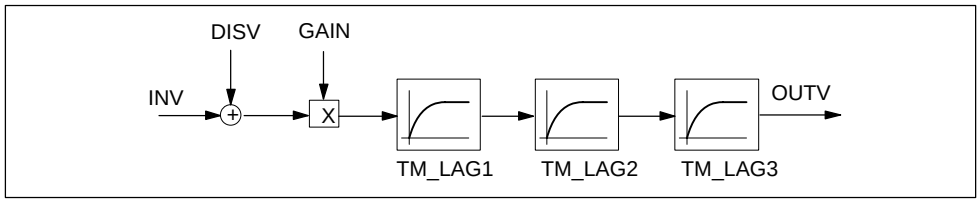


Figure 7-11 Structure and Parameters of the Process Block PROC_C

Block Structure

Example2 is put together from the function APP_2, which encompasses the blocks for the simulated process as well as the call blocks for a complete restart (OB100) and a watchdog interrupt level (OB35 with 100 ms cycle).

Table 7-3 Blocks for Example2

Block	Name (in the symbol bar)	Description
OB100		Complete restart OB
OB35		Time-driven OB: 100 ms
FC100	APP_2	Example 2
FB1	PID_CP	Continuous PID controller
FB100	PROC_C	Process for a continuous controller
DB100	PROCESS	Instance DB for PROC_C
DB101	CONTROL	Instance DB for PID_CP

The two function blocks (Figure 7-12) are assigned the instance data blocks DB100 for the process and DB101 for the controller.

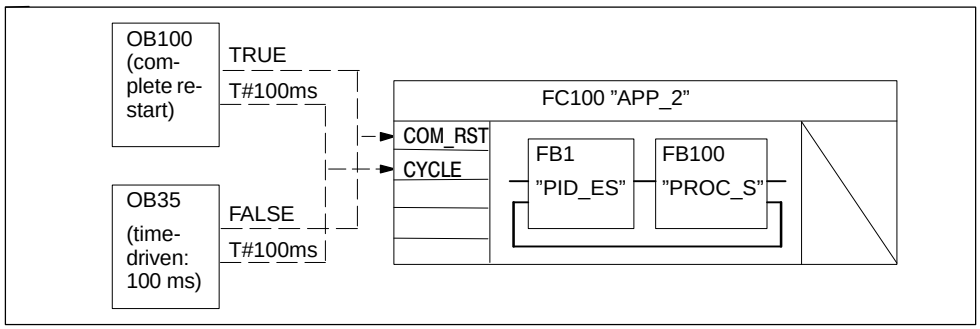


Figure 7-12 Blocks for Example2 Interconnection and Calling

The Parameters of the Process Model

The parameters of the control block PID_CP and their meaning are described in Chapter 6 The parameters for the process block PROC_C are listed in the following table.

Table 7-4 Parameters of the Process Block "PROC_C" (DB100: FB100)

Parameter	Type	range of values	Description
INV	REAL		Input value
COM_RST	BOOL		Complete restart
CYCLE	TIME	≥ 1ms	Sampling time
DISV	REAL		Disturbance variable
GAIN	REAL		Loop gain factor
TM_LAG1	TIME	≥ CYCLE/2	Time lag 1
TM_LAG2	TIME	≥ CYCLE/2	Time lag 2
TM_LAG3	TIME	≥ CYCLE/2	Time lag 3
OUTV	REAL		Output variable

Interconnection of and Calling Example2

Figure 7-13 shows how the continuous controller is interconnected internally via the function FC100 with the process model to a control loop.

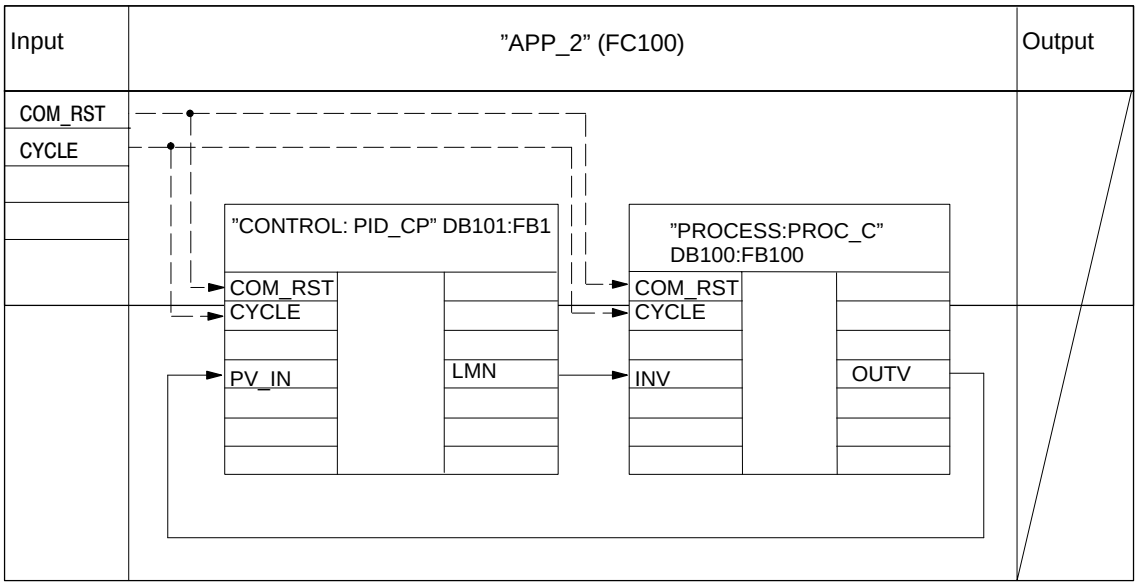
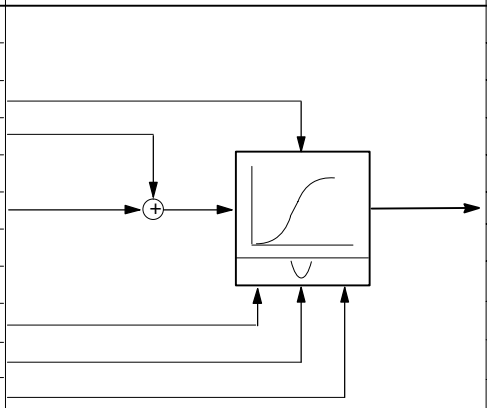


Figure 7-13 Connecting and Calling FC100 (APP_2)

Parameters of the Model Process for Continuous Controllers

Figure 7-14 shows the function scheme and the parameters of the process.

At a **complete restart** or a **warm restart** the closed-loop control behaves as described in Section 3.5.

Input Parameter			PROC_C (FB100)	Output Parameter		
Parameter	**Type**	***)**		**Signal**	**Type**	***)**
COM_RST	BOOL	FALSE				
CYCLE	TIME	T#1s				
GAIN	REAL	0.0				
DISV	REAL	0.0				
INV	REAL	0.0		OUTV	REAL	0.0
TM_LAG1	TIME	T#10s				
TM_LAG2	TIME	T#10s				
TM_LAG3	TIME	T#10s				

*) Default when the instance DB is created

Figure 7-14 Functions and Parameters of the Process Model PROC_C

Parameters and Step Response

The reaction of a control loop with a simulated third order PT process is shown on the basis of a concrete configuration of the continuous controller with PID action. The process parameters selected with a 10 sec. time lag approximate the response of a pressure control or a tank level control.

Setting one of the time lags TM_LAGx = 0 sec. reduces the process from third to second order.

The curve (configuration tool) illustrates the transfer and settling response of the closed loop after a series of setpoint changes of 20% of the measuring range (Figure 7-15). The table contains the values set for the relevant parameters of the controller and process.

Parameter	Type	Parameter Assignment	Description
Controller:			
CYCLE	TIME	100ms	Sampling time
GAIN	REAL	0.31	Proportional gain
TI	TIME	22.720s	Reset time
TD	TIME	5.974s	Derivative action time
TM_LAG	TIME	1.195s	Time lag of the D component
Process			
GAIN	REAL	1.5	Loop gain
TM_LAG1	TIME	10s	Time lag 1
TM_LAG2	TIME	10s	Time lag 2
TM_LAG3	TIME	10s	Time lag 3

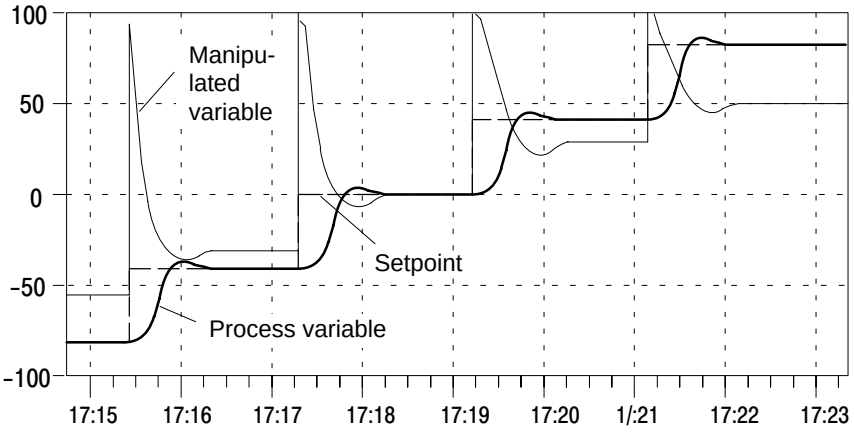


Figure 7-15 Controlling With a Continuous Controller and Setpoint Step Changes Over the Entire Measuring Range

7.4 Example3: Multi-loop Ratio Control

Application

Example3 contains all the blocks required to configure a two-loop ratio control.

Example3 provides a simple example of generating a ratio control for two components as it is often used in combustion processes. The structure can easily be extended to create a controller for more than two process variables with a constant ratio.

Functions of Example3

Example3 encompasses the loop scheduler (LP_SCHED) with the corresponding shared data block (DB-LOOP) as well as the function block (FB1) for continuous standard controllers with two instance DBs for the configuration data of the two controllers.

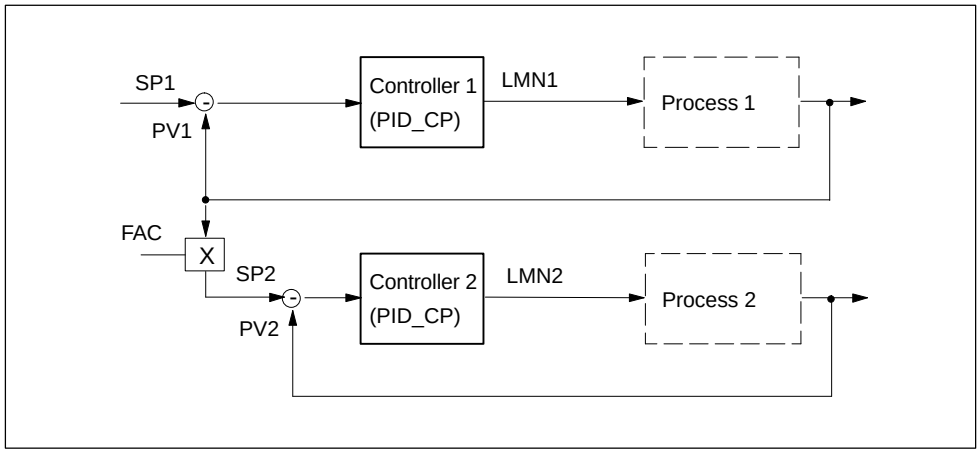


Figure 7-16 Ratio Control With Two Loops (Example 3)

The controllers (Figure 7-16) are called by the loop scheduler from the cyclic interrupt class with a 100 ms time base at fixed points in the cycle.

Controller 1 acts as the primary controller for setting the setpoint to control the second process variable. The ratio between PV1 and PV2 therefore also remains constant when process variable PV 1 fluctuates due to disturbances.

Block Structure

Example3 is put together from the function APP_3, which encompasses the blocks for the loop scheduler and the two controllers as well as the call blocks for a complete restart (OB100) and a watchdog interrupt level (OB35 with 100 ms cycle).

Table 7-5 Blocks for Example3

Block	Name (in the symbol bar)	Description
OB100		Complete restart OB
OB35		Time-driven OB: 100 ms
FC100	APP_3	Example 3
FC1	LP_SCHED	Loop scheduler
FB1	PID_CP	Continuous PID controller
DB1	DB_LOOP	Shared DB for call data for LP_SCHED
DB100	CONTROL1	1st Instance DB for PID_CP
DB101	CONTROL2	2nd Instance DB for PID_CP

The two instance data blocks DB100 and DB101 for realizing two-loop ratio controls are assigned to the function block PID_CP (FB1).

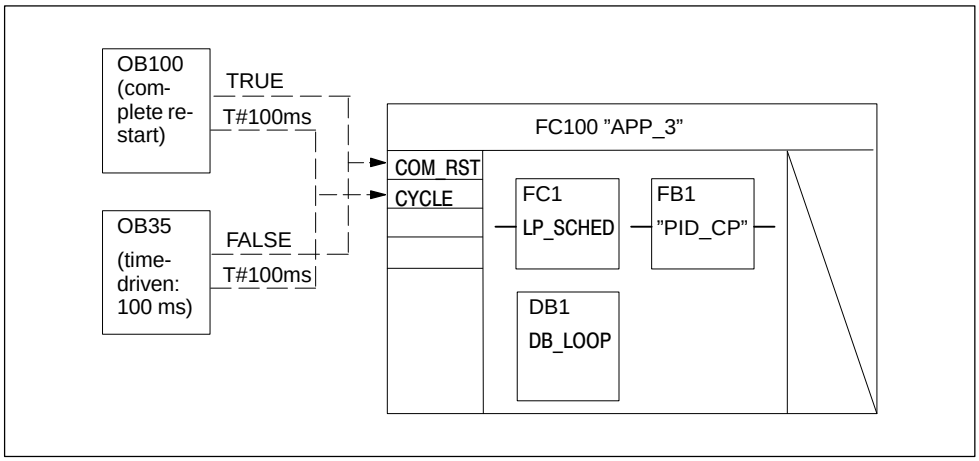


Figure 7-17 Blocks for Example3 Interconnection and Calling

Configuration of Example3

Figure 7-18 shows how the PID controllers are interconnected internally via the function FC100 with the loop scheduler and with each other.

At a **complete restart** or a **warm restart** the closed-loop control behaves as described in Section 3.5.

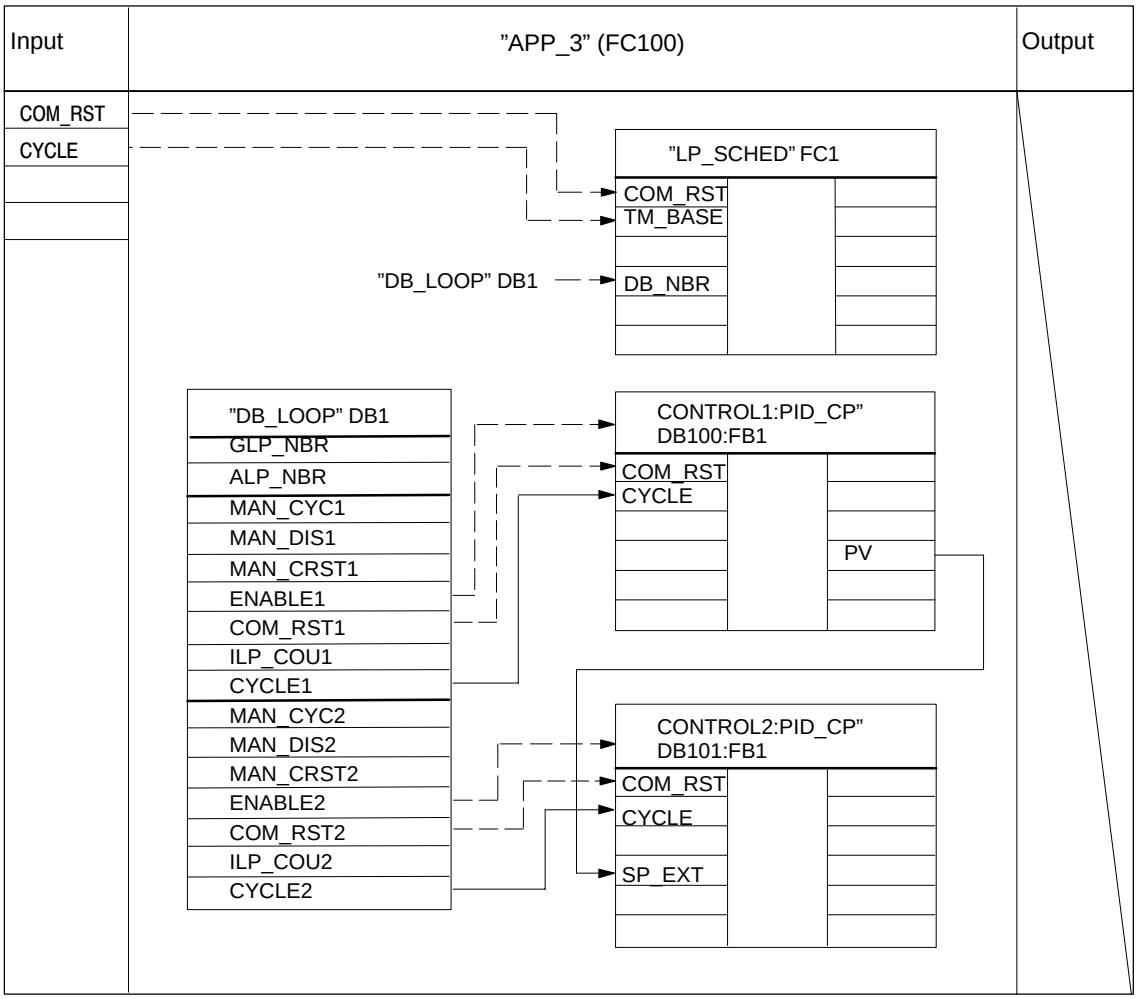


Figure 7-18 Circuit Diagram and Parameters for the FC Block APP_3

7.5 Example4: Blending Control

Application

Example4 contains all the blocks required to configure a blending control with one main and two secondary components.

Example4 is a simple example of how to generate a controller, required for blending processes, for the total quantity with constant shares of the individual quantities (for three components) which are used in the blend. The structure can be extended easily to include more than three components.

Functions of Example4

Example4 encompasses the loop scheduler (LP_SCHED) with the corresponding shared data block (DB-LOOP) as well as the function block (FB1) for continuous standard controllers and the function block (FB2) for step controllers with four instance DBs for the configuration data of the four controllers.

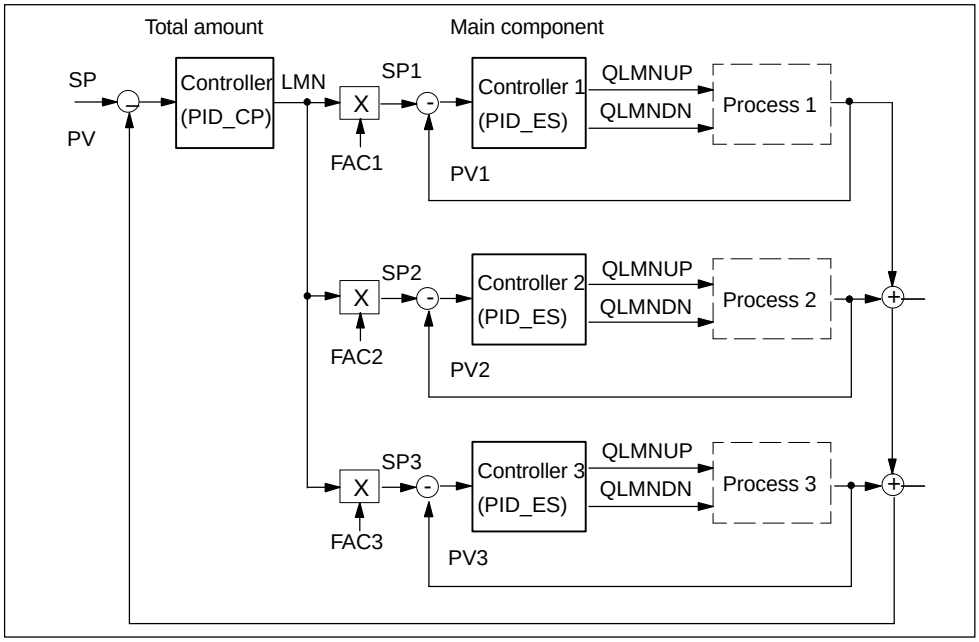


Figure 7-19 Blending Control for Three Components (Example4)

The four controllers are called using the loop scheduler in the cyclic interrupt class with a 100 ms time base at fixed points in the cycle. The controller for the total quantity with continuous output (PID_CP) acts as the master controller on the setting of the setpoint values, i.e. on the quantities of the respective components. The quantities of the main components and of the two secondary components are controlled in Example4 by step controllers (PID_ES) in accordance with the share settings at FAC1...3.

Remember that the values assigned to the blending factors FAC1 to FAC3 must add up to 100%.

Block Structure

Example4 is put together from the function APP_4, which encompasses the blocks for the loop scheduler and the four controllers as well as the call blocks for a complete restart (OB100) and a watchdog interrupt level (OB35 with 100 ms cycle).

Table 7-6 Blocks for Example4

Block	Name (in the symbol bar)	Description
OB100		Complete restart OB
OB35		Time-driven OB: 100 ms
FC100	APP_4	Example 4
FC1	LP_SCHED	Loop scheduler
FB1	PID_CP	Continuous PID controller
FB2	PID_ES	Step controller
DB1	DB_LOOP	Shared DB for call data for LP_SCHED
DB100	CONT_C1	Instance DB for PID_CP
DB101	CONT_S1	1stInstance DB for PID_ES
DB102	CONT_S2	2ndInstance DB for PID_ES
DB103	CONT_S3	3rdInstance DB for PID_ES

Three instance data blocks (DB101, DB 102 and DB103) for realizing the quantity controls of the three individual components are assigned to the function block PID_ES (FB2).

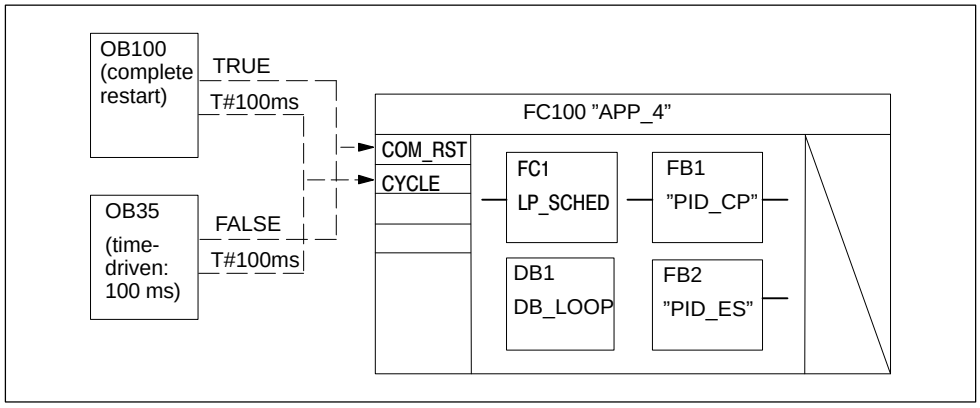


Figure 7-20 Blocks for Example4 Connection and Call

Configuration of Example4

Figure 7-21 shows how the controllers are interconnected internally via the function FC100 with the loop scheduler and with each other.

At a **complete restart** or a **warm restart** the closed-loop control behaves as described in Section 3.5.

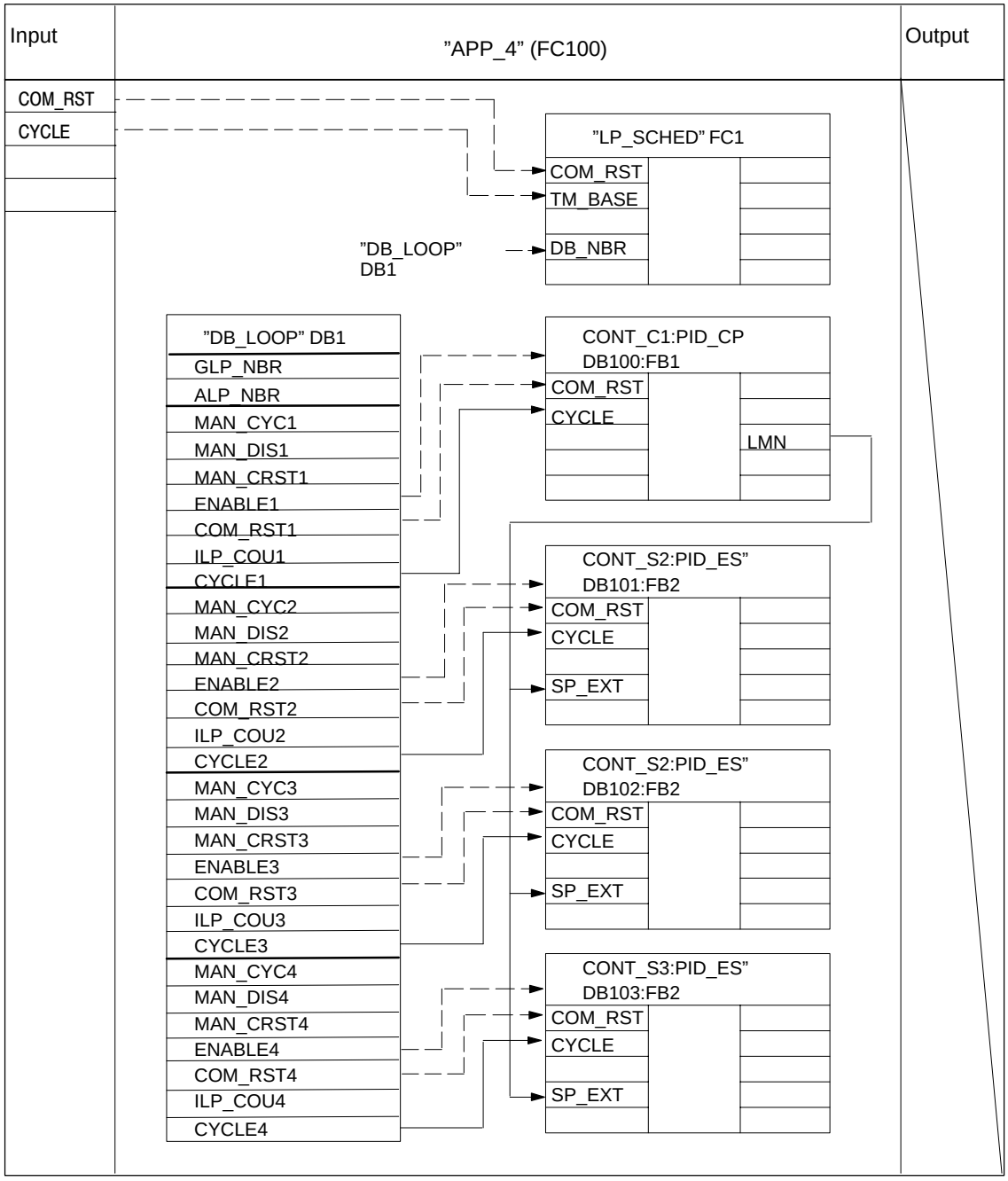


Figure 7-21 Block Diagram and Parameters of the FC Block APP_4

7.6 Example5: Cascade Control

Application

Example5 contains all the blocks required to configure a cascade control with one main and one secondary component.

Example5 provides a simple example of generating a cascade control with one master and one follower loop. The structure can be easily extended to include more than one secondary loop.

Functions of Example5

Example5 encompasses the loop scheduler (LP_SCHED) with the corresponding shared data block (DB-LOOP), the function block (FB1) for the continuous standard controller (master controller) as well as FB2 for the step controller (secondary controller) with the two instance DBs for the configuration data of the controllers.

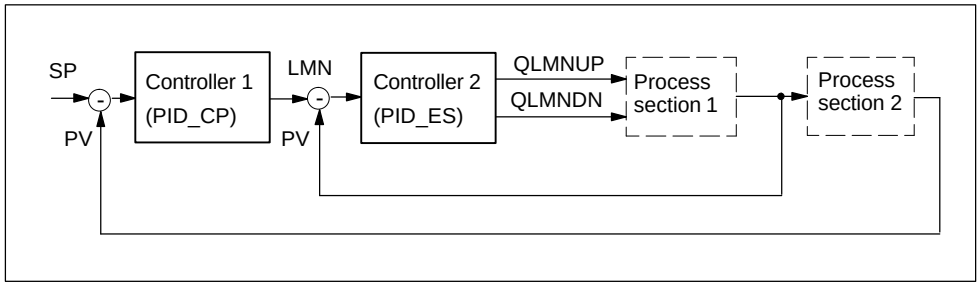


Figure 7-22 Two-Loop Cascade Control System (Example 5)

The controllers are called cyclically by the loop scheduler from within the cyclic interrupt class with a 100 ms time base.

The controller with the continuous output (PID_CP) acts as the master controller on the setpoint value of the secondary controller so that the main control variable at the output of process section 2 is held to the reference variable SP. Disturbances acting on control section 1 are controlled by the step controller in the secondary control loop (PID_ES) without influencing the main reference variable PV.

Block Structure

Example5 is put together from the function APP_5, which encompasses the blocks for the loop scheduler and the two controllers as well as the call blocks for a complete restart (OB100) and a watchdog interrupt level (OB35 with 100 ms cycle).

Table 7-7 Blocks for Example5

Block	Name (in the symbol bar)	Description
OB100		Complete restart OB
OB35		Time-driven OB: 100 ms
FC100	APP_5	Example 5
FC1	LP_SCHED	Loop scheduler
FB1	PID_CP	Continuous PID controller
FB2	PID_ES	Step controller
DB1	DB_LOOP	Shared DB for call data for LP_SCHED
DB100	CONT_C	Instance DB for PID_CP
DB101	CONT_S	Instance DB for PID_ES

The instance data blocks DB100 and DB101 respectively are assigned to the function blocks PID_CP and PID_ES.

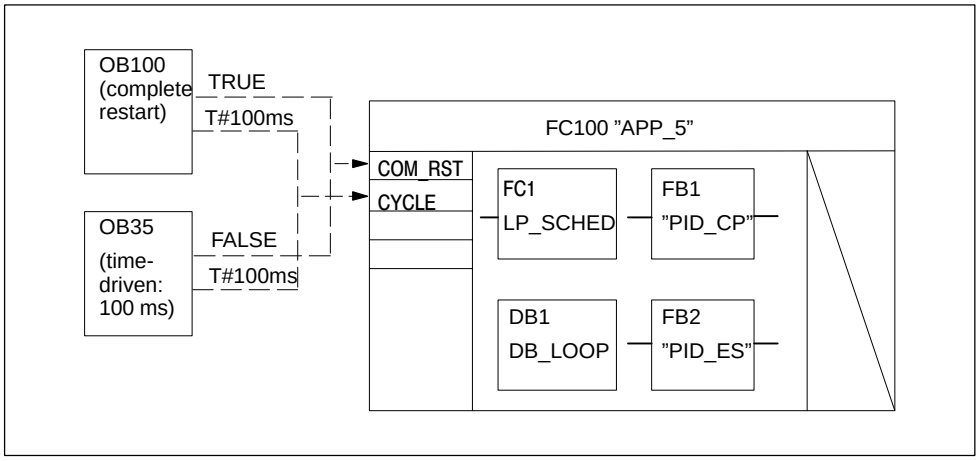


Figure 7-23 Blocks for Example5 Interconnection and Calling

Configuration of Example5

Figure 7-24 shows how the controllers are interconnected internally via the function FC100 with the loop scheduler and with each other.

At a **complete restart** or a **warm restart** the closed-loop control behaves as described in Section 3.5.

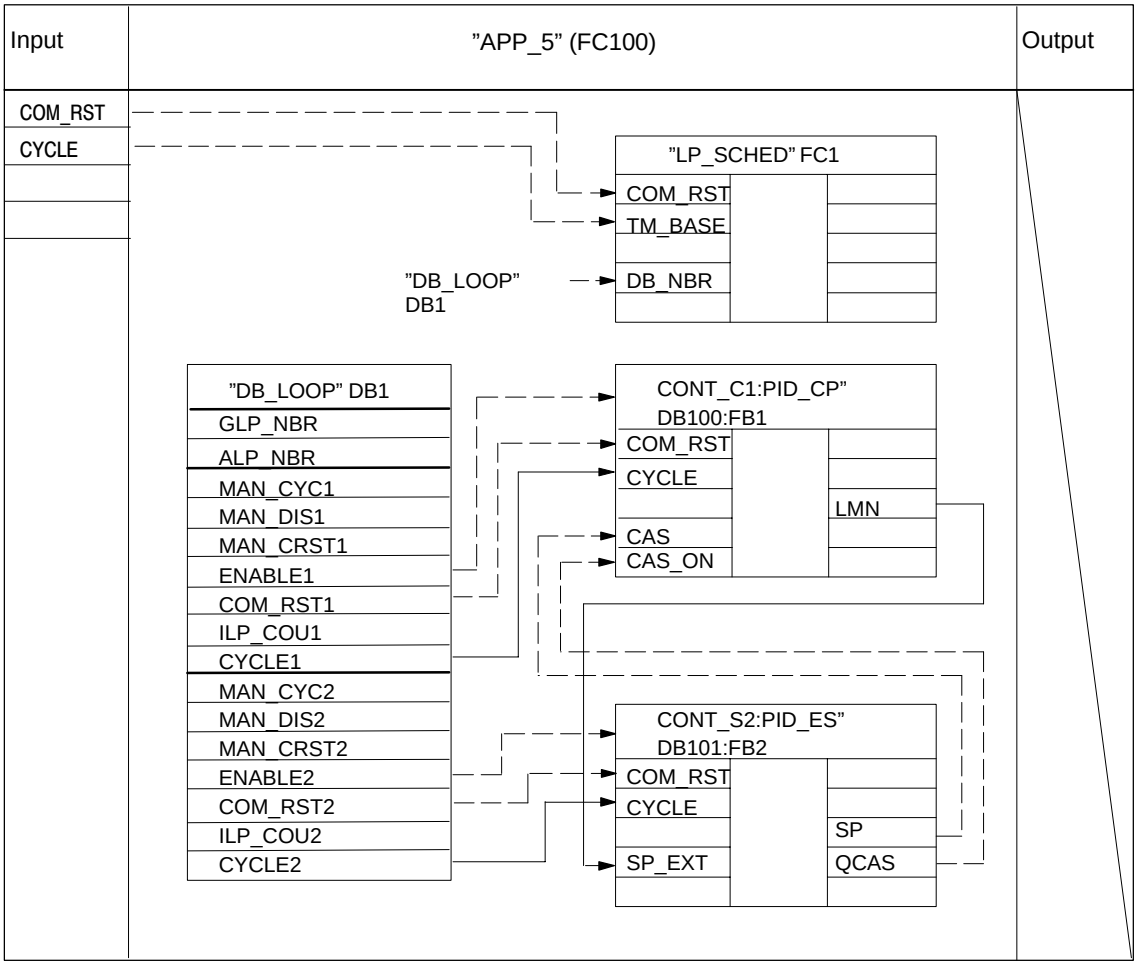


Figure 7-24 Block Diagram and Parameters of the FC Block APP_5

7.7 Example6: Pulsegen: Continuous Controller with Pulse Outputs and Process Simulation

Application

The example (Pulsegen) encompasses a continuous controller (PID_CP) with a positive and negative pulse output in combination with a simulated process, which consists of a third order delay element (PT3).

Pulsegen is a simple example of how to generate a continuous PID controller with pulse outputs and to configure and test it in all its properties in off-line mode with a typical process setup.

The example will help inexperienced users to understand how controllers with binary pulse outputs are used and configured in control systems involving processes with proportional actuators. Such controllers are used, for example, for temperature controls with electrical heating.This example can be used as an introduction or for training purposes.

After approximating the process to the characteristics of the real process by selecting suitable parameters, a set of controller characteristic data can be obtained by going through a process identification run using the configuration tool.

Functions of Example 6

Example6 essentially consists of the two combined function blocks PID_CP (FB1) and PROC_CP (FB100). PID_CP embodies the controller used including pulse generators, and PROC_CP simulates a third order self-regulating process (Figure 7-25).

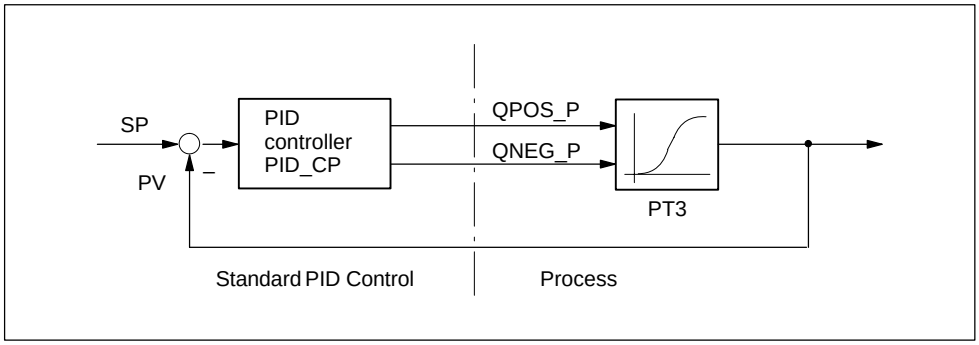


Figure 7-25 Example6, Control Loop

The function block PROC_CP emulates a series connection which consists of three first order delay elements (Figure 7-26). Not only the pulse inputs POS_P and NEG_P act as input signals for the process, but also the disturbance variable **DISV** as an additional input signal so that process disturbances can be feedforwarded manually at this point. The factor **GAIN** can be used to determine the static process gain.

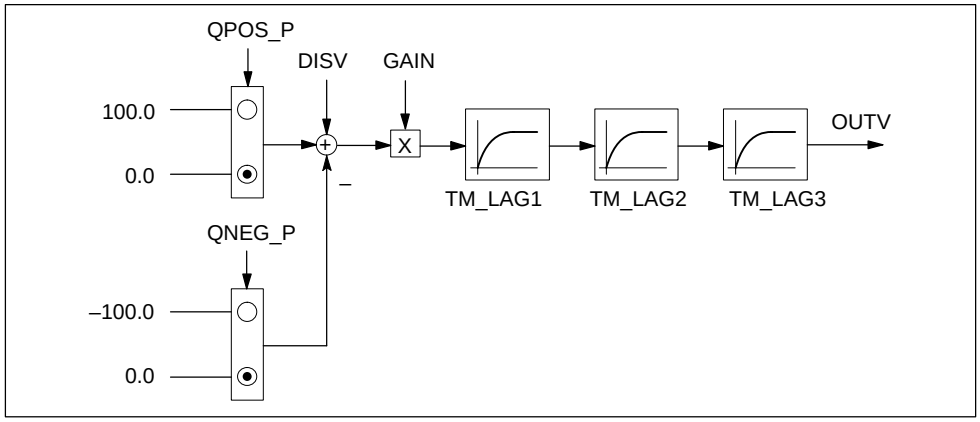


Figure 7-26 Structure and Parameters of the Process Block PROC_CP

Block Structure

Example6 is put together from the function APP_Pulsegen, which encompasses the blocks for the two controller and the simulated process as well as the call blocks for a complete restart (OB100) and a watchdog interrupt level (OB35 with 100 ms cycle).

Table 7-8 Blocks for Example 6

Block	Name (in the symbol bar)	Description
OB100	RESTART	Complete restart OB
OB35	CYC_INT1	Time-driven OB: 100 ms
FC100	APP_Pulsegen	Example 6
FB1	PID_CP	Continuous PID controller with pulse generator
FB100	PROC_CP	Process for continuous controller with pulse inputs
DB100	PROCESS	Instance DB for PROC_C
DB101	CONTROL	Instance DB for PID_CP

The two function blocks (Figure 7-27) are assigned to the instance data blocks PROCESS DB100 for the process and CONTROL DB 101 for the controller.

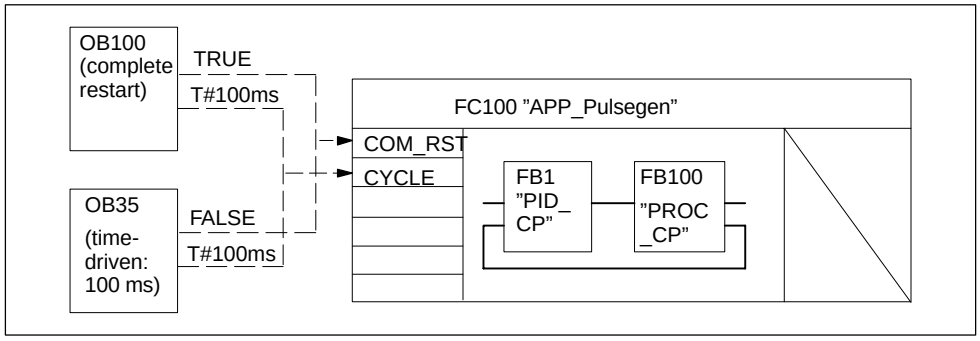


Figure 7-27 Blocks for Example6 Interconnection and Calling

The Parameters of the Process Model

The parameters of the control block PID_CP and their meaning are described in Chapter 6 The parameters of the process block PROC_CP are listed in the following table.

Table 7-9 Parameters of the Process Block "PROC_CP" (DB100: FB100)

Parameter	Type	range of values	Description
DISV	REAL		Disturbance variable
GAIN	REAL		Loop gain factor
TM_LAG1	TIME	≥ CYCLE/2	Time lag 1
TM_LAG2	TIME	≥ CYCLE/2	Time lag 2
TM_LAG3	TIME	≥ CYCLE/2	Time lag 3
POS_P	BOOL		Positive pulse
NEG_P	BOOL		Negative pulse
COM_RST	BOOL		Complete restart
CYCLE	TIME	≥ 1ms	Sampling time
OUTV	REAL		Output variable

Interconnection of and Calling Example6

Figure 7-28 shows how the continuous controller is interconnected internally via the function FC100 with the process model to a control loop.

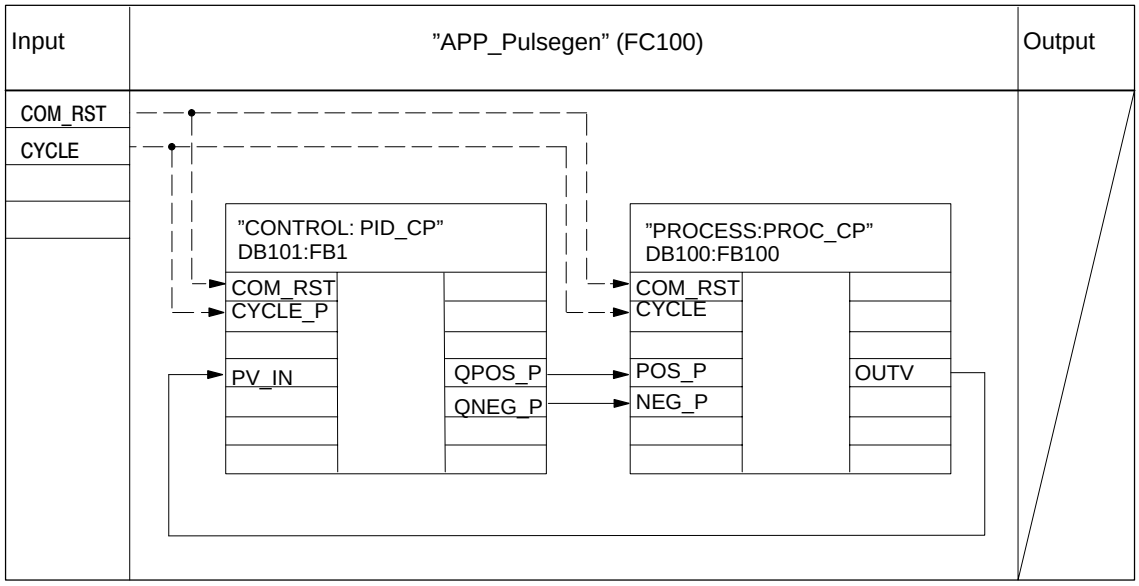
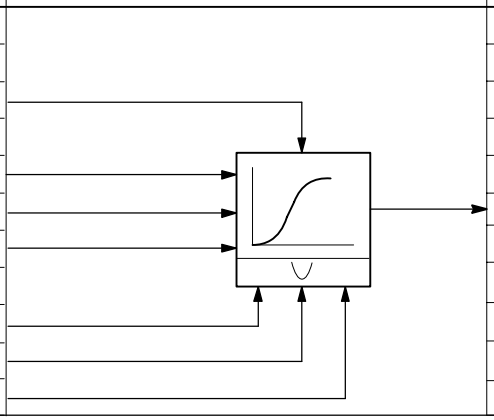


Figure 7-28 Interconnection and Calling the FC100 (APP_Pulsegen)

Parameters of the Model Process for Continuous Controllers

Figure 7-29 shows the function scheme and the parameters of the process.

At a **complete restart** or a **warm restart** the closed-loop control behaves as described in *Section3.5*.

Input Parameter			PROC_CP (FB100)	Output Parameter		
Parameter	**Type**	***)**		**Signal**	**Type**	***)**
COM_RST	BOOL	FALSE				
CYCLE	TIME	T#1s				
GAIN	REAL	0.0				
DISV	REAL	0.0				
POS_P	BOOL	FALSE		OUTV	REAL	0.0
NEG_P	BOOL	FALSE				
TM_LAG1	TIME	T#10s				
TM_LAG2	TIME	T#10s				
TM_LAG3	TIME	T#10s				

*) Default when the instance DB is created

Figure 7-29 Functions and Parameters of the PROC_CP Process Model

Parameters and Step Response

The reaction of a control loop with a simulated third order PT process is shown on the basis of a concrete configuration of the continuous controller with PID action. The selected loop parameters with a 10 sec. time lag realize a faster process than would be usual at a temperature control. However the relatively fast process means that the function of the controller can be tested faster.However the property of the simulated process can be approximated easily to a real process by changing the time constant of time delay.

The curve (configuration tool) illustrates the transfer and settling response of the closed loop after a series of setpoint changes of 20 % of the measuring range (Figure 7-30). The continuous manipulated variable of the controller is shown, not the pulse outputs. The table contains the values set for the relevant parameters of the controller and process.

Parameter	Type	Parameter Assignment	Description
Controller:			
CYCLE	TIME	1s	Sampling time of the controller
CYCLE_P	TIME	100ms	Sampling time
GAIN	REAL	1.535	Proportional gain
TI	TIME	22.720s	Reset time
TD	TIME	5.974s	Derivative action time
TM_LAG	TIME	1.195s	Time lag of the D component
Process:			
GAIN	REAL	1.5	Loop gain
TM_LAG1	TIME	10s	Time lag 1
TM_LAG2	TIME	10s	Time lag 2
TM_LAG3	TIME	10s	Time lag 3

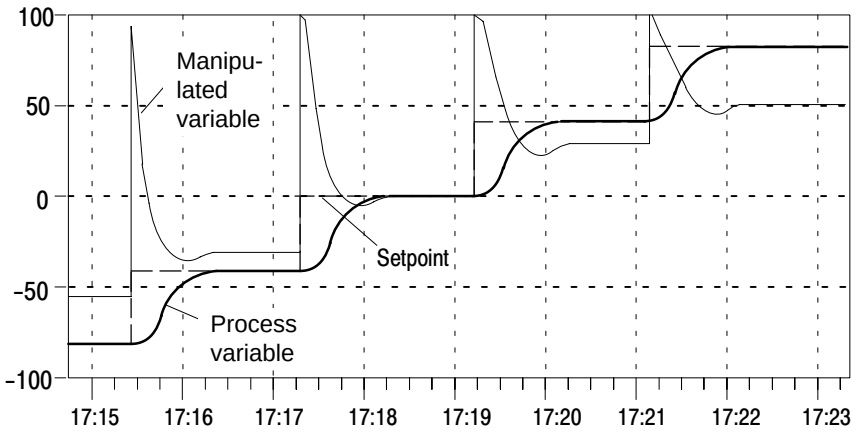


Figure 7-30 Controlling With a Continuous Controller with Pulse Outputs and Setpoint Step Changes Over the Entire Measuring Range

Technical Data and Block Diagrams

8

8.1 Technical Data: Function Blocks

CPU Load

To be able to estimate the load on a particular CPU resulting from installing the Standard PID Controls, you can use the following guidelines:

- The controller FB only needs to exist once in the user memory of the CPU for any number of controllers.
- Per controller one DB with approx. o,5 KBytes
- Data for typical run times (processing times) of the blocks when the default parameters are assigned for controller operation:

Block name	Boundary conditions	Processing time in [ms] 315-2AG10	Processing time in [ms] CPU 416-2XK02
PID_CP	Typical boundary conditions	1.3	0.14
PID_ES	Without position feedback, typical boundary conditions	1.5	0.16

Work Memory Used

The size of the area required in the user memory and therefore the number of control loops that could theoretically be installed with the available memory capacity can be seen in the following table:

Block name		Load memory required	User memory required	Local data
PID_CP	FB 1	8956 bytes	7796 bytes	122 bytes
PID_ES	FB 2	9104 bytes	7982 bytes	152 bytes
LP_SCHED	FC 1	1064 bytes	976 bytes	20 bytes

Instance DB or shared DB	Load memory required	User memory required
DB to PID_CP	1168 bytes	510 bytes
DB to PID_ES	1124 bytes	484 bytes
DB_LOOP (at 5 control loops)	184 bytes	100 bytes
DB_RMPSK (with a sart point and 4 time slices)	142 bytes	78 bytes

Sampling Time

The shortest selectable sampling time depends on the performance of the CPU being used.

Note

The limited accuracy in calculation restricts the sampling time that can be implemented. As the sampling time becomes smaller, the constants of the algorithms adopt smaller and smaller numerical values. This can lead to incorrect calculation of the manipulated variable.

Recommendation:

S7-300: sampling time ≥ 20 ms

S7-400: sampling time ≥ 5 ms

Calling the Controller

Depending on the sampling time, the function block for a particular control loop must be called at constant intervals. The operating system of the S7 PLC calls the cyclic interrupt OB.

The sampling time and cyclic interrupt time must match.

8.2 Block Diagrams of Standard PID Control

Conventions Used With Parameters and Field Names

A maximum of eight characters are used to identify the parameters and block names. This saves having to write long names when implementing controllers using STEP 7 STL or SCL and takes up less space on the monitor.

The names of the parameters are based largely on the IEC 1131-3 standard. The following conventions were used to name the parameters of the Standard PID Control:

SP	Setpoint	Setpoint value, reference variable
PV	Process variable	Actual value (measured value), process variable
ER	Error signal	Error signal
LMN	Manipulated variable	Manipulated variable (analog actuating signal to be output)
DISV	Disturbance variable	Disturbance variable
MAN	Manual value	Manual manipulated value
CAS	Cascade	Cascade
SQRT	Square root	Square root
.._ROC	Rate of change	Rate of change (slope)
Q..	(Q stands for 'O')	General output of type BOOL
.._INT	(internal value)	Internal
.._EXT	(external value)	external
.._ON		Boolean value = switching signal
..URLM	Up rate limit	Up rate limit
..DRLM	Down rate limit	Down rate limit

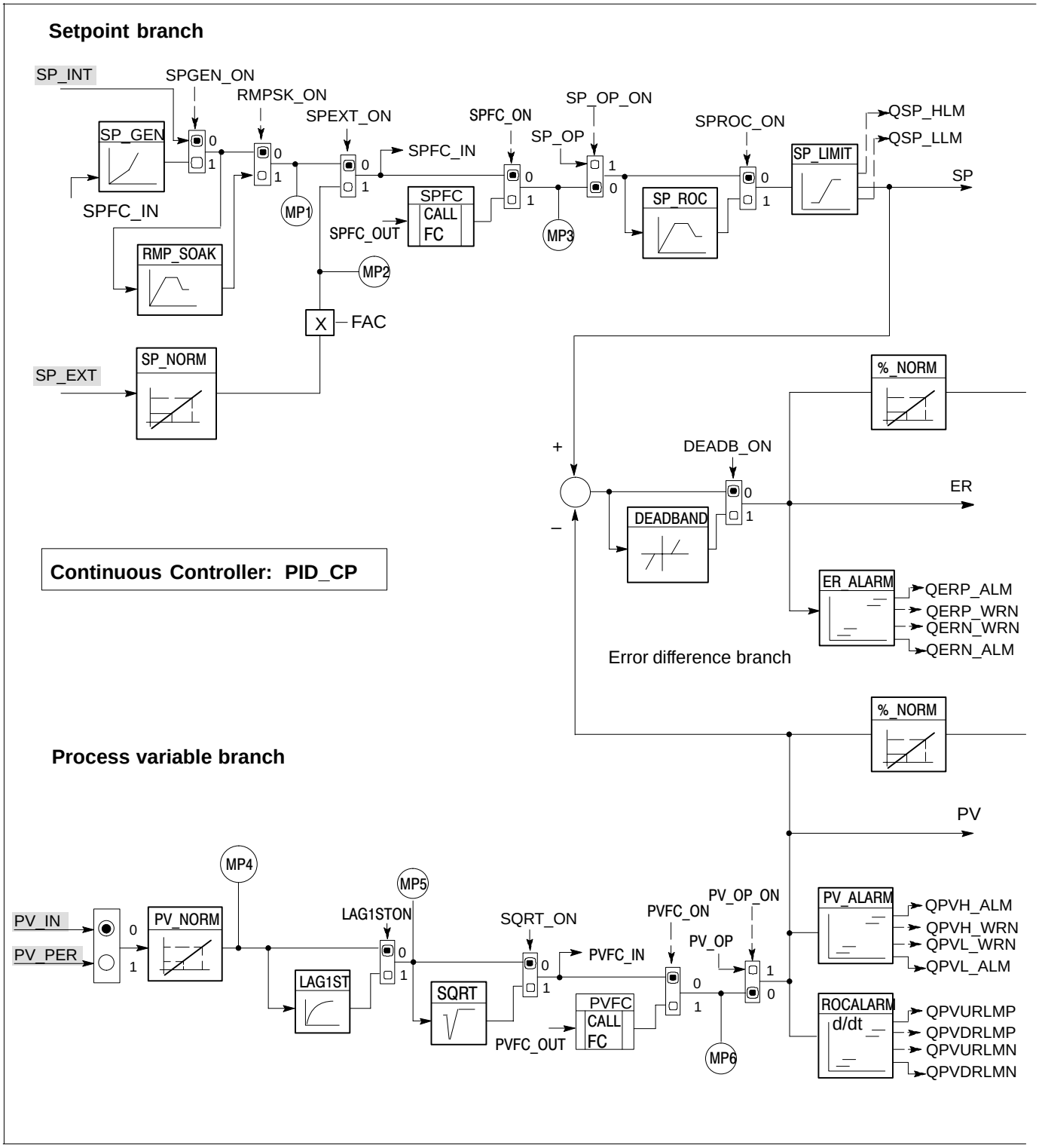
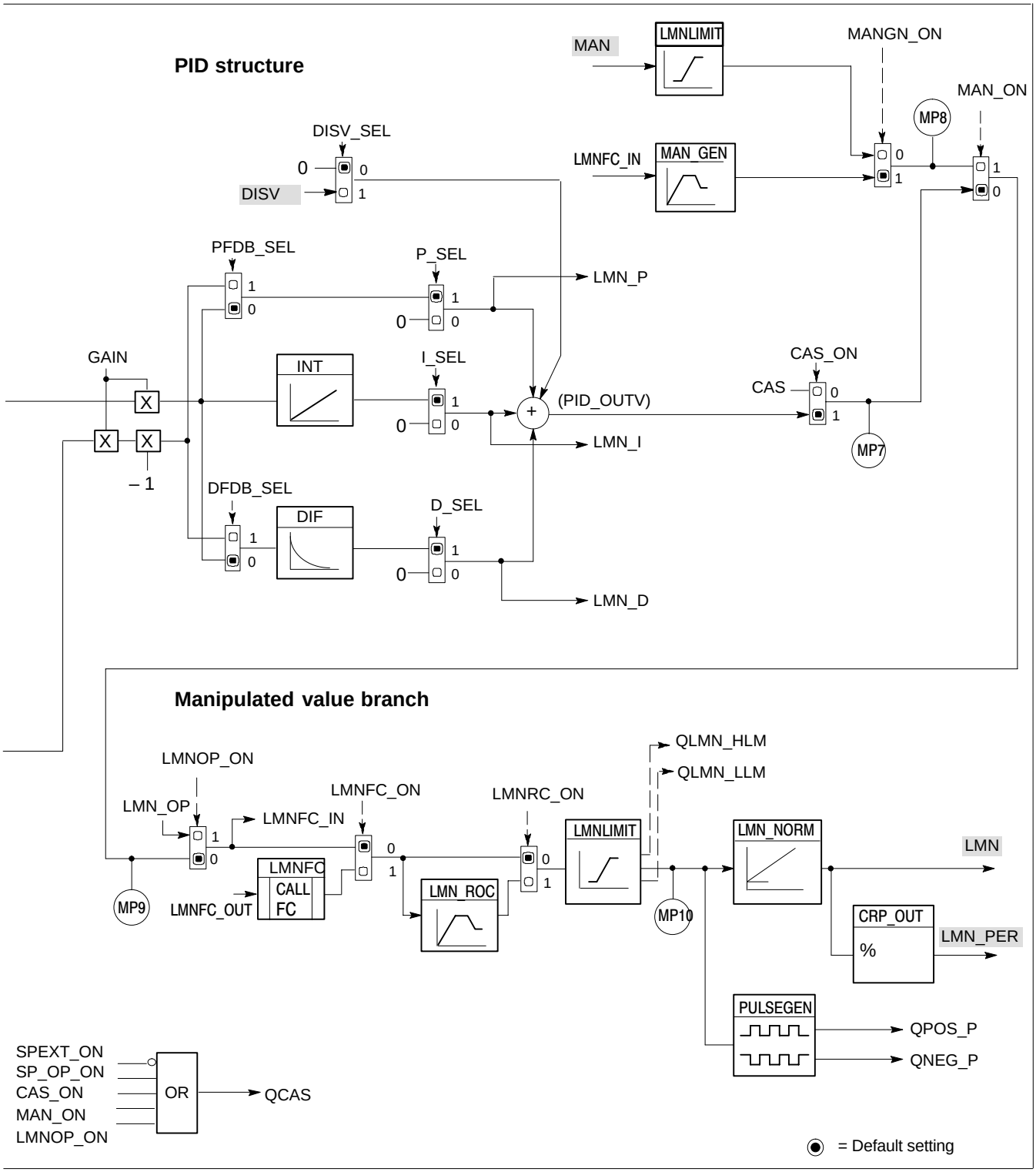


Figure 8-1 Block Diagram of the Continuous Controller: PID_CP


PID structure
MAN
LMNLIMIT
MANGN_ON
MAN_ON
MP8
DISV_SEL
DISV
LMNFC_IN
MAN_GEN
PFDB_SEL
P_SEL
LMN_P
GAIN
INT
I_SEL
CAS_ON
CAS
(PID_OUTV)
LMN_I
MP7
– 1
DFDB_SEL
DIF
D_SEL
LMN_D
Manipulated value branch
LMNOP_ON
LMN_OP
LMNFC_ON
LMNRC_ON
QLMN_HLM
QLMN_LLM
LMNFC
CALL
FC
LMNFC_OUT
LMN_ROC
LMN_NORM
LMN
MP9
MP10
CRP_OUT
%
LMN_PER
PULSEGEN
QPOS_P
QNEG_P
SPEXT_ON
SP_OP_ON
CAS_ON
MAN_ON
LMNOP_ON
OR
QCAS
= Default setting

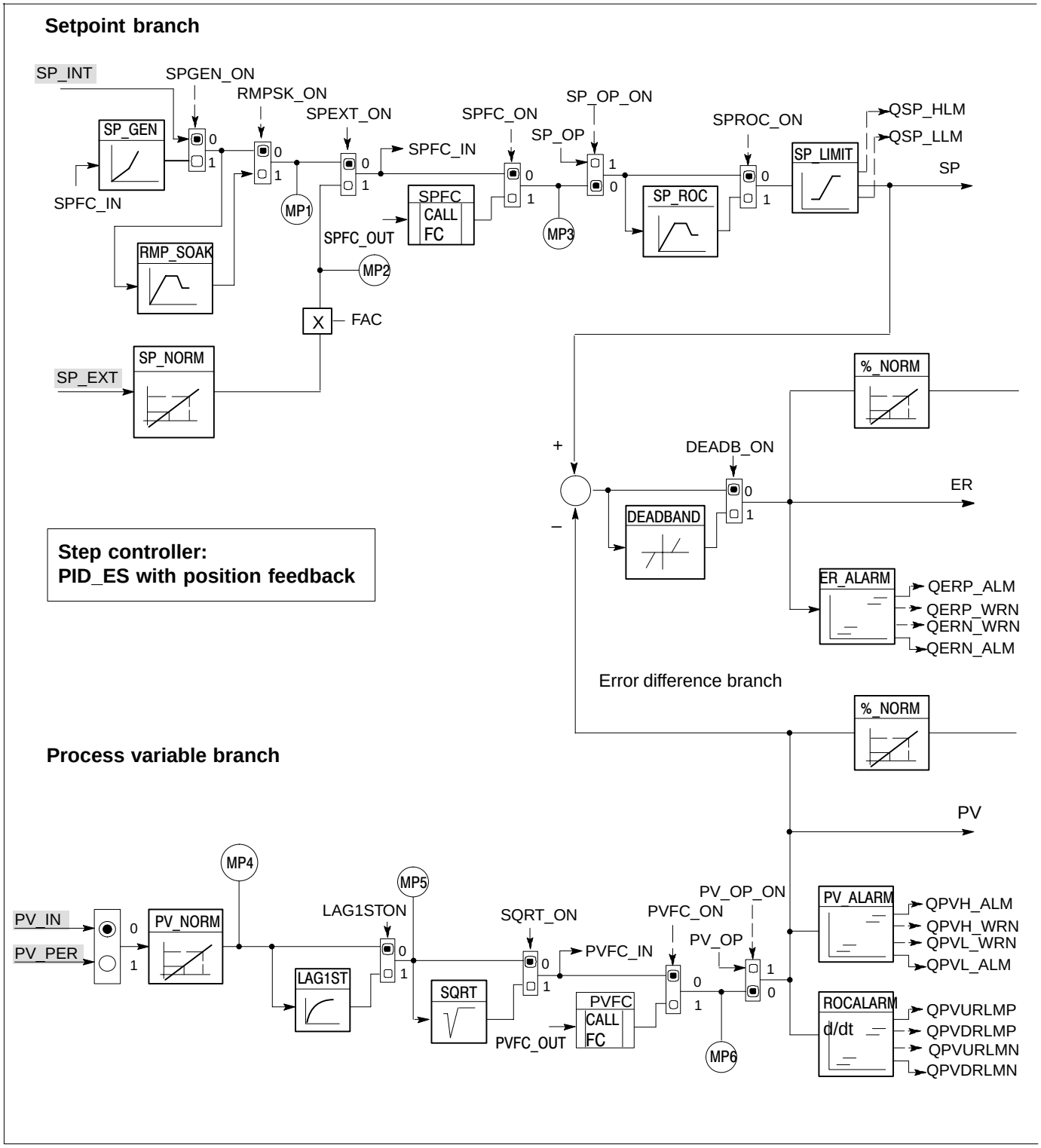
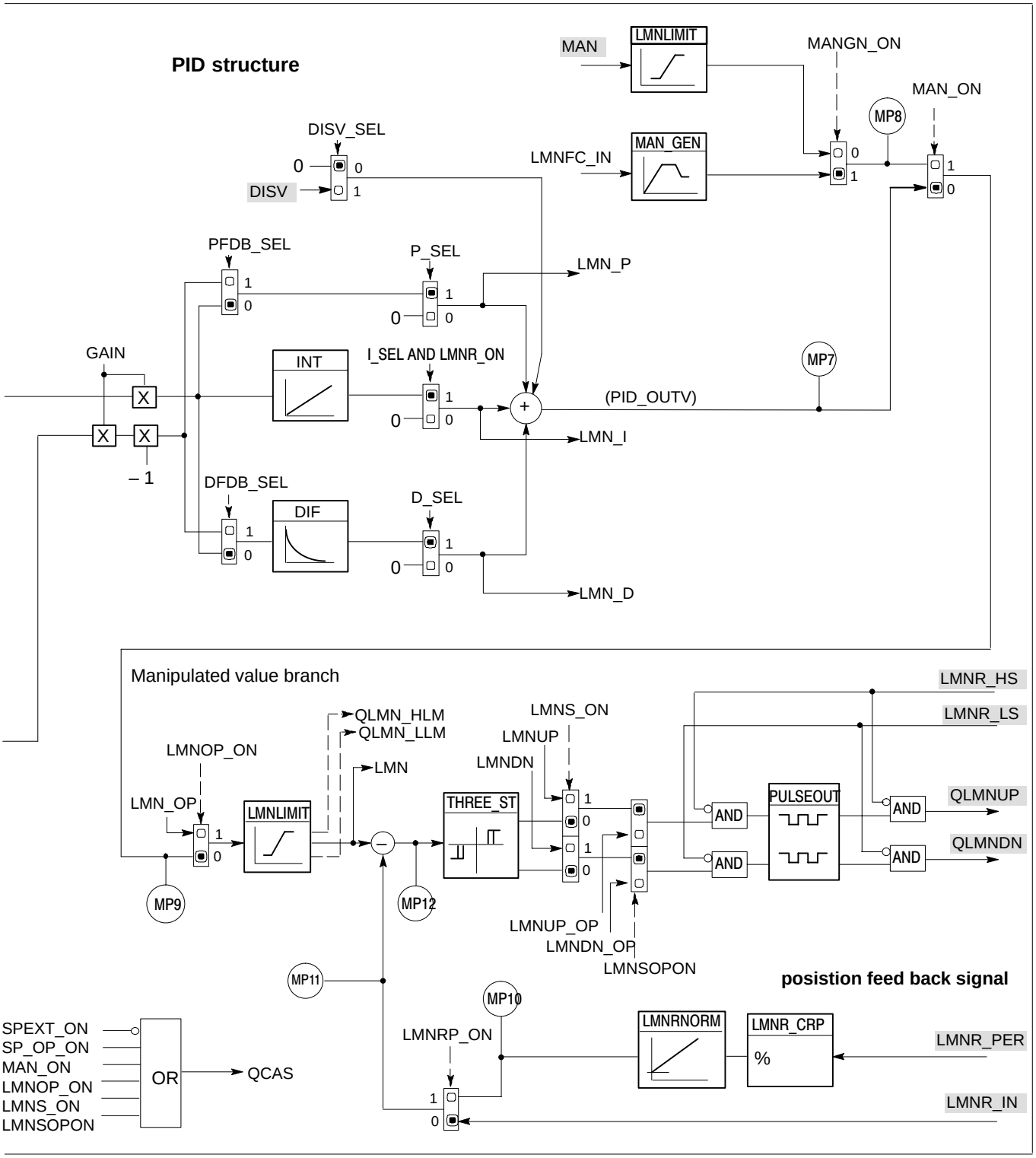


Figure 8-2 Block Diagram of the Step Controller: PID_ES (with position feedback signal “LMNR = TRUE”)


PID structure
MAN
LMNLIMIT
MANGN_ON
MAN_ON
MP8
DISV_SEL
DISV
LMNFC_IN
MAN_GEN
PFDB_SEL
P_SEL
LMN_P
GAIN
INT
I_SEL AND LMNR_ON
MP7
(PID_OUTV)
LMN_I
– 1
DFDB_SEL
DIF
D_SEL
LMN_D
Manipulated value branch
LMNR_HS
LMNR_LS
QLMN_HLM
QLMN_LLM
LMNS_ON
LMNUP
LMNDN
LMNOP_ON
LMN
LMN_OP
LMNLIMIT
THREE_ST
AND
PULSEOUT
QLMNUP
QLMNDN
MP9
MP12
LMNUP_OP
LMNDN_OP
LMNSOPON
posistion feed back signal
MP11
MP10
LMNRP_ON
LMNRNORM
LMNR_CRP
%
LMNR_PER
LMNR_IN
SPEXT_ON
SP_OP_ON
MAN_ON
LMNOP_ON
LMNS_ON
LMNSOPON
OR
QCAS

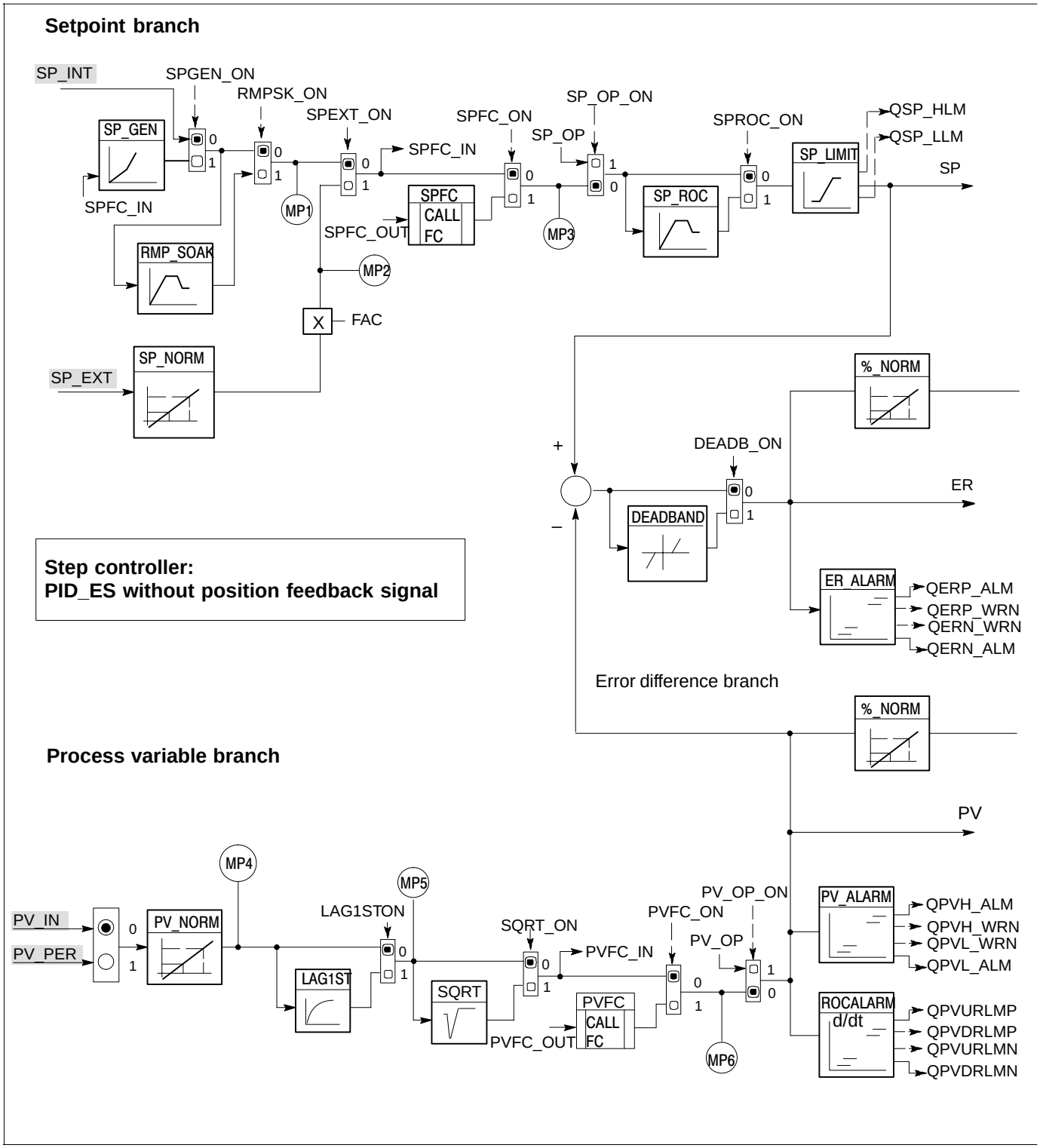
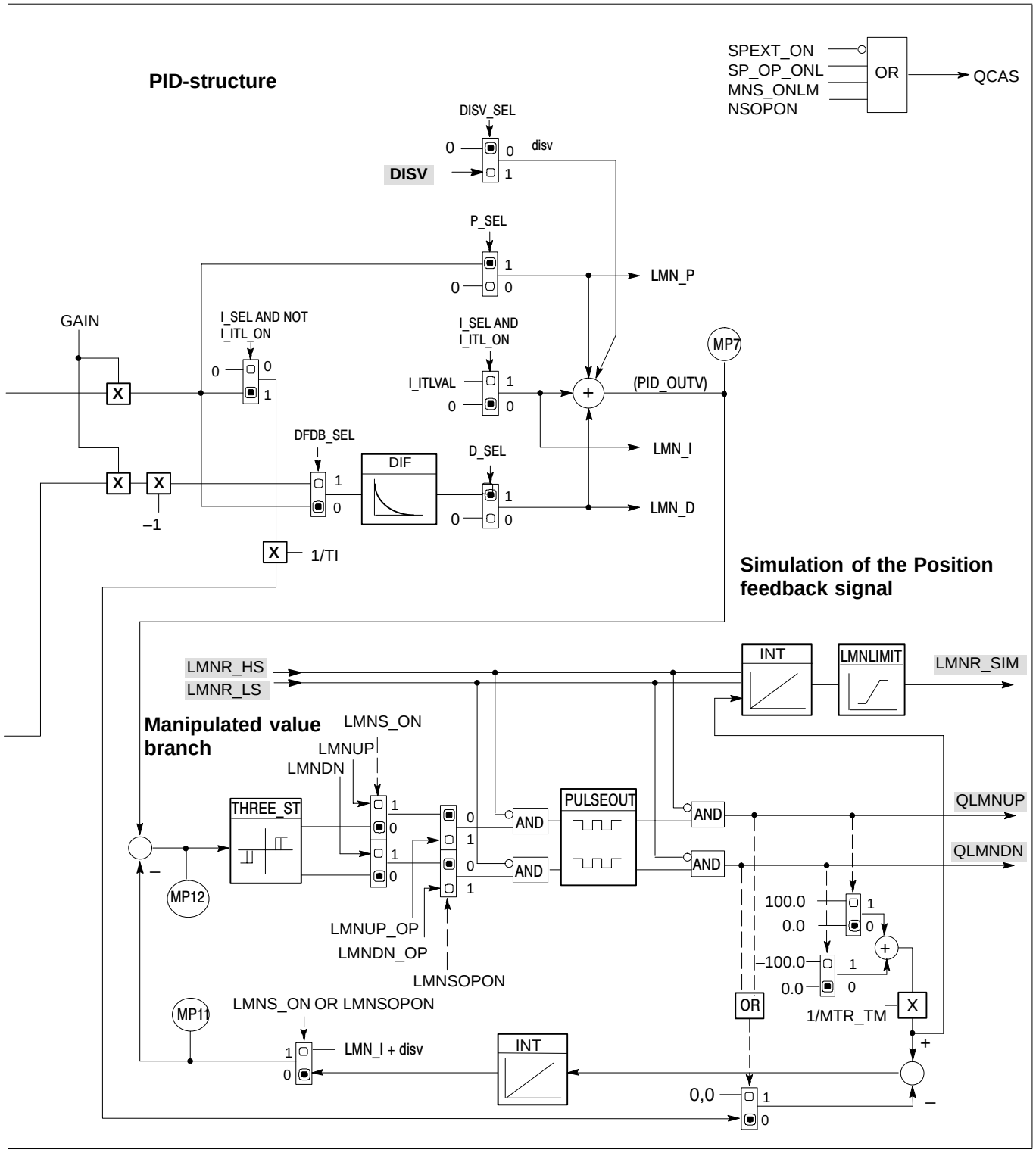


Figure 8-3 Block Diagram of the Step Controller: PID_ES (without position feedback signal “LMNR_ON = FALSE”)


PID-structure
SPEXT_ON
SP_OP_ONL
MNS_ONLM
NSOPON
OR
QCAS
DISV_SEL
DISV
disv
P_SEL
LMN_P
GAIN
I_SEL AND NOT I_ITL_ON
I_SEL AND I_ITL_ON
I_ITLVAL
MP7
(PID_OUTV)
LMN_I
DFDB_SEL
DIF
D_SEL
LMN_D
–1
1/TI
Simulation of the Position feedback signal
LMNR_HS
LMNR_LS
INT
LMNLIMIT
LMNR_SIM
Manipulated value branch
LMNS_ON
LMNUP
LMNDN
THREE_ST
PULSEOUT
AND
QLMNUP
QLMNDN
MP12
LMNUP_OP
LMNDN_OP
LMNSOPON
100.0
0.0
–100.0
1/MTR_TM
MP11
LMNS_ON OR LMNSOPON
LMN_I + disv
0,0

Parameter Lists of the Standard PID Control

9

Note

- The parameter lists in this appendix represent the order and content of the structures in the instance blocks of the SIMATIC S7 standard function blocks.
- The **range of values** is shown for each parameter.

  ”Entire range of values” means: the numerical range fixed for the particular STEP 7 address type.

  ”Technical range of values” means: a restricted range which represents reality with adequate accuracy, here -10^5 to $+10^5$. This avoids awkward large or small numerical ranges for the parameters.
- All the parameters have the specified **default** value when the instance DB is created.

  These values have been selected so that it is unlikely that a critical state can arise if they are used as they stand.

  Using the STEP 7 program editor, you can change the default to any other value in the **permitted range of values**. It is, however, more convenient to use the configuration tool with its parameter assignment functions.
- For the conventions used in naming the parameters, refer to *Section 8.2*.

9.1 Parameters of the PID_CP Function Block

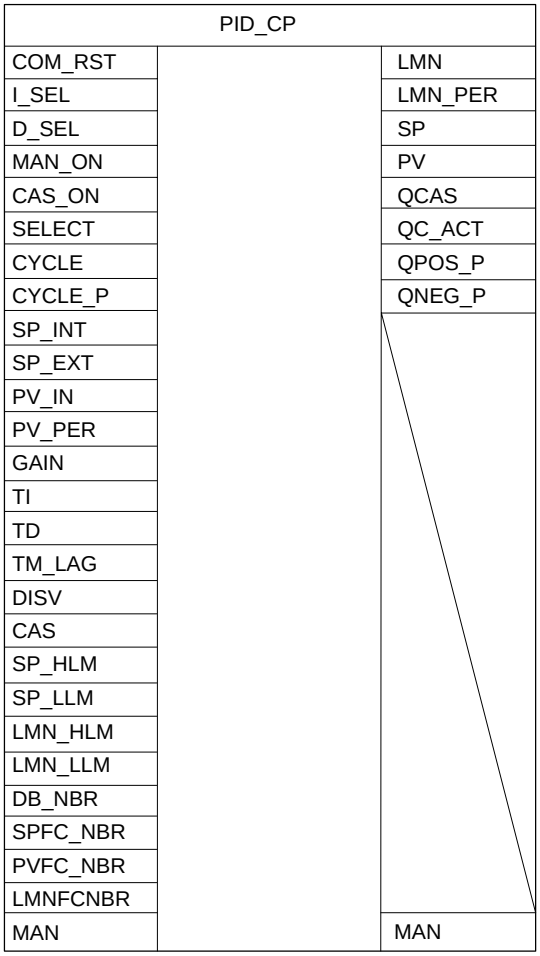


Table 9-1 Input Parameters of PID_CP (continuous controller)

Parameter	Data Type	Explanation	Permitted range of values	Default
COM_RST	BOOL	Complete restart (initialization routine of the FB is processed)		FALSE
I_SEL	BOOL	I action on		TRUE
D_SEL	BOOL	D action on		FALSE
MAN_ON	BOOL	Manual mode on (loop opened, LMN set manually)		TRUE
CAS_ON	BOOL	Cascade control on (connected to QCAS of the secondary controller)		FALSE

Table 9-1 Input Parameters of PID_CP (continuous controller)

Parameter	Data Type	Explanation	Permitted range of values	Default
SELECT	BYTE	If PULS_ON = TRUE: 0: PID and pulse generator 1: PID (block call in OB1) 2: Pulse generator (block call in watchdog-interrupt OB) 3: PID (block call in watchdog-interrupt OB)	0, 1, 2, 3	0
CYCLE	TIME	Sampling time (time between two block calls = constant) Be sure to configure this parameter with the watchdog-interrupt cycle of the OB in which the “PID_CP” FB runs! Otherwise the time-dependent functions do not function correctly. (Exception:You use a pulse scaling, for example via the controller call distribution.)	> 20 ms (S7-300)	T#1s
CYCLE_P	TIME	Sampling time of the pulse generator Be sure to configure this parameter with the watchdog-interrupt cycle of the OB in which the “PID_CP” FB runs! Otherwise the time-dependent functions do not function correctly. (Exception: You use a pulse scaling, for example via the controller call distribution.)		T#10ms
SP_INT	REAL	Internal setpoint (for setting the setpoint with operator interface functions)	Technical range of values (physical dimension)	0.0
SP_EXT	REAL	External setpoint (SP in floating-point format)	Technical range of values (physical dimension)	0.0
PV_IN	REAL	Process variable input (PV in floating-point format)	Technical range of values (physical dimension)	0.0
PV_PER	INT	Process variable from I/Os (PV in peripheral format)		W#16#0000
GAIN	REAL	Proportional gain (= controller gain)	Entire range of values (no dimension)	2.0
TI	TIME	Reset time	TI $\geq$ CYCLE	T#20s
TD	TIME	Derivative action time	TD $\geq$ CYCLE	T#10s
TM_LAG	TIME	Time lag of the D component	TM_LAG $\geq$ CYCLE/2	T#2s

Table 9-1 Input Parameters of PID_CP (continuous controller)

Parameter	Data Type	Explanation	Permitted range of values	Default
DISV	REAL	Disturbance variable	–100.0 ... 100.0	0.0
CAS	REAL	Input for cascade operation (connection to PV of secondary controller)	Technical range of values (physical dimension)	0.0
SP_HLM	REAL	Setpoint high limit	Technical range of values (physical dimension)	100.0
SP_LLM	REAL	Setpoint low limit	Technical range of values (physical dimension)	0.0
LMN_HLM	REAL	Manipulated value: high limit	LMN_LLM ... 100.0	100.0
LMN_LLM	REAL	Manipulated value: low limit	–100.0 ... LMN_HLM	0.0
DB_NBR	BLOCK_DB	Data block number (DB with the time slices of the ramp soak)		DB1
SPFC_NBR	BLOCK_FC	Setpoint FC number (self-defined FC in the setpoint branch)		FC0
PVFC_NBR	BLOCK_FC	Process variable FC number (self-defined FC in the process variable branch)		FC0
LMNFCNBR	BLOCK_FC	Manipulated value FC number (self-defined FC in the manipulated value branch)		FC0

Table 9-2 Output Parameters of PID_CP (continuous controller)

Parameter	Data Type	Explanation	Default
LMN	REAL	Manipulated value (manipulated value in floating-point format)	0.0
LMN_PER	INT	Manipulated value for I/Os (LMN in peripheral format)	W#16#0000
SP	REAL	Setpoint (effective setpoint)	0.0
PV	REAL	Process variable (output of the effective process variable in cascade control)	0.0
QCAS	BOOL	Signal for cascade control (connected to CAS_ON of the primary controller)	FALSE

Table 9-2 Output Parameters of PID_CP (continuous controller)

Parameter	Data Type	Explanation	Default
QC_ACT	BOOL	Display whether the control part is processed at the next block call (only relevant if SELECT has the value 0 or 1)	TRUE
QPOS_P	BOOL	Pulse generator Positive pulse on	FALSE
QNEG_P	BOOL	Pulse generator Negative pulse on	FALSE

Table 9-3 In/Out parameter PID_CP (continuous controller)

Parameter	Data Type	Explanation	Default
MAN	REAL	Manual manipulated value (for setting the manipulated value with operator interface functions)	0.0

Table 9-4 Static block data of PID_CP (inputs)

Parameter	Data Type	Explanation	Permitted range of values	Default
PVH_ALM	REAL	Process variable: high alarm limit	PVH_WRN...100.0	100.0
PVH_WRN	REAL	Process variable: high warning limit	PVL_WRN... PVH_ALM	90.0
PVL_WRN	REAL	Process variable: low warning limit	PVL_ALM... PVH_WRN	–90.0
PVL_ALM	REAL	Process variable: low alarm limit	–100.0...PVL_WRN	–100.0
SPGEN_ON	BOOL	Setpoint generator on (to adjust the setpoint using up/down switches)		FALSE
SPUP	BOOL	Setpoint up		FALSE
SPDN	BOOL	Setpoint down		FALSE
RMPSK_ON	BOOL	Ramp soak on (setpoint follows preset curve)		FALSE
SPEXT_ON	BOOL	External setpoint on (to connect to other controller blocks)		FALSE
MANGN_ON	BOOL	Manual generator on (LMN set by generator)		FALSE
MANUP	BOOL	Manual manipulated value up		FALSE
MANDN	BOOL	Manual manipulated value down		FALSE
DFRMP_ON	BOOL	Set ramp soak output to default (SP_INT is set at the output)		FALSE
CYC_ON	BOOL	Repetition on (ramp soak automatically repeated)		FALSE

Table 9-4 Static block data of PID_CP (inputs), Fortsetzung

Parameter	Data Type	Explanation	Permitted range of values	Default
RMP_HOLD	BOOL	Hold ramp soak (setpoint value) (the output of the ramp soak is frozen)		FALSE
CONT_ON	BOOL	Continue ramp soak (the ramp soak is continued at the next time slice)		FALSE
TUPDT_ON	BOOL	Total time update on (the total time of the ramp soak is recalculated)		FALSE
SPFC_ON	BOOL	Call the setpoint FC		FALSE
SPROC_ON	BOOL	Rate of change limits on (Limitationof the SP rate of change)		FALSE
PVPER_ON	BOOL	Process variable from I/Os on (connection to I/O modules)		FALSE
LAG1STON	BOOL	Activate time lag 1st order		FALSE
SQRT_ON	BOOL	Square root function on		FALSE
PVFC_ON	BOOL	Call process variable FC		FALSE
DEADB_ON	BOOL	Dead band on (small disturbances and noise are filtered)		FALSE
P_SEL	BOOL	P action on		TRUE
PFDB_SEL	BOOL	P action in feedback path		FALSE
INT_HPOS	BOOL	Freezing of the integral component in the positive direction		FALSE
INT_HNEG	BOOL	Freezing of the integral component in the negative direction		FALSE
I_ITL_ON	BOOL	Initialize I action		FALSE
DFDB_SEL	BOOL	D action in feedback path		FALSE
DISV_SEL	BOOL	Disturbance variable on		FALSE
LMNFC_ON	BOOL	Call manipulated value FC		FALSE
LMNRC_ON	BOOL	manipulated value rate of change limits on (LMN rate of change limited)		FALSE
SMOO_CHG	BOOL	Smooth changeover from manual to automatic		TRUE
PULSE_ON	BOOL	Pulse generator on		FALSE
STEP3_ON	BOOL	Pulse generator Three-step control on		TRUE
ST2BI_ON	BOOL	Pulse generator Two-step control for binary manipulated variable range on (for unipolar range STEP3_ON = FALSE must be set)		FALSE

Table 9-4 Static block data of PID_CP (inputs), Fortsetzung

Parameter	Data Type	Explanation	Permitted range of values	Default
TM_SNBR	INT	No. of time slice to continue	≥ 0 (no dimension)	0
TM_CONT	TIME	Time to continue (time after time slice TM_SNBR at which the ramp soak is resumed)	Entire range of values(no dimension)	T#0s
FAC	REAL	Factor (ratio or blending factor)	Entire range of values (no dimension)	1.0
NM_SPEHR	REAL	Setpoint normalization Operating range input top		100.0
NM_SPELR	REAL	Setpoint normalization Operating range input bottom		–100.0
SPFC_OUT	REAL	Setpoint FC output (connected to the output of the FC in the setpoint branch)	–100.0 ... 100.0	0.0
SPURLM_P	REAL	Setpoint up rate limit in the pos. range	≥ 0 [physical dimension/s]	10.0
SPDRLM_P	REAL	Setpoint down rate limit in the pos. range	≥ 0 [physical dimension/s]	10.0
SPURLM_N	REAL	Setpoint up rate limit in the neg. range	≥ 0 [physical dimension/s]	10.0
SPDRLM_N	REAL	Setpoint down rate limit in the neg. range	≥ 0 [physical dimension/s]	10.0
NM_PIHR	REAL	Process variable normalization Measuring range input top		100.0
NM_PILR	REAL	Process variable normalization Measuring range input bottom		–100.0
NM_PVHR	REAL	Process variable normalization Measuring range output top		100.0
NM_PVLR	REAL	Process variable normalization Measuring range output bottom		–100.0
PV_TMLAG	TIME	Process variable time lag (time lag of the PT1 element in the PV branch)	Entire range of values	T#5s
SQRT_HR	REAL	Square root: Operating range output top		100.0
SQRT_LR	REAL	Square root: Operating range output bottom		0.0
PVFC_OUT	REAL	Process variable FC output (connected to the output of the FC in the process variable branch)	–100.0 ... 100.0	0.0
PVURLM_P	REAL	Process variable up rate limit in the pos. range	≥ 0 [/s]	10.0

Table 9-4 Static block data of PID_CP (inputs), Fortsetzung

Parameter	Data Type	Explanation	Permitted range of values	Default
PVDRLM_P	REAL	Process variable-down rate limit in the pos. range	≥ 0 [/s]	10.0
PVURLM_N	REAL	Process variable up rate limit in the neg. range	≥ 0 [/s]	10.0
PVDRLM_N	REAL	Process variable down rate limit in the neg. range	≥ 0 [%/s]	10.0
PV_HYS	REAL	Process variable hysteresis (avoids flickering of the indicator)	≥ 0	1.0
DEADB_W	REAL	Dead band width (= range zero to dead band upper limit) (determines size of dead band)	0.0 to 100.0	1.0
ERP_ALM	REAL	Error signal: positive alarm limit	0 to 200.0	100.0
ERP_WRN	REAL	Error signal: positive warning limit	0 ... 200.0	90.0
ERN_WRN	REAL	Error signal: Neg. warning limit	–200.0 ... 0	–90.0
ERN_ALM	REAL	Error signal: negative alarm limit	–200.0 ... 0	–100.0
ER_HYS	REAL	Error signal hysteresis (avoids flickering of the indicator)	≥ 0 [%]	1.0
I_ITLVAL	REAL	Initialization value for I action	–100.0 to 100.0 [%]	0.0
LMNFCOUT	REAL	Manipulated value FC output (connected to the output of the FC in the manipulated value branch)	–100.0 to 100.0 [%]	0.0
LMN_URLM	REAL	Manipulated value up rate limit	≥ 0 [%/s]	10.0
LMN_DRLM	REAL	Manipulated value down rate limit	≥ 0 [%/s]	10.0
LMN_FAC	REAL	Manipulated value factor (factor for adapting the manipulated value range)	Entire range of values (no dimension)	1.0
LMN_OFF	REAL	Manipulated value offset (zero point of the manipulated value normalization)	Entire range of values (no dimension)	0.0
PER_TM_P	TIME	Pulse generator Period time of the positive pulse		T#1s
PER_TM_N	TIME	Pulse generator Period time of the negative pulse		T#1s
P_B_TM_P	TIME	Pulse generator: Minimum pulse or minimum break time of the positive pulse		T#50ms
P_B_TM_N	TIME	Pulse generator:Minimum pulse or minimum break time of the negative pulse		T#50ms
RATIOFAC	REAL	Pulse generator Ratio factor (ratio of the positive pulse duration and negative pulse duration)	0.1 ... 10.0 (dimensionslos)	1.0
PHASE	INT	Phase of PID Self Tuner		0

Table 9-5 Static local data of PID_CP (outputs)

Parameter	Data Type	Explanation	Default
QPVH_ALM	BOOL	Process variable: high alarm limit triggered	FALSE
QPVH_WRN	BOOL	Process variable: high warning limit triggered	FALSE
QPVL_WRN	BOOL	Process variable: low warning limit triggered	FALSE
QPVL_ALM	BOOL	Process variable: low alarm limit triggered	FALSE
QR_S_ACT	BOOL	Time table for ramp soak being processed	FALSE
QSP_HLM	BOOL	Setpoint: high limit triggered	FALSE
QSP_LLM	BOOL	Setpoint: low limit triggered	FALSE
QPVURLMP	BOOL	Process variable: up rate limit in the positive range triggered	FALSE
QPVDRLMP	BOOL	Process variable: down rate limit in the positive range triggered	FALSE
QPVURLMN	BOOL	Process variable: up rate limit in the negative range triggered	FALSE
QPVDRLMN	BOOL	Process variable: down rate limit in the negative range triggered	FALSE
QERP_ALM	BOOL	Error signal: positive alarm limit triggered	FALSE
QERP_WRN	BOOL	Error signal: positive warning limit triggered	FALSE
QERN_WRN	BOOL	Error signal; negative warning limit triggered	FALSE
QERN_ALM	BOOL	Error signal: negative alarm limit triggered	FALSE
QLMN_HLM	BOOL	Manipulated value: high limit triggered	FALSE
QLMN_LLM	BOOL	Manipulated value: low limit triggered	FALSE
NBR_ATMS	INT	Number of the time slice the ramp soak is moving to	0
RS_TM	TIME	Time remaining until the next time slice	T#0s
T_TM	TIME	Total time of the ramp soak	T#0s
RT_TM	TIME	Total time remaining to end of ramp soak	T#0s
ER	REAL	Error signal	0.0
LMN_P	REAL	P action	0.0
LMN_I	REAL	I action	0.0
LMN_D	REAL	D action	0.0
SPFC_IN	REAL	Setpoint FC input (connected to the input of the user-defined FC)	0.0
PVFC_IN	REAL	Process variable FC input (connected to the input of the user-defined FC)	0.0
LMNFC_IN	REAL	Manipulated value FC input (connected to the input of the user-defined FC)	0.0

Table 9-6 Static local data used by the configuration tool PID_CP

Parameter	Data Type	Explanation	Default
SP_OP_ON	BOOL	Setpoint generator on (the value of SP_OP is used as the setpoint)	FALSE
PV_OP_ON	BOOL	Process variable operation on (the value of PV_OP is used as the setpoint)	FALSE
LMNOP_ON	BOOL	Manipulated value operation on (the value of LMN_OP is used as the setpoint)	FALSE
SP_OP	REAL	Setpoint generator of configuration tool	0.0
PV_OP	REAL	Process variable operation of configuration tool	0.0
LMN_OP	REAL	Manipulated value operation of configuration tool	0.0
MP1	REAL	Measuring point 1: Internal setpoint	0.0
MP2	REAL	Measuring point 2: External setpoint	0.0
MP3	REAL	Measuring point 3: Unlimited setpoint	0.0
MP4	REAL	Measuring point 4: Process variable from I/O module	0.0
MP5	REAL	Measuring point 5: Process variable after 1st order time lag	0.0
MP6	REAL	Measuring point 6: Effective process variable (PV)	0.0
MP7	REAL	Measuring point 7: Manipulated value from PID algorithm	0.0
MP8	REAL	Measuring point 8: Manual manipulated value	0.0
MP9	REAL	Measuring point 9: Unlimited manipulated value	0.0
MP10	REAL	Measuring point 10: Limited manipulated value	0.0

The static local data used by the configuration tool are at the start of the range of values of the static local data.

Note

All the other static local data may not be influenced.

9.2 Parameters of the PID_ES Function Block

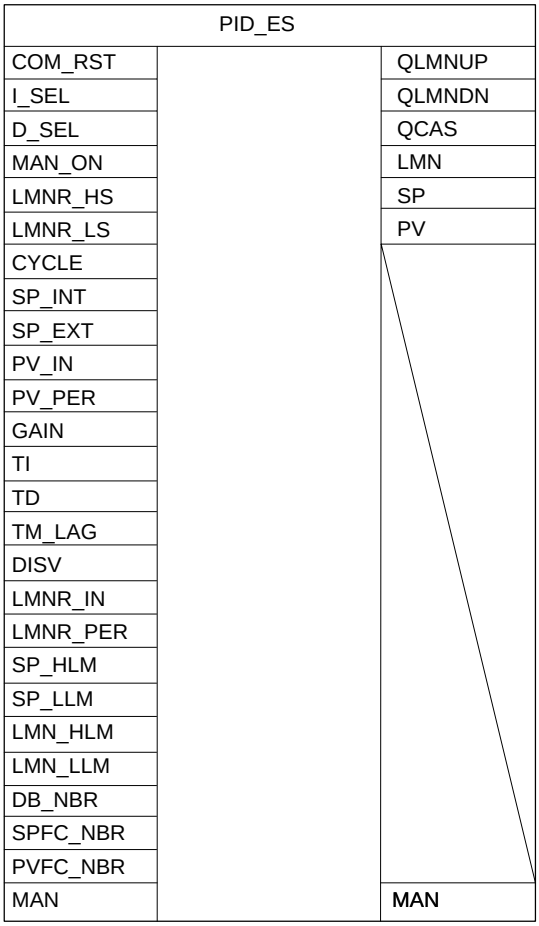


Table 9-7 Input Parameters of PID_ES (step controller)

Parameter	Data Type	Explanation	Permitted range of values	Default
COM_RST	BOOL	Complete restart (initialization routine of the FB is processed)		FALSE
I_SEL	BOOL	I action on		TRUE
D_SEL	BOOL	D action on		FALSE
MAN_ON	BOOL	Manual mode on (loop opened, LMN set manually)		TRUE
LMNR_HS	BOOL	Upper limit stop signal of the position feedback signal		FALSE
LMNR_LS	BOOL	Lower limit stop signal of the position feedback signal		FALSE

Table 9-7 Input Parameters of PID_ES (step controller)

Parameter	Data Type	Explanation	Permitted range of values	Default
CYCLE	TIME	Sampling time (time between block calls = constant) Be sure to configure this parameter with the watchdog-interrupt cycle of the OB in which the “PID_CP” FB runs! Otherwise the time-dependent functions do not function correctly. (Exception: You use a pulse scaling, for example via the controller call distribution.)	$\geq$ 20 ms (S7-300)	T#1s
SP_INT	REAL	Internal setpoint (for setting the setpoint with operator interface functions)	Technical range of values (physical value)	0.0
SP_EXT	REAL	External setpoint (SP in floating-point format)	Technical range of values (physical value)	0.0
PV_IN	REAL	Process variable input (PV in floating-point format)	Technical range of values (physical value)	0.0
PV_PER	INT	Process variable from I/Os		W#16#0000
GAIN	REAL	Proportional gain (= controller gain)	Entire range of values (no dimension)	2.0
TI	TIME	Reset time	TI $\geq$ CYCLE	T#20s
TD	TIME	Derivative action time	TD $\geq$ CYCLE	T#10s
TM_LAG	TIME	Time lag of the D component	TM_LAG $\geq$ CYCLE/2	T#2s
DISV	REAL	Disturbance variable	–100.0 to 100.0 [%]	0.0
LMNR_IN	REAL	Position feedback signal (LMNR in floating-point format)	0.0 to 100.0 [%]	0.0
LMNR_PER	WORD	Position feedback signal from I/Os (LMNR in peripheral format)		W#16#0000
SP_HLM	REAL	Setpoint high limit	Technical range of values (physical value)	100.0
SP_LLM	REAL	Setpoint low limit	Technical range of values (physical value)	0.0
LMN_HLM	REAL	Manipulated value: high limit	LMN_LLM .. 100.0[%]	100.0
LMN_LLM	REAL	Manipulated value: low limit	0.0 to LMN_HLM [%]	0.0
DB_NBR	BLOCK_DB	Data block number (DB with the time slices of the ramp soak)		DB1

Table 9-7 Input Parameters of PID_ES (step controller)

Parameter	Data Type	Explanation	Permitted range of values	Default
SPFC_NBR	BLOCK_FC	Setpoint FC number (self-defined FC in the setpoint branch)		FC0
PVFC_NBR	BLOCK_FC	Process variable FC number (self-defined FC in the process variable branch)		FC0

Table 9-8 Output Parameters of PID_ES (step controller)

Parameter	Data Type	Explanation	Default
QLMNUP	BOOL	Manipulated value signal up	FALSE
QLMNDN	BOOL	Manipulated value signal down	FALSE
QCAS	BOOL	Signal for cascade control (connected to CAS_ON of the primary controller)	FALSE
LMN	REAL	Manipulated value signal (after control algorithm)	0.0
SP	REAL	Setpoint (effective setpoint)	0.0
PV	REAL	Process variable (output of the effective process variable in cascade control)	0.0

Table 9-9 In/Out Parameters of PID_ES (step controller)

Parameter	Data Type	Explanation	Default
MAN	REAL	Manual manipulated value (for setting the manipulated value with operator interface functions)	0.0

Table 9-10 Static Local Data of PID_ES (inputs)

Parameter	Data Type	Explanation	Permitted range of values	Default
PVH_ALM	REAL	Process variable: high alarm limit	PVH_WRN...100.0	100.0
PVH_WRN	REAL	Process variable: high warning limit	PVL_WRN... PVH_ALM	90.0
PVL_WRN	REAL	Process variable: low warning limit	PVL_ALM... PVH_WRN	–90.0
PVL_ALM	REAL	Process variable: low alarm limit	–100.0...PVL_WRN	–100.0
SPGEN_ON	BOOL	Setpoint generator on (to adjust the setpoint using up/down switches)		FALSE
SPUP	BOOL	Setpoint up		FALSE
SPDN	BOOL	Setpoint down		FALSE
RMPSK_ON	BOOL	Ramp soak on (setpoint follows preset curve)		FALSE
SPEXT_ON	BOOL	External setpoint on (to connect to other controller blocks)		FALSE
MANGN_ON	BOOL	Manual generator on (LMN set by generator)		FALSE
MANUP	BOOL	Manual manipulated value up		FALSE
MANDN	BOOL	Manual manipulated value down		FALSE
LMNS_ON	BOOL	Manual mode actuating signals on		FALSE
LMNUP	BOOL	Manipulated value signal up (the output signal QLMNUP is set manually)		FALSE
LMNDN	BOOL	manipulated value signal down (the output signal QLMNDN is set manually)		FALSE
DFRMP_ON	BOOL	Set ramp soak output to default (SP_INT is set at the output)		FALSE
CYC_ON	BOOL	Repetition on (ramp soak automatically repeated)		FALSE
RMP_HOLD	BOOL	Hold ramp soak (setpoint value) (the output of the ramp soak is frozen)		FALSE
CONT_ON	BOOL	Continue ramp soak (the ramp soak is continued at the next time slice)		FALSE
TUPDT_ON	BOOL	Total time update on (the total time of the ramp soak is recalculated)		FALSE
SPFC_ON	BOOL	Call the setpoint FC		FALSE
SPROC_ON	BOOL	Rate of change limits on (Limitation of the SP rate of change)		FALSE
PVPER_ON	BOOL	Process variable from I/Os on (connection to I/O modules)		FALSE
LAG1STON	BOOL	Activate time lag 1st order		FALSE
SQRT_ON	BOOL	Square root function on		FALSE

Table 9-10 Static Local Data of PID_ES (inputs), Fortsetzung

Parameter	Data Type	Explanation	Permitted range of values	Default
PVFC_ON	BOOL	Call process variable FC		FALSE
DEADB_ON	BOOL	Dead band on (small disturbances and noise are filtered)		FALSE
P_SEL	BOOL	P action on		TRUE
PFDB_SEL	BOOL	P action in feedback path		FALSE
INT_HPOS	BOOL	Freezing of the integral component in the positive direction		FALSE
INT_HNEG	BOOL	Freezing of the integral component in the negative direction		FALSE
I_ITL_ON	BOOL	Initialize I action		FALSE
DFDB_SEL	BOOL	D action in feedback path		FALSE
DISV_SEL	BOOL	Disturbance variable on		FALSE
LMNR_ON	BOOL	position feedback signal on (Modes: Step controller with/witout position feedback) Do not switch over in closed-loop control!		FALSE
LMNRP_ON	BOOL	Position feedback signal from I/Os on		FALSE
TM_SNBR	INT	Number of the next time slice for continuing the curve	≥ 0 (no dimension)	0
TM_CONT	TIME	Time lag until contimuatio of the curve (Time lag before the time scheduler contuinues to run after the curve has been interrupted at time slice TM_SNBR)	Entire range of values (no dimension)	T#0s
FAC	REAL	Factor (ratio or blending factor)	Entire range of values (no dimension)	1.0
NM_SPEHR	REAL	Setpoint normalization: Input top		100.0
NM_SPELR	REAL	Setpoint normalization: Input bottom		–100.0
SPFC_OUT	REAL	Setpoint FC output (connected to the output of the FC in the setpoint branch)	–100.0 ... 100.0	0.0
SPURLM_P	REAL	Setpoint up rate limit in the pos. range	≥ 0 [/s]	10.0
SPDRLM_P	REAL	Setpoint down rate limit in the pos. range	≥ 0 [/s]	10.0
SPURLM_N	REAL	Setpoint up rate limit in the neg. range	≥ 0 [/s]	10.0
SPDRLM_N	REAL	Setpoint down rate limit in the neg. range	≥ 0 [/s]	10.0
NM_PIHR	REAL	Process variable normalization Input top		100.0
NM_PILR	REAL	Process variable normalization Input bottom		–100.0
NM_PVHR	REAL	Process variable normalization Output top		100.0
NM_PVLR	REAL	Process variable normalization: Output bottom		–100.0

Table 9-10 Static Local Data of PID_ES (inputs), Fortsetzung

Parameter	Data Type	Explanation	Permitted range of values	Default
PV_TMLAG	TIME	Process variable time lag (time lag of the PT1 element in the PV branch)	Entire range of values	T#5s
SQRT_HR	REAL	Square root: Measuring range output top		100.0
SQRT_LR	REAL	Square root: Measuring range output bottom		0.0
PVFC_OUT	REAL	Process variable FC output (connected to the output of the FC in the process variable branch)	–100.0 ... 100.0	0.0
PVURLM_P	REAL	Process variable up rate limit in the pos. range	≥ 0 [/s]	10.0
PVDRLM_P	REAL	Process variable-down rate limit in the pos. range	≥ 0 [/s]	10.0
PVURLM_N	REAL	Process variable up rate limit in the neg. range	≥ 0 [/s]	10.0
PVDRLM_N	REAL	Process variable down rate limit in the neg. range	≥ 0 [/s]	10.0
PV_HYS	REAL	Process variable hysteresis (avoids flickering of the indicator)	≥ 0	1.0
DEADB_W	REAL	Dead band width (determines size of dead band)	0.0 to 100.0	1.0
ERP_ALM	REAL	Error signal: positive alarm limit	0 to 200.0	100.0
ERP_WRN	REAL	Error signal: positive warning limit	0 to 200.0	90.0
ERN_WRN	REAL	Error signal: Neg. warning limit	–200.0 ... 0	–90.0
ERN_ALM	REAL	Error signal: negative alarm limit	–200.0 ... 0	–100.0
ER_HYS	REAL	Error signal hysteresis (avoids flickering of the indicator)	≥ 0	1.0
I_ITLVAL	REAL	Initialization value for I action	–100.0 to 100.0 [%]	0.0
LMNR_FAC	REAL	Position feedback signal factor (factor for adapting the position feedback range)	Entire range of values (no dimension)	1.0
LMNR_OFF	REAL	Position feedback signal offset (zero point of the position feedback normalization)	–100.0 to 100.0 [%]	0.0
PULSE_TM	TIME	Minimum pulse time	= n ∗ CYCLE /n=0,1,2...	T#3s
BREAK_TM	TIME	Minimum break time	= n ∗ CYCLE /n=0,1,2...	T#3s
MTR_TM	TIME	Motor actuating time	$\geq$ CYCLE	T#30s
PHASE	INT	Phase of PID Self Tuner		0

Table 9-11 Static local data of PID_ES (outputs)

Parameter	Data Type	Explanation	Default
QPVH_ALM	BOOL	Process variable: high alarm limit triggered	FALSE
QPVH_WRN	BOOL	Process variable: high warning limit triggered	FALSE
QPVL_WRN	BOOL	Process variable: low warning limit triggered	FALSE
QPVL_ALM	BOOL	Process variable: low alarm limit triggered	FALSE
QR_S_ACT	BOOL	Time table for ramp soak being processed	FALSE
QSP_HLM	BOOL	Setpoint: high limit triggered	FALSE
QSP_LLM	BOOL	Setpoint: low limit triggered	FALSE
QPVURLMP	BOOL	Process variable: up rate limit in the positive range triggered	FALSE
QPVDRLMP	BOOL	Process variable: down rate limit in the positive range triggered	FALSE
QPVURLMN	BOOL	Process variable: up rate limit in the negative range triggered	FALSE
QPVDRLMN	BOOL	Process variable: down rate limit in the negative range triggered	FALSE
QERP_ALM	BOOL	Error signal: positive alarm limit triggered	FALSE
QERP_WRN	BOOL	Error signal: positive warning limit triggered	FALSE
QERN_WRN	BOOL	Error signal; negative warning limit triggered	FALSE
QERN_ALM	BOOL	Error signal: negative alarm limit triggered	FALSE
QLMN_HLM	BOOL	Manipulated value: high limit triggered	FALSE
QLMN_LLM	BOOL	Manipulated value: low limit triggered	FALSE
NBR_ATMS	INT	Number of the time slice the ramp soak is moving to	0
RS_TM	TIME	Time remaining until the next time slice	T#0s
T_TM	TIME	Total elapsed time of the ramp soak	T#0s
RT_TM	TIME	Total time remaining to end of ramp soak	T#0s
ER	REAL	Error signal	0.0
LMN_P	REAL	P action	0.0
LMN_I	REAL	I action	0.0
LMN_D	REAL	D action	0.0
SPFC_IN	REAL	Setpoint FC input (connected to the input of the user-defined FC)	0.0
PVFC_IN	REAL	Process variable FC input (connected to the input of the user-defined FC)	0.0

Table 9-12 Static Local Data used by the Configuration Tool (step controller PID_ES)

Parameter	Data Type	Explanation	Default
SP_OP_ON	BOOL	Setpoint generator on (the value of SP_OP is used as the setpoint)	FALSE
PV_OP_ON	BOOL	Process variable operation on (the value of PV_OP is used as the setpoint)	FALSE
LMNOP_ON	BOOL	Manipulated value operation on (the value of LMN_OP is used as the setpoint)	FALSE
LMNSOPON	BOOL	Manipulated value signal operation on (LMNUP_OP and LMNDN_OP are used as actuating signals)	FALSE
LMNUP_OP	BOOL	Manipulated value signal up	FALSE
LMNDN_OP	BOOL	manipulated value signal down	FALSE
LMNRS_ON	BOOL	Simulation of the position feedback signal on	FALSE
SP_OP	REAL	Setpoint generator of configuration tool	0.0
PV_OP	REAL	Process variable operation of configuration tool	0.0
LMN_OP	REAL	Manipulated value operation of configuration tool	0.0
LMNRSVAL	REAL	Start value of simulated position feedback signal	0.0
LMNR_SIM	REAL	Current value of simulated position feedback signal	0.0
MP1	REAL	Measuring point 1: Internal setpoint	0.0
MP2	REAL	Measuring point 2: External setpoint	0.0
MP3	REAL	Measuring point 3: Unlimited setpoint	0.0
MP4	REAL	Measuring point 4: Process variable from I/O module	0.0
MP5	REAL	Measuring point 5: Process variable after 1st order time lag	0.0
MP6	REAL	Measuring point 6: Effective process variable (PV)	0.0
MP7	REAL	Measuring point 7: Manipulated value from PID algorithm	0.0
MP8	REAL	Measuring point 8: Manual manipulated value	0.0
MP9	REAL	Measuring point 9: Unlimited manipulated value	0.0
MP10	REAL	Measuring point 10: Position feedback signal I/Os	0.0
MP11	REAL	Measuring point 11: Feedback value (LMNR_ON = FALSE) Position feedback signal (LMNR_ON = TRUE)	0.0
MP12	REAL	Measuring point 12: Three-step element input	0.0

The static local data used by the configuration tool are at the start of the range of values of the static local data.

Note

All the other static local data may not be influenced.

Table 9-13 RMP_SOAK Function (PID_CP and PID_ES): Shared Data Block (DB_NBR), with Default of Start Point and Four Time Slices

Parameter	Data Type	Comment	Permitted range of values	Default
NBR_PTS	INT	Number of coordinates	0 to 255	4
PI[0].OUTV	REAL	Output value [0]: start point	Entire range of values	0.0
PI[0].TMV	TIME	Time value [0]: start point	Entire range of values	T#1 s
PI[1].OUTV	REAL	Output value [1]: coordinate 1	Entire range of values	0.0
PI[1].TMV	TIME	Time value [1]: coordinate 1	Entire range of values	T#1 s
PI[2].OUTV	REAL	Output value [2]: coordinate 2	Entire range of values	0.0
PI[2].TMV	TIME	Time value [2]: coordinate 2	Entire range of values	T#1 s
PI[3].OUTV	REAL	Output value [3]: coordinate 3	Entire range of values	0.0
PI[3].TMV	TIME	Time value [3]: coordinate 3	Entire range of values	T#1 s
PI[4].OUTV	REAL	Output value [4]: coordinate 4	Entire range of values	0.0
PI[4].TMV	TIME	Time value [4]: coordinate 4	Entire range of values	T#0 s

9.3 Parameter of the LP_SCHED Function

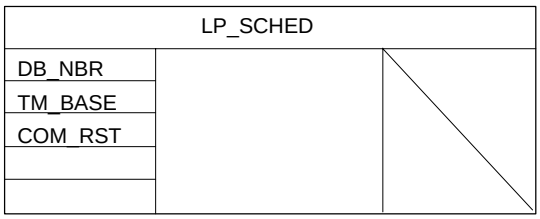


Figure 9-1 LP_SCHED Function

Table 9-14 Input Parameters of LP_SCHED

Parameter	Data Type	Explanation	Permitted range of values	Default
TM_BASE	TIME	Time base (time base of the cyclic interrupt class in which LP-SCHED is called)	≥20 ms (S7-300) ≥ 5 ms (S7-400)	100 ms
COM_RST	BOOL	Complete restart (complete restart routine of LP_SCHED is processed)		FALSE
DB_NBR	BLOCK_DB	Data block number (DB with the call data of the control loops)		DB1

Table 9-15 Global Data Area “DB_NBR”

Parameter	Data Type	Explanation	Permitted range of values	Default
GLP_NBR	INT	Highest control loop number	1 to 256	2
ALP_NBR	INT	Current control loop number	1 to 256	0
LOOP_DAT[1] MAN_CYC	TIME	Control loop data [1]: manual sampling time	≥20 ms (S7-300) ≥ 5 ms (S7-400)	T#1s
LOOP_DAT[1] MAN_DIS	BOOL	Control loop data [1]: disable manual controller call		FALSE
LOOP_DAT[1] MAN_CRST	BOOL	Control loop data [1]: set manual complete restart (user can reset the particular control loop)		FALSE
LOOP_DAT[1] ENABLE	BOOL	Control loop data [1]: controller enable (User must program the conditional call for the control loop)		FALSE
LOOP_DAT[1] COM_RST	BOOL	Control loop data [1]: complete restart (this parameter is connected to COM_RST of the control loop)		FALSE
LOOP_DAT[1] ILP_COU	INT	Control loop data [1]: internal control loop counter (internal count variable)		0
LOOP_DAT[1] CYCLE	TIME	Control loop data [1]: sampling time	≥20 ms (S7-300) ≥ 5 ms (S7-400)	T#1s
...		...	...	...

Configuration Software for Standard PID Control

10

Prerequisites

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

Supply Form

The software is supplied on a CD.

Installation

Proceed as follows to install the software:

1. Insert the CD with the Standard PID Control Tool into the CD drive.
2. Start the dialog box for installing the software under WINDOWS by double-clicking on the ”Software” icon in the ”Control panel”.
3. In the dialog box select the drive and the file Setup.exe and start the installation process.
   The configuration tool is then installed on your programming device/personal computer.
4. Follow the instructions displayed by the installation toool step-by-step.

Reading Out the Readme File

The Readme file may contain important last-minute information on the software supplied.This file is positioned in the start menu of WINDOWS under SIMATIC\STEP7\Notes.

Purpose

The configuration tool supports you when installing and assigning parameters to the standard controller block so that you can spend more time on the actual control problems.

Using the configuration tool you can assign parameters to the standard controller blocks

- PID_CP (Controller with continuous output)
- PID_ES (controller with output for step control)

and optimize the parameters to match the characteristics of the process.

The Functions of the Configuration Tool

The overall performance of the configuration tool can be divided into individual functions. Each of these functions runs in its own window. A function can also be called more than once, in other words, you can, for example, display the loop windows of several controllers simultaneously.

Monitoring the Controller

Using the **Curve Recorder**function, you can record and display the values of a selected variable of the control loop over a defined period of time. Up to four variables can be displayed simultaneously.

With the **Loop Monitor**function, you can display the relevant control loop variables (setpoint, manipulated variable and process variable) of a selected controller. Values exceeding the limit values of the process variable are also displayed.

Process Identification

Using the **Process Identification** function, you can determine the optimum controller setting for a specific control loop. The characteristic parameters of the control loop are calculated experimentally. The ideal controller parameters are then calculated so that you can use them as required.

During this procedure, it is irrelevant whether the values recorded while the process is settling originate from a controller acting on a simulated process or acting on a real process on-line.

Modifying a Controller

Using the Loop Monitor function, you can change the control loop variables of the currently displayed controller or enter new values.

Integrated Help

The configuration tool has an integrated help which support you. You have the following possibilities for calling up the integrated help:

- Use the menu command **Help > Help topics**
- Press F1
- Click on the help icon/button in the individual masks
- Use the menu command **Help > Context help** and then select the function block or parameter for which you require help
- Use the ”Help” button (arrow with question mark) in the toolbar and then select the function block or parameter for which you require help

If you point the mouse at an input box or at a connection line in the main window, the parameter name and the address in the data block are displayed. In you have opened the block on-line, the on-line value of the variables is also displayed.

Literature List

A

/70/ Manual: *Programmable Controller S7-300,*
Hardware and Installation

/71/ Reference Manual: *Programmable Controllers S7-300 and M7-300*
Module Specifications

/100/ Manual: *Programmable Controllers S7-400 and M7-400,*
Hardware and Installation

/101/ Reference Manual: *Programmable Controllers S7-400 and M7-40*
Module Specifications

/231/ Manual: *Configuring Hardware and Communication Connections STEP 7 V5.0*

/232/ Reference Manual: *Statement List (STL) for S7-300 and S7-400,*
Programming

/234/ Manual: *Programming with STEP 7 V5.0*

/352/ J. Gißler, M. Schmid: From process to control. Analysis, Design, Implementation in the Practise. Siemens AG. ISBN 3-8009-1551-0.

Glossary

Adjustment Profile

In blending and cascade controls with several secondary loops, the setpoint of the secondary loops can be influenced by a specific factor [FAC]. This determines the degree of intervention at this point in the system resulting in the overall adjustment profile.

Alignment Factor

In a ratio controller, the alignment factor FAC is used to align the setpoints of the control loops with each other so that the set ratio corresponds to the actual ratio of the two process variables (→ratio controller)

In a blending controller the alignmennt factor FAC is used to set the desired quantities of the individual components. The sum of the blending factors FAC must be 1 (→ blending control).

Analog Input/Output

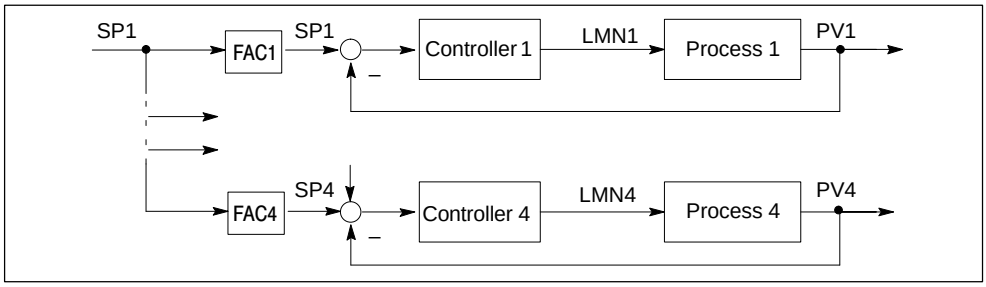
The analog input/output (CRP_IN and CRP_OUT) is an algorithm (function) for converting an input value in the peripheral (I/O data) format to a floating point and normalizing the value to a percentage and in the other direction, converting an internal percentage to an output value in the I/O (peripheral) format.

Automatic Mode

The controller operates and calculates the manipulated variable with the aim of minimizing the error signal (closed loop).

Blending Control

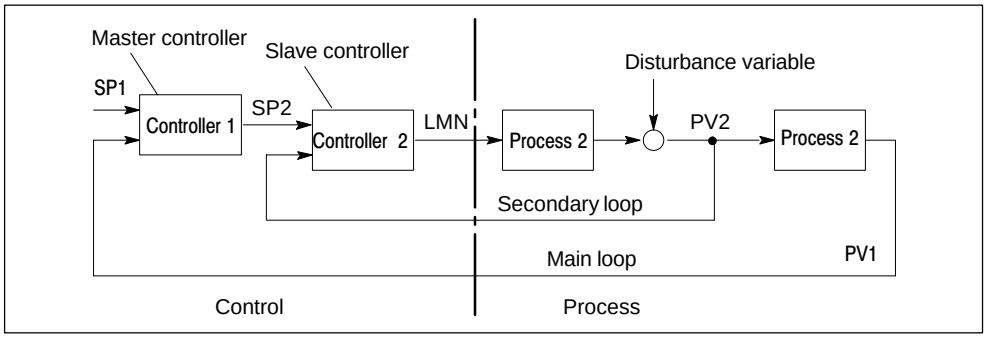
Blending control involves a controller structure in which the setpoint for the total amount SP is converted to percentages of the individual components. The total of the blending factors FAC must be 1.



Cascade Control

Cascade control involves a series of interconnected controllers, in which the master controller adjusts the setpoint for the secondary (slave) controllers according to the instantaneous error signal of the main process variable.

A cascade control system can be improved by including additional process variables. A secondary process variable PV2 is measured at a suitable point and controlled to the reference setpoint (output of the maser controller SP2). The master controller controls the process variable PV1 to the fixed setpoint SP1 and sets SP2 so that the target is achieved as quickly as possible without overshoot.



Closed-Loop Controller

A closed-loop controller is a device in which the error signal is continuously detected (comparator) and a (time-dependent) function for generating the actuating signal (output variable) is generated with the aim of eliminating the error signal quickly and without overshoot.

Complete Restart

During a complete restart, a controller is set to a defined initial status. The output parameters and local static data of the controller are assigned default values during the complete restart routine.

Configuration

A software tool for creating and designing a standard controller and optimizing the controller settings using the data from a process identification procedure.

Control Device

Totality of the controllers, process control units and detectors (measuring devices) for the process variables.

Control Loop

The control loop is the connection between the process output (process variable) and the controller input and the controller output (manipulated variable) with the process input, so that the controller and process form a closed loop.

Control Settling Time

In the case of a step response on a higher-level PT process (= self-regulating process) the control settling time is the section in which the inflectional tangent cuts the parallel lines to the time axis through the starting and end times.

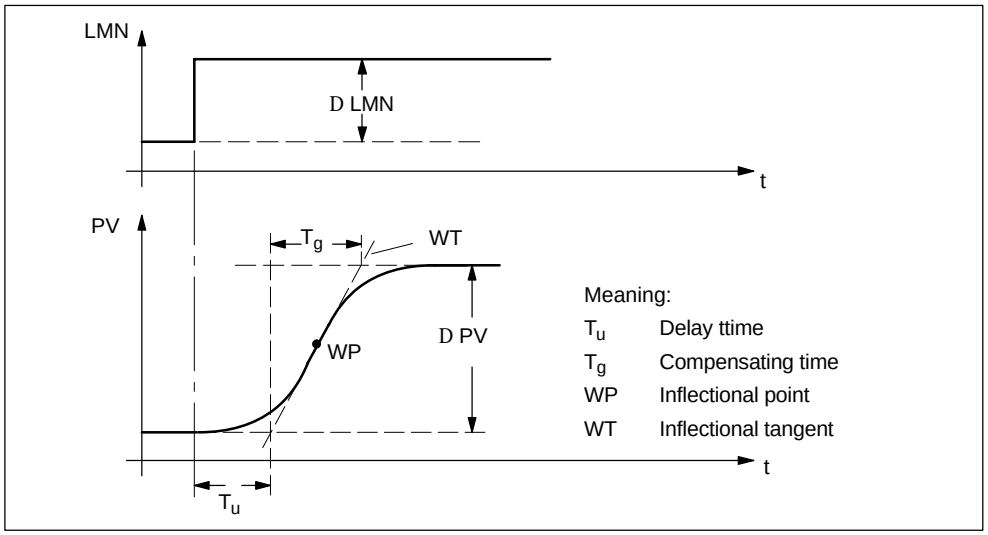


Figure 1-1 Transition function of a PT3 process

Controller Parameters

Controller parameters are characteristic values for the static and dynamic adaptation of the controller response to the given loop or process characteristics.

DDC

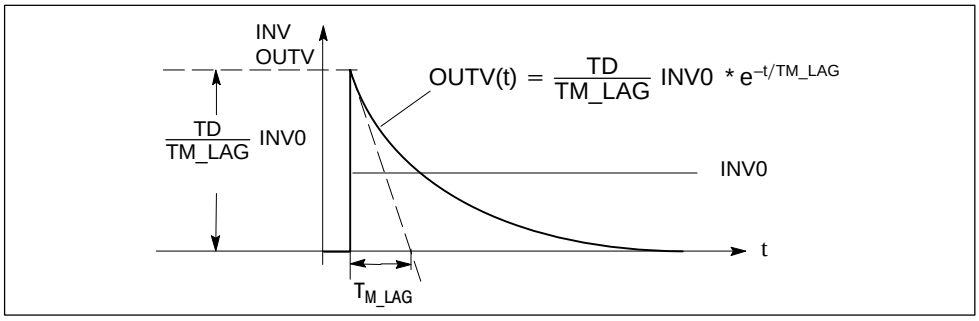
DDC is a discrete controller in which the error signal is updated at the sampling point (→ sampling time, → digital controller).

Dead Time

Dead time is the time delay in the process variable reaction to disturbances or manipulated value changes in processes involving transport. The input variable of a dead time element is displaced by the value of the dead time at the output.

Derivative Action

A method (algorithm) for differentiating an analog variable whereby the time response is determined by the derivative time TD (= reset time). The output signal of the derivative unit is proportional to the rate of change of deviation of its input signal. A first order time lag TM_LAG is provided to suppress peak derivative values or disturbance signals. The step function has the following format:



Derivative Component

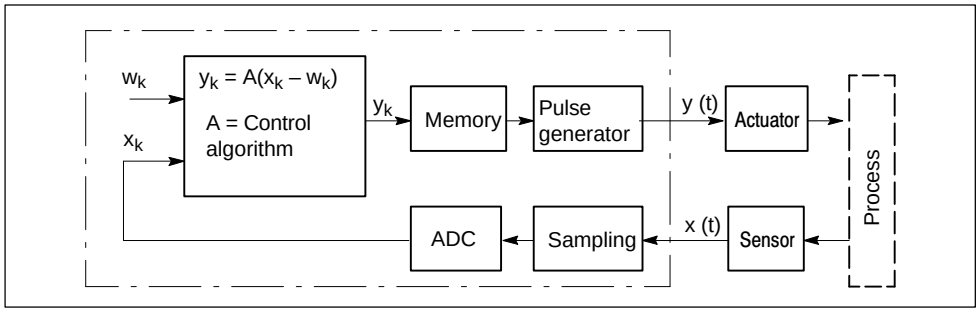
The derivative component is the differentiating component of the controller. D elements alone are unsuitable for control since they do not produce an output signal if the input signal remains at a constant value.

Derivative Time T_V

The derivative time determines the time response of the derivative component in the PD or PID controller (T_V = TD).

Digital Control

A controller that acquires a new value for the controlled variable (process variable) constant intervals (→ sampling time) and then calculates a new value for the manipulated variable depending on the value of the current error signal.



Disturbance Variable

All influences on the process variable (with the exception of the manipulated variable) are known as disturbances. Influences adding to the process output signal can be compensated by superimposing the actuating signal.

Error Signal (ER)

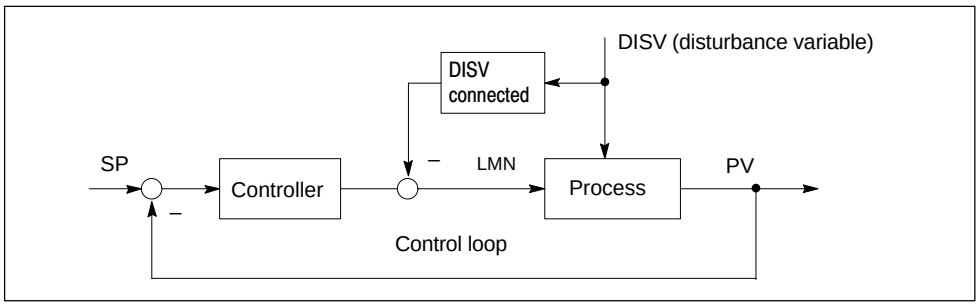
The error signal function forms the error signal ER = SP-PV. At the point at which the comparison is made, the difference between the desired value (setpoint) and the actual process value is calculated. This value is applied to the input of the control algorithm.

Error Signal Monitoring

This function monitors four selectable limits for the the value (amplitude) of the error signal. If these limits are reached or exceeded a warning (1st limit) or an alarm (2nd limit) is generated. A hysteresis can be set for the off threshold of the limit signals to prevent signal “flickering”.

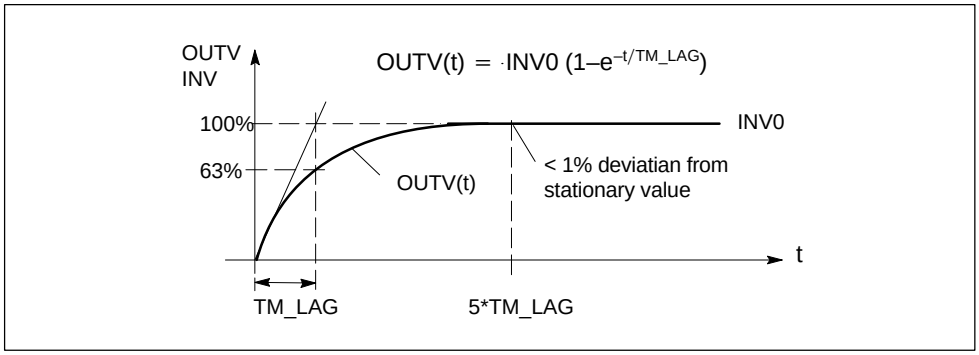
Feedforward Control

Feedforward control is a technique for reducing or eliminating the influence of a dominant (measurable) disturbance (for example ambient temperature) in the control loop. The measured disturbance variable DISV, is compensated before it affects the process. Ideally, the influence can be fully compensated so that the controller itself does not need to take corrective action itself (with the I action).



First Order Lag

A first order lag is a function for damping (applying a time lag) the changes in the analog process variable. The time lag constant TM_LAG specifies the time required by the output signal to reach 63 % of the stationary end value. The transfer ratio in the settled state is 1 : 1.



Fixed Setpoint Control

A fixed setpoint controller is a controller with a fixed setpoint that is only changed occasionally. This controller is used to compensate for disturbances occurring in the process.

Follow-Up Control

Follow-up control involves a controller in which the setpoint is constantly influenced externally (secondary controller of a multi-loop control system). The task of the secondary controller is to correct the local process variable as quickly and accurately as possible so that it matches the setpoint.

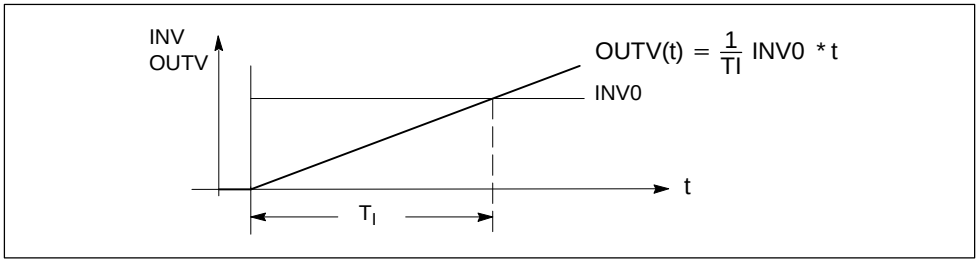
Integral Action

A procedure (algorithm) for integrating an analog value where the time response is determined by the reset time TI. The rate of change of the output signal of the integrator is proportional to the static change in the input signal. The integral action coefficient KI = 1/TI is a measure of the rate of rise of the output signal when the input signal is not zero. The step response is as follows:

Integral Component

Integral action or component of the controller.

After a step change in the process variable (or error signal) the output variable changes with a ramp function over time at a rate of change proportional to the integral-action factor KI (= 1/TI). The integral component in a closed control loop has the effect of correcting the controller output variable until the error signal becomes zero.



Interpolation

Interpolation is a method of calculating interim values based on the values known at the start and end of an interval (→ ramp soak).

Limit Alarm Monitor

An algorithm (function) for monitoring four selectable limits of an analog value. When these limits are reached or exceeded, a warning (first limit) or alarm (second limit) signal is generated. To avoid signal flickering, the off threshold of the limit signals can be selected with a hysteresis parameter.

Limiter

An algorithm (function) for restricting the range of values of constant variables to selectable upper or lower limit values.

Linear Scaling

Linear scaling is a function for converting or correcting process values.

Algorithm: Output = Input * FACTOR + OFFSET

Loop Gain

The loop gain is the product of the proportional gain (GAIN) and the gain of the process (K_S)

Loop Scheduler

The loop scheduler organizes the calls for several controllers in one cyclic interrupt priority class and the calls for all controllers during a complete restart. The loop scheduler is used when there are too many controllers for one cyclic interrupt priority class or when controllers with long sampling times are used.

Manipulated Variable

The manipulated variable is the output variable of the controller or input variable of the process. The actuating signal can take the form of an analog percentage or a pulse duration value. With integrating actuators (for example motor-driven) binary up/down or forwards/backwards signals are adequate.

Manual Value

A value injected into the interrupted loop (→ manual mode) as an absolute value or as an increment (using the up or down switch) as a percentage of the range.

Master Controller

The master controller is the primary controller in a multi-loop control system. It generates the setpoint for the secondary controller (S) (→ cascade control).

Master Control Response

The master control response is the time response of the process variable in the closed loop after a step change in the setpoint.

Manual Mode

In the manual mode, the value of the manipulated variable (LMN) is influenced manually. The current manipulated value is specified by the operator or by a STEP 7 user program as a percentage of the possible range.

If rate of change limitations for the up rate and down rate are selected (function: LMN_ROC), the changeover between the automatic and manual mode can be achieved smoothly without sudden changes in the manipulated variable.

Manual Value

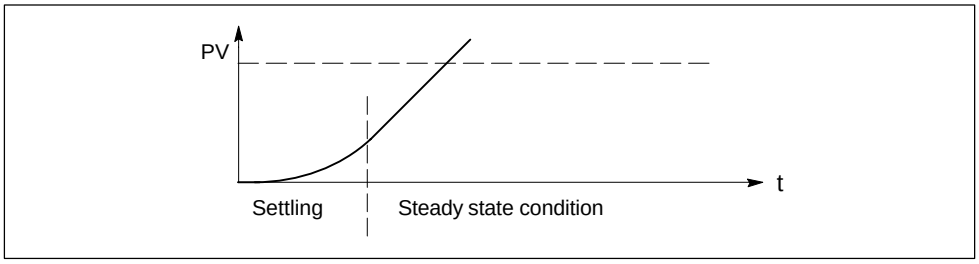
A value injected into the interrupted loop (→ manual mode) as an absolute value or as an increment (using the up or down switch) as a percentage of the range.

Modular PID Control

A modular control system is a controller structure in which the user can configure the signal processing and control functions extremely freely. Controllers configured in this way can be structured to meet the specific requirements of a task (separate S7 software package).

Non Balanced Process

A non balanced process is a process in which the slope of the process variable as a step response to a disturbance or manipulated variable change is proportional to the input step in the steady-state condition (I action).



Normalization

Normalization is a technique (algorithm) for converting the physical values of a process to the internal percentages used by the standard controller and converting the percentages to physical values at the output. The normalization curve is determined by the start value (OFFSET) and the slope (FACTOR).

Numerical Representation

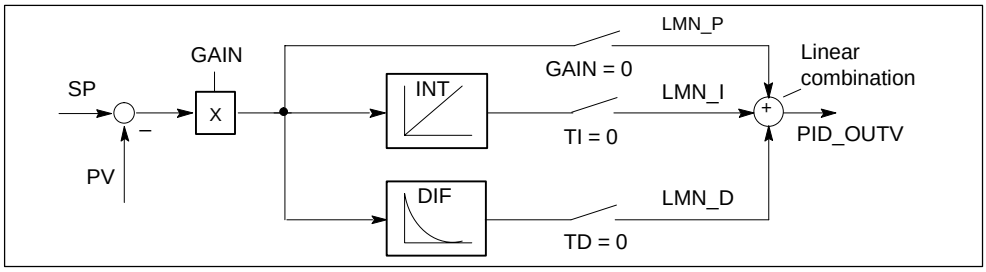
The values of analog values are implemented as floating point numbers (format: 32 bit words, range of values: $8{,}43*10^{-37}$ to $3{,}37*10^{38}$). Values denoting times are implemented as time values in the form of 16-bit BCD numbers (format: 16-bit words, range of values: 0 to 9990 seconds).

Operating Point

The operating point identifies the manipulated value at which the deviation of the process variable from the setpoint becomes zero. This value is important for controllers without an I action in which a steady state error is necessary to maintain the required manipulated value. If no steady state error is required, the operating point parameters must be adapted accordingly.

Parallel Structure

The parallel structure is a special type of signal processing in the controller (mathematical processing). The P, I and D components are calculated parallel to each other with no interaction and then totalled.



P Algorithm

Algorithm for calculating an output signal in which there is a proportional relationship between the error signal and manipulated variable change. Characteristics: steady-state error signal, not to be used with processes including dead time.

PI Algorithm

Algorithm for calculating an output signal in which the change in the manipulated variable is made up of a component proportional to the error signal and an I component proportional to the error signal and time. Characteristics: no steady-state error signal, faster compensation than with an I algorithm, suitable for all processes.

PID Algorithm

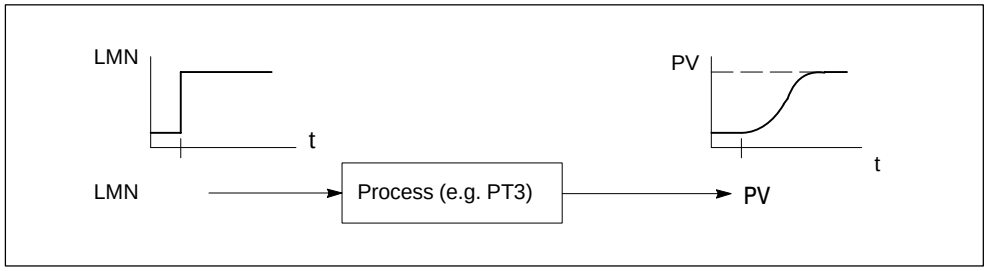
Algorithm for calculating an output signal formed by multiplication, integration and differentiation of the error signal. The PID algorithm is a → parallel structure. Characteristics: high degree of control quality can be achieved providing the dead time of the process is not greater than the other time constants.

PLC

A programmable logic controller consisting of one or more central processing units (CPU), peripheral units with digital/analog inputs and or outputs, units for interconnection and communication with other system units and in some cases with a power supply unit.

Process (Unit)

The process is the part of the system in which the process variable is influenced by the manipulated variable (by changing the level of energy or mass). The process can be divided into the actuator and the actual process being controlled.

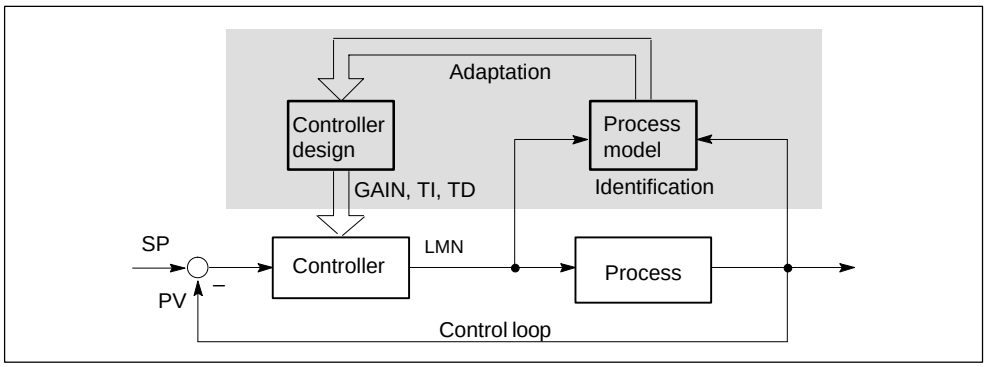


Process Control Unit

The process control unit designates that part of the control loop which is used to influence the manipulated variable at the process input.

Process Identification

Process identification is a function of the configuration tool that provides information about the transfer function and structure of the process. The result is a device-independent process model that describes the static and dynamic response of the process. The optimum settings and design of the controller are calculated based on this model



Process Simulation

Process simulation is a function for simulating a control loop with specific time lag elements so that a real process can be simulated. After stimulating the ”process” with disturbance variables or a setpoint step change, the process variables can be archived or displayed in the form of a curve.

Process Variable

Process variable (output variable of the process) that is compared with the instantaneous value of the setpoint.

Pulse Width Modulation

Pulse width modulation is a method of influencing the manipulated variable at a discontinuous output. The calculated manipulated value as a percentage is converted to a proportional signal pulse time T_p at the manipulated variable output, for example, 100 % $T_p = T_A$ or = CYCLE.

Ramp Soak

The ramp soak is a function for generating curves for the setpoint according to a fixed program. The time-dependent settings of the output variable are defined using time slices and linear interpolation. The ramp soak can be repeated cyclically.

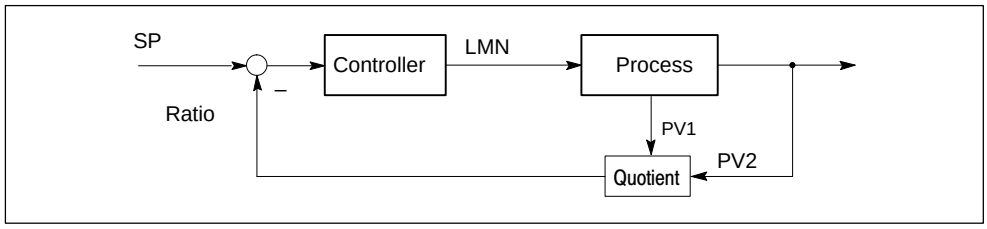
Rate of Change (ROC)

Method of limiting the rate of change of analog values (separate for up and down rate). Step changes at the input become finite slopes at the output.

Ratio Control

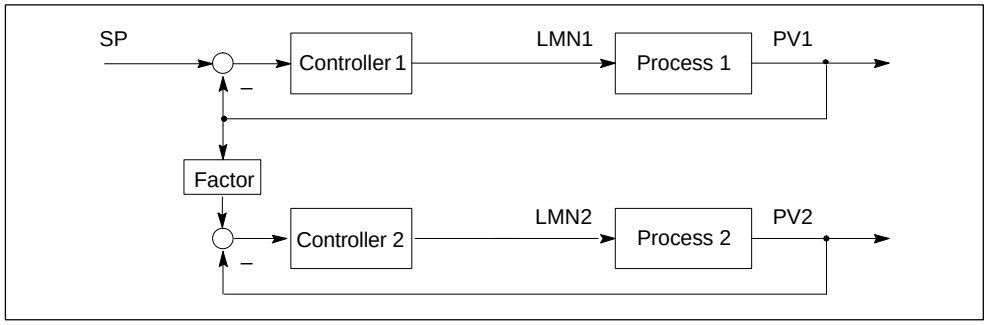
- Single loop ratio controller

  A single loop ratio controller is used when the ratio of two process variables is more important than the absolute values of the variables.



- Multi-loop ratio controller

  In a multi-loop ratio controller, the ration of the two process variables PV1 and PV2 must be kept constant. To do this, the setpoint of the 2nd control loop is calculated from the process variable of the 1st control loop. Even if the process variable PV1 changes dynamically, the ratio is maintained.



Reset Time T_N

The reset time determines the time response of the integral component in the PI or PID controller (T_N = TI).

Response Threshold

The response threshold of the step controller is adapted automatically in a three-step unit (THREE_ST). This lerads to a reduction of the pulse and reduces wear and tear on switch elements. In addition the length of the pulses and the break duration can be set by means of the minimum pulse time or the minimum break time.

The minimum pulse time (PULSE_TM) or the minimum break time (BREAK_TM) determine the minimum time that an output must be on or off.

Restart

When a controller is restarted, it starts up again using the data and operating state it had when it was interrupted. This means that the controller continues to work with the values calculated at the time of the interruption.

Sampling Controller

A sampling controller is a controller that acquires the analog input values (setpoint, process variable) at constant intervals, saves them until the next sampling point and calculates the manipulated variable.

Sampling Time T_A

The sampling time is the time between two sampling points or processing cycles of the control algorithm for a particular measurement/control channel. These intervals are constant and can be adapted to the time response of the process:

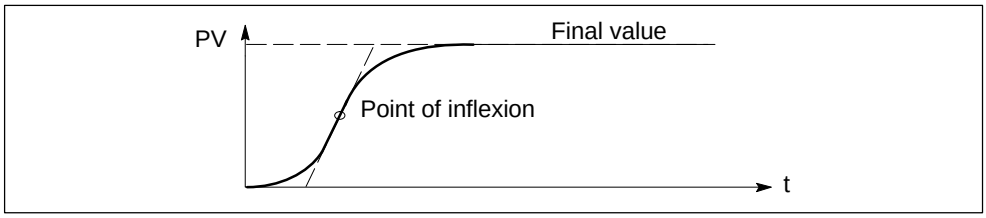
T_A = CYCLE.

Selection Control

Selection control is used in processes that demand different control structures under different operating conditions. A criterion must be selected to trigger the changeover from one structure to another.

Self-Regulating Process

A self-regulating process is a process in which a steady state is achieved after a step response (1st order time lag).



Setpoint

The setpoint is the instantaneous reference input that specifies the desired value or course of the process variable being controlled. The setpoint is the value that the process variable should adopt under the influence of the controller.

Setpoint Generator

The setpoint generator is a function with which the user can change the setpoint value using switches. During the first 3 seconds after activating the function, the rate of change is only 10% of the final rate of change that is proportional to the size of the permitted adjustment range.

Settling Time

With a step response in a higher order self-regulating process, the section created where the tangent intersects the line parallel to the time axis drawn from the start to end value.

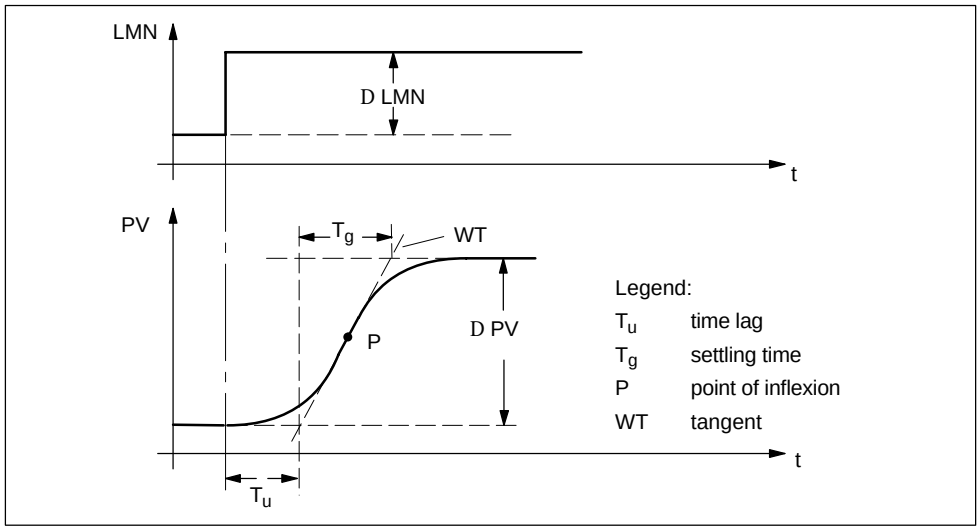


Figure Step Response of a Self-Regulating Third Order Process

The control settling time is the time between leaving the previous steady state until the process variable is finally re-established within the tolerance band (± 5 %) around the setpoint after changes in the setpoint or after disturbances.

Signal Flow Chart

The signal flow chart represents the important relationships within a control system or process. The chart consists of transfer blocks representing the transfer response of the real elements of the control loop and lines indicating the direction in which influence is exerted.

Square Root

The square root function SQRT linearizes quadratic characteristic curves.

Standard PID Control

A standard PID control is a complete and fixed controller structure containing all the functions of a controller application. The user can activate or deactivate functions using software switches.

Startup

An ”automatic startup” is started when power returns after a power down, ”a manual startup” is triggered by a switch or by a command (→ complete restart, → restart).

Step Controller

A step controller is a quasi continuous controller with a discontinuous output (and motor-driven actuator with an I action). The actuator has a three-step response, for example up – stop – down (or open – hold – close)

(→ Three-step controller).

Three-Step Controller

A controller that can only adopt three discrete states; for example ”heat – off cool” or ”right – stop – left”

(→ step controller).

Trapezoidal Rule

Method for algorithmic simulation of continuous I and D and delay elements by means of recursive differential calculation. When the trapezoidal rule is used, the control algorithm of the digital controller can be considered as an analog controller.

Two-Step Controller

A two-step controller is a controller that can only set two states for the manipulated variable (for example, on – off).

Value Range

The controller operates internally with percentages in the floating point format (for example –100,0 to +100,0). At certain input parameters, for example at external setpoints, physical values can also be entered in the floating point range of STEP 7 (→ Numerical representation).

Index

E

F

I

L

M

N

O

P

Q

R

S

T

U

V

W