Unity Pro
Control
Block Library

04/2015

33002535.14

www.schneider-electric.com

Schneider Electric

The information provided in this documentation contains general descriptions and/or technical characteristics of the performance of the products contained herein. This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither Schneider Electric nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information contained herein. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

Table of Contents

	Safety Information	**9**
	About the Book	**11**
Part I	**General Information**	**13**
Chapter 1	**Block Types and their Applications**	**15**
	Block Types	16
	FFB Structure	18
	EN and ENO	21
Chapter 2	**Block Availability on the Various Hardware Platforms**	**25**
	Block Availability on the Various Hardware Platforms	25
Chapter 3	**General information about the Control block library**	**29**
	Operating mode	30
	Scanning	32
	Error management	33
	Convention	35
Part II	**Conditioning**	**37**
Chapter 4	**DTIME: Delay**	**39**
	Description	40
	Parametering	43
	Initialization and Operating modes	45
	Example for measuring a rate of flow	46
	Runtime error	47
Chapter 5	**INTEGRATOR: Integrator with limit**	**49**
	Description	50
	Detailed description	54
Chapter 6	**LAG_FILTER: Time lag device: 1st order**	**57**
	Description	58
	Detailed description	61
Chapter 7	**LDLG: PD device with smoothing**	**63**
	Description	64
	Detailed description	67
	Examples of function block LDLG	68
Chapter 8	**LEAD: Differentiator with smoothing**	**71**
	Description	72
	Detailed description	75

Chapter 9 MFLOW: Mass flow block 77
Description 78
Detailed description 81
Runtime error 83
Chapter 10 QDTIME: Deadtime device 85
Description 86
Detailed description 89
Chapter 11 SCALING: Scaling 91
Description 92
Parametering 94
Runtime error 95
Chapter 12 TOTALIZER: Integrator 97
Description 98
Formulas 102
Detailed description 103
Runtime error 107
Chapter 13 VEL_LIM: Velocity limiter 109
Description 110
Detailed description 113
Part III Controller 115
Chapter 14 AUTOTUNE: Automatic tuner setting 117
Description 118
Principle of autotuning 123
Identification principle 125
Parametering 126
Controller coupling 129
Diagnosis 131
Causes of autotuning termination 133
Generating a test after stopping the autotuning 135
Runtime error 139
Chapter 15 IMC: Model corrector 141
Description 142
Delay management 149
Block diagram of the IMC controller 150
Execution Error 151

Chapter 16 PI_B: Simple PI controller 153
Description 154
Formulas 159
Parametering 160
Detailed equations 163
Runtime error 165
Chapter 17 PIDFF: Complete PID controller 167
Description 168
Formulas 173
Structure diagram of the PIDFF controller 175
Parametering 177
Operating mode 181
Detailed equations 182
Detailed equations: Incremental algorithm PID controller 184
Detailed equations: Incremental algorithms in integral mode 186
Example for the PIDFF block 188
Runtime error 193
Chapter 18 SAMPLETM: Sample time 195
Description 195
Chapter 19 STEP2: Two point controller 197
Description 198
Detailed description 201
Runtime error 203
Chapter 20 STEP3: Three point controller 205
Description 206
Detailed description 209
Runtime error 211
Part IV Mathematics 213
Chapter 21 COMP_DB: Comparison 215
Description 216
Detailed description 219
Chapter 22 K_SQRT: Square root 221
Description 221
Chapter 23 MULDIV_W: Multiplication/Division 225
Description 225
Chapter 24 SUM_W: Summer 229
Description 229

Part V Measurement 233
Chapter 25 AVGMV: Moving average with fixed window size 235
Description 236
Detailed description 239
Chapter 26 AVGMV_K: Moving average with frozen correction factor 241
Description 242
Detailed description 245
Chapter 27 DEAD_ZONE, DEAD_ZONE_REAL: Dead zone 247
Description 248
Detailed description 250
Chapter 28 LOOKUP_TABLE1: Polygon with interpolation of the 1st order 253
Description 254
Detailed description 256
Chapter 29 SAH: Detecting and holding a rising edge 259
Description 259
Chapter 30 HYST_*: Indicator signal for maximum value delimiters with hysteresis 261**
Description 262
Detailed description 264
Chapter 31 INDLIM_*: Indicator signal for delimiters with hysteresis 265**
Description 266
Detailed description 269
Part VI Output Processing 271
Chapter 32 MS: Manual control of an output 273
Description 274
Detailed description 278
Example 281
Runtime error 282
Chapter 33 MS_DB: Manually controlling and output with dead zone 285
Description 286
Detailed description 290
Runtime error 294

Chapter 34 PWM1: Pulse width modulation 297
Description 298
Detailed description 301
Example of the PWM1 block 303
Chapter 35 SERVO: Control for servo motors 305
Description 306
Parametering 310
SERVO function block algorithms 312
Operating mode 313
Examples of function block SERVO 314
Runtime error 321
Chapter 36 SPLRG: Controlling 2 actuators 323
Description 324
Detailed description 326
Runtime error 328
Part VII Setpoint management 331
Chapter 37 RAMP: Ramp generator 333
Description 334
Detailed description 336
Runtime error 338
Chapter 38 RATIO: Ratio controller 339
Description 340
Detailed description 343
Runtime error 345
Chapter 39 SP_SEL: Setpoint switch 347
Description 348
Detailed description 351
Runtime error 354
Appendices 355
Appendix A EFB Error Codes and Values 357
Tables of Error Codes for the CONT_CTL Library 358
Common Floating Point Errors 365
Glossary 367
Index 371

Safety Information

Important Information

NOTICE

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

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

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

⚠ DANGER

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

⚠ WARNING

WARNING indicates a hazardous situation which, if not avoided, **could result in** death or serious injury.

⚠ CAUTION

CAUTION indicates a hazardous situation which, if not avoided, **could result** in minor or moderate injury.

NOTICE

NOTICE is used to address practices not related to physical injury.

PLEASE NOTE

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

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and its installation, and has received safety training to recognize and avoid the hazards involved.

About the Book

At a Glance

Document Scope

This document describes the functions and function blocks of the Control library.

Validity Note

This document is valid for Unity Pro 10.0 or later.

Related Documents

Title of Documentation	Reference Number
Unity Pro Program Languages and structure, Reference Manual	35006144 (English), 35006145 (French), 35006146 (German), 35013361 (Italian), 35006147 (Spanish), 35013362 (Chinese)
Unity Pro System Bits and Words, Reference Manual	EIO0000002135 (English), EIO0000002136 (French), EIO0000002317 (German), EIO0000002138 (Italian), EIO0000002139 (Spanish), EIO0000002140 (Chinese)

You can download these technical publications and other technical information from our website at www.schneider-electric.com.

Part I
General Information

Overview

This section contains general information about the Control library.

NOTE: For a detailed description of system objects (%S and %SW), refer to *Unity Pro System Bits and Words, Reference Manual*.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
1	Block Types and their Applications	15
2	Block Availability on the Various Hardware Platforms	25
3	General information about the Control block library	29

Chapter 1
Block Types and their Applications

Overview

This chapter describes the different block types and their applications.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Block Types	16
FFB Structure	18
EN and ENO	21

Block Types

Block Types

Block Types

Different block types are used in Unity Pro. The general term for the block types is FFB.

There are the following types of block:

- Elementary Function (EF)
- Elementary Function Block (EFB)
- Derived Function Block (DFB)
- Procedure

NOTE: Motion Function Blocks are not available on the Quantum platform.

Elementary Function

Elementary functions (EF) have no internal status and one output only. If the input values are the same, the output value is the same for the executions of the function, for example the addition of two values gives the same result at every execution.

An elementary function is represented in the graphical languages (FBD and LD) as a block frame with inputs and an output. The inputs are represented on the left and the outputs on the right of the block frame. The name of the function, that is the function type, is shown in the center of the block frame.

The number of inputs can be increased with some elementary functions.

⚠ CAUTION
UNEXPECTED BEHAVIOR OF EQUIPMENT For Unity Pro V4.0 and earlier versions, do not use links to connect function blocks outputs, when your application relies on persistent output data of an EF. **Failure to follow these instructions can result in injury or equipment damage.**

NOTE: With Unity Pro V4.0 and earlier versions the deactivation of an EF (EN=0) causes links connected to its Input/Output to be reset. To transfer the state of the signal do not use a link. A variable must be connected to the EF's output and must be used to connect the input of the element. With Unity Pro V4.1 and later versions you can maintain the output links even if an EF is deactivated by activating the option **Maintain output links on disabled EF (EN=0)** via the menu **Tools →Program →Languages →Common**.

Elementary Function Block

Elementary function blocks (EFB) have an internal status. If the inputs have the same values, the value on the outputs can have another value during the individual executions. For example, with a counter, the value on the output is incremented.

An elementary function block is represented in the graphical languages (FBD and LD) as a block frame with inputs and outputs. The inputs are represented on the left and the outputs on the right of the block frame. The name of the function block, that is the function block type, is shown in the center of the block frame. The instance name is displayed above the block frame.

Derived Function Block

Derived function blocks (DFBs) have the same properties as elementary function blocks. They are created by the user in the programming languages FBD, LD, IL and/or ST.

Procedure

Procedures are functions with several outputs. They have no internal state.

The only difference from elementary functions is that procedures can have more than one output and they support variables of the `VAR_IN_OUT` data type.

Procedures do not return a value.

Procedures are a supplement to IEC 61131-3 and must be enabled explicitly.

There is no visual difference between procedures and elementary functions.

⚠ CAUTION

UNEXPECTED BEHAVIOR OF EQUIPMENT

For Unity Pro V4.0 and earlier versions, do not use links to connect function blocks outputs, when your application relies on persistent output data of an EF.

Failure to follow these instructions can result in injury or equipment damage.

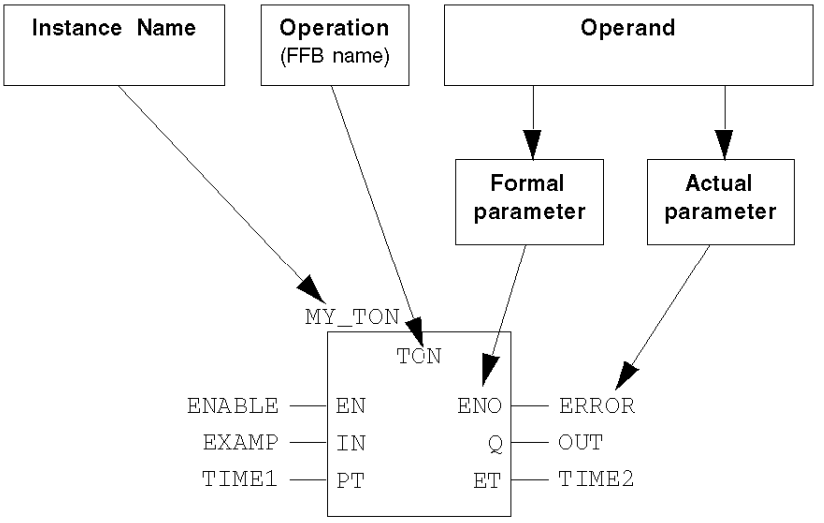
NOTE: With Unity Pro V4.0 and earlier versions the deactivation of an EF (EN=0) causes links connected to its Input/Output to be reset. To transfer the state of the signal do not use a link. A variable must be connected to the EF's output and must be used to connect the input of the element. With Unity Pro V4.1 and later versions you can maintain the output links even if an EF is deactivated by activating the option **Maintain output links on disabled EF (EN=0)** via the menu **Tools →Program →Languages →Common**.

FFB Structure

Structure

Each FFB is made up of an operation (name of the FFB), the operands are required for the operation (formal and actual parameters) and an instance name for elementary/derived function blocks.

Call of a function block in the FBD programming language:



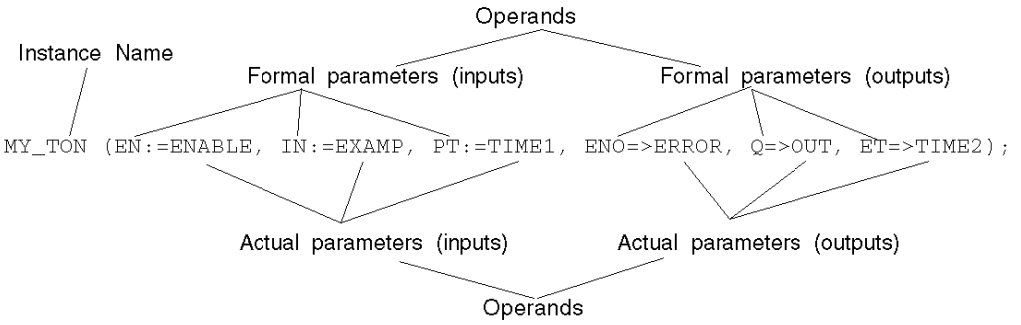
⚠ CAUTION

UNEXPECTED APPLICATION BEHAVIOR

Do not call several times the same block instance within a PLC cycle

Failure to follow these instructions can result in injury or equipment damage.

Formal call of a function block in the ST programming language:



Operation

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

Operand

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

Formal/actual parameters

Inputs and outputs are required to transfer values to or from an FFB. These are called formal parameters.

Objects are linked to formal parameters; these objects contain the current process states. They are called actual parameters.

At program runtime, the values from the process are transferred to the FFB via the actual parameters and then output again after processing.

The data type of the actual parameters must match the data type of the input/output (formal parameters). The only exceptions are generic inputs/outputs whose data type is determined by the actual parameter. If the actual parameters consist of literals, a suitable data type is selected for the function block.

FFB Call in IL/ST

In text languages IL and ST, FFBs can be called in formal and in informal form. For detail, refer to chapter *Programming Language (see Unity Pro, Program Languages and Structure, Reference Manual)*.

Example of a formal function call:

```
out:=LIMIT (MN:=0, IN:=var1, MX:=5);
```

Example of an informal function call:

```
out:=LIMIT (0, var1, 5);
```

NOTE: The use of EN and ENO is only possible for formal calls.

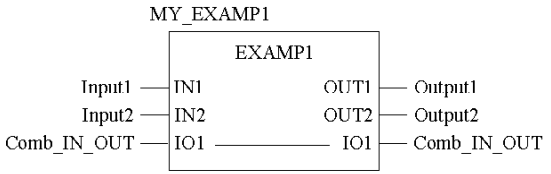
VAR_IN_OUT variable

FFBs are often used to read a variable at an input (input variables), to process it and to output the altered values of the **same** variable (output variables).

This special type of input/output variable is also called a VAR_IN_OUT variable.

The input and output variable are linked in the graphic languages (FBD and LD) using a line showing that they belong together.

Function block with VAR_IN_OUT variable in FBD:



Function block with VAR_IN_OUT variable in ST:

```
MY_EXAMP1 (IN1:=Input1, IN2:=Input2, IO1:=Comb_IN_OUT,
           OUT1=>Output1, OUT2=>Output2);
```

The following points must be considered when using FFBs with VAR_IN_OUT variables:

- The VAR_IN_OUT inputs must be assigned a variable.
- Literals or constants cannot be assigned to VAR_IN_OUT inputs/outputs.

The following additional limitations apply to the graphic languages (FBD and LD):

- When using graphic connections, VAR_IN_OUT outputs can only be connected with VAR_IN_OUT inputs.
- Only one graphical link can be connected to a VAR_IN_OUT input/output.
- Different variables/variable components can be connected to the VAR_IN_OUT input and the VAR_IN_OUT output. In this case the value of the variables/variable component on the input is copied to the output variables/variable component.
- No negations can be used on VAR_IN_OUT inputs/outputs.
- A combination of variable/address and graphic connections is not possible for VAR_IN_OUT outputs.

EN and ENO

Description

An EN input and an ENO output can be configured for the FFBs.

If the value of EN is equal to "0" when the FFB is invoked, the algorithms defined by the FFB are not executed and ENO is set to "0".

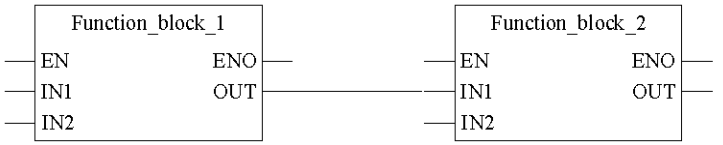
If the value of EN is equal to "1" when the FFB is invoked, the algorithms defined by the FFB will be executed. After the algorithms have been executed successfully, the value of ENO is set to "1". If certain error conditions are detected when executing these algorithms, ENO is set to "0".

If the EN pin is not assigned a value, when the FFB is invoked, the algorithm defined by the FFB is executed (same as if EN equals to "1"), Please refer to *Maintain output links on disabled EF (see Unity Pro, Operating Modes)*.

If the algorithms are executed successfully, then value of ENO is set to "1", else ENO is set to "0".

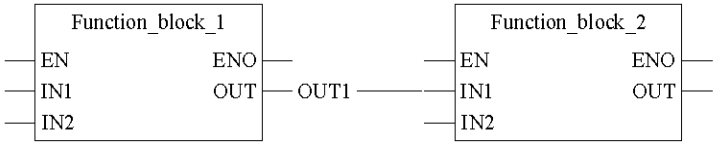
If ENO is set to "0" (caused by EN=0 or a detected error condition during execution or unsuccessful algorithm execution):

- Function blocks
  - EN/ENO handling with function blocks that (only) have one link as an output parameter:



If EN from FunctionBlock_1 is set to "0", the output connection OUT from FunctionBlock_1 retains the status it had in the last correctly executed cycle.

  - EN/ENO handling with function blocks that have one variable and one link as output parameters:



If EN from FunctionBlock_1 is set to "0", the output connection OUT from FunctionBlock_1 retains the status it had in the last correctly executed cycle. The variable OUT1 on the same pin, either retains its previous status or can be changed externally without influencing the connection. The variable and the link are saved independently of each other.

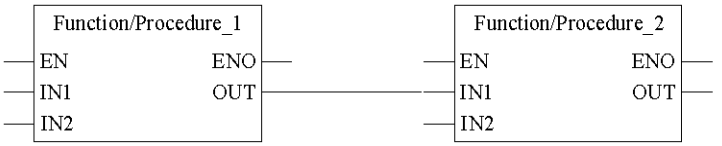
- Functions/Procedures

⚠ CAUTION
UNEXPECTED BEHAVIOR OF EQUIPMENT For Unity Pro V4.0 and earlier versions, do not use links to connect function blocks outputs, when your application relies on persistent output data of an EF. **Failure to follow these instructions can result in injury or equipment damage.**

NOTE: With Unity Pro V4.0 and earlier versions the deactivation of an EF (EN=0) causes links connected to its Input/Output to be reset. To transfer the state of the signal do not use a link. A variable must be connected to the EF's output and must be used to connect the input of the element. With Unity Pro V4.1 and later versions you can maintain the output links even if an EF is deactivated by activating the option **Maintain output links on disabled EF (EN=0)** via the menu **Tools →Program →Languages →Common**.

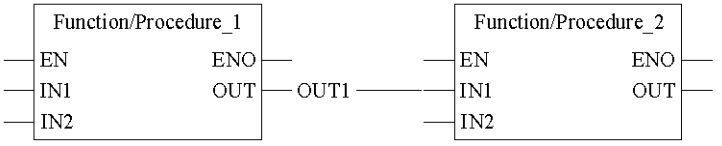
As defined in IEC61131-3, the outputs from deactivated functions (`EN`-input set to "0") is undefined. (The same applies to procedures.)
Here is an explanation of the output status in this case:

- EN/ENO handling with functions/procedures that (only) have one link as an output parameter:



If `EN` from `Function/Procedure_1` is set to "0", the output connection `OUT` from `Function/Procedure_1` is also set to "0".

- EN/ENO handling with function blocks that have one variable and one link as output parameters:



If `EN` from `Function/Procedure_1` is set to "0", the output connection `OUT` from `Function/Procedure_1` is also set to "0". The variable `OUT1` on the same pin, either retains its previous status or can be changed externally without influencing the connection. The variable and the link are saved independently of each other.

The output behavior of the FFBs does not depend on whether the FFBs are called up without `EN`/`ENO` or with `EN`=1.

Conditional/Unconditional FFB Call

"Unconditional" or "conditional" calls are possible with each FFB. The condition is realized by pre-linking the input EN.

- EN connected
  conditional calls (the FFB is only processed if EN = 1)
- EN shown, hidden, and marked TRUE, or shown and not occupied
  unconditional calls (FFB is processed independent from EN)

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

⚠ CAUTION
UNEXPECTED APPLICATION EQUIPMENT
Do not disable function blocks with internal time function during their operation.
Failure to follow these instructions can result in injury or equipment damage.

Note for IL and ST

The use of EN and ENO is only possible in the text languages for a formal FFB call, e.g.

```
MY_BLOCK (EN:=enable, IN1:=var1, IN2:=var2,
ENO=>error, OUT1=>result1, OUT2=>result2);
```

Assigning the variables to ENO must be done with the operator =>.

With an informal call, EN and ENO cannot be used.

Chapter 2
Block Availability on the Various Hardware Platforms

Block Availability on the Various Hardware Platforms

Introduction

Not all blocks are available on all hardware platforms. The blocks available on your hardware platform can be found in the following tables.

NOTE: The functions and function blocks in this library are not defined in IEC 61131-3.

Conditioning

Availability of the blocks:

Block name	Block type	M340	M580	Quantum	Momentum	Premium
DTIME	EFB	+	+	+	+	+
INTEGRATOR	EFB	+	+	+	+	+
LAG_FILTER	EFB	+	+	+	+	+
LDLG	EFB	+	+	+	+	+
LEAD	EFB	+	+	+	+	+
MFLOW	EFB	+	+	+	+	+
QDTIME	EFB	+	+	+	+	+
SCALING	EFB	+	+	+	+	+
TOTALIZER	EFB	+	+	+	+	+
VEL_LIM	EFB	+	+	+	+	+
Legend:						
+	Yes					
-	No					

Controller

Availability of the blocks:

Block name	Block type	M340	M580	Quantum	Momentum	Premium
AUTOTUNE	EFB	+	+	+	+	+
IMC	EFB	+	+	+	+	+
PI_B	EFB	+	+	+	+	+
PIDFF	EFB	+	+	+	+	+
SAMPLETM	EFB	+	+	+	+	+
STEP2	EFB	+	+	+	+	+
STEP3	EFB	+	+	+	+	+
Legend:						
+	Yes					
-	No					

Mathematics

Availability of the blocks:

Block name	Block type	M340	M580	Quantum	Momentum	Premium
COMP_DB	EFB	+	+	+	+	+
K_SQRT	EF	+	+	+	+	+
MULDIV_W	EF	+	+	+	+	+
SUM_W	EF	+	+	+	+	+
Legend:						
+	Yes					
-	No					

Measurement

Availability of the blocks:

Block name	Block type	M340	M580	Quantum	Momentum	Premium
AVGMV	EFB	+	+	+	+	+
AVGMV_K	EFB	+	+	+	+	+
DEAD_ZONE, DEAD_ZONE_REAL	EF	+	+	+	+	+
LOOKUP_TABLE1	Procedure	+	+	+	+	+
SAH	EFB	+	+	+	+	+
HYST_***	EFB	+	+	+	+	+
INDLIM_***	EFB	+	+	+	+	+
Legend:						
+	Yes					
-	No					

Output Processing

Availability of the blocks:

Block name	Block type	M340	M580	Quantum	Momentum	Premium
MS	EFB	+	+	+	+	+
MS_DB	EFB	+	+	+	+	+
PWM1	EFB	+	+	+	+	+
SERVO	EFB	+	+	+	+	+
SPLRG	EFB	+	+	+	+	+
Legend:						
+	Yes					
-	No					

Set point processing

Availability of the blocks:

Block name	Block type	M340	M580	Quantum	Momentum	Premium
RAMP	EFB	+	+	+	+	+
RATIO	EFB	+	+	+	+	+
SP_SEL	EFB	+	+	+	+	+
Legend:						
+	Yes					
-	No					

Chapter 3
General information about the Control block library

Overview

This section contains general information about the Control block library.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Operating mode	30
Scanning	32
Error management	33
Convention	35

Operating mode

Operating mode

Several function blocks have integrated operating mode control available.

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

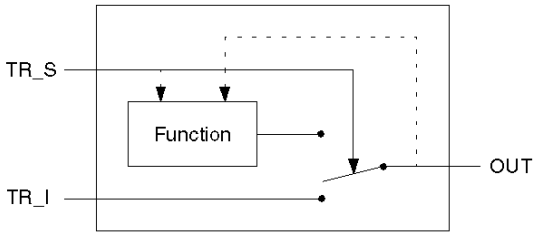
The Order of priorities of the operating modes are explained below.

Tracking

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

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

Tracking mode



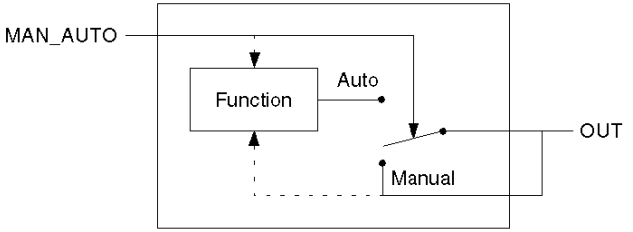
This operating mode can be used in various situations:
- Initializing during the start phase,
- Tracking operating mode with a redundant PLC, to guarantee a bumpless start for the Standby device,
- Controlling the operating mode using a program, for example to avoid direct control of the manipulated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

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

Manual/Automatic

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

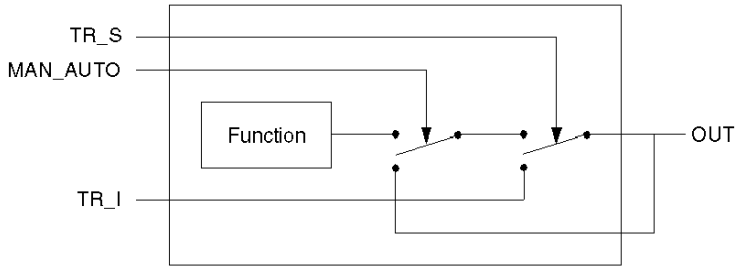
Manual/Automatic mode



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

Order of priorities of the operating mode

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



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

Scanning

Scanning

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

Set time intervals provide the following advantages:

- Run time optimization of the PLC program by dividing the control operations into several cycles,
- improved control quality, where scanning the servo-loop too frequently is prevented
- Minimizing the demands on the actuators

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

If the scan interval of the servo-loop exceeds 1 second, the function block *MS: Manual control of an output, page 273* should be switched to the function blocks *PIDFF: Complete PID controller, page 167* and *PI_B: Simple PI controller, page 153* so that the servo-loops can be controlled manually independently of the scan interval.

Error management

Principle

This section describes the error recording and notification routines of

function blocks in the `Conditioning`, `Controller`, `Output Processing` and `setpoint processing` families.

Most function blocks in these families are provided with a `STATUS` output word.

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

Status word

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

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

Note 1 (input parameter)

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

Note 2 (thresholds)

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

Convention

Specifying the convention

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

Part II
Conditioning

Overview

This section describes the elementary functions and elementary function blocks of the `Conditioning` family.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
4	DTIME: Delay	39
5	INTEGRATOR: Integrator with limit	49
6	LAG_FILTER: Time lag device: 1st order	57
7	LDLG: PD device with smoothing	63
8	LEAD: Differentiator with smoothing	71
9	MFLOW: Mass flow block	77
10	QDTIME: Deadtime device	85
11	SCALING: Scaling	91
12	TOTALIZER: Integrator	97
13	VEL_LIM: Velocity limiter	109

Chapter 4
DTIME: Delay

Introduction

This chapter describes the DTIME block.

What Is in This Chapter?

This chapter contains the following topics:

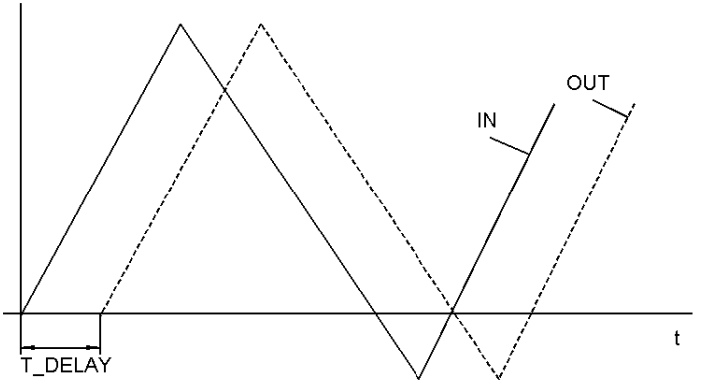
Topic	Page
Description	40
Parametering	43
Initialization and Operating modes	45
Example for measuring a rate of flow	46
Runtime error	47

Description

Function description

The DTIME function block generates a delay when transfering the numerical input value [IN]. The numerical output variable OUT generates the same behavior as the numerical input value when the delay T_DELAY, which can vary, is included.

Behavior of the function block DTIME:



EN and ENO can be configured as additional parameters.

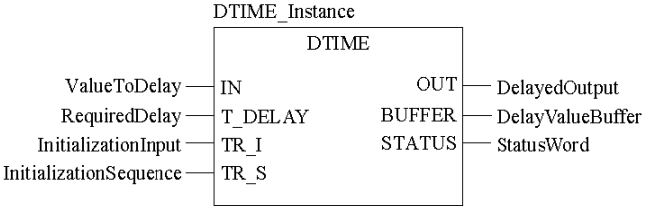
Formula

This function block implements the following transfer function :

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

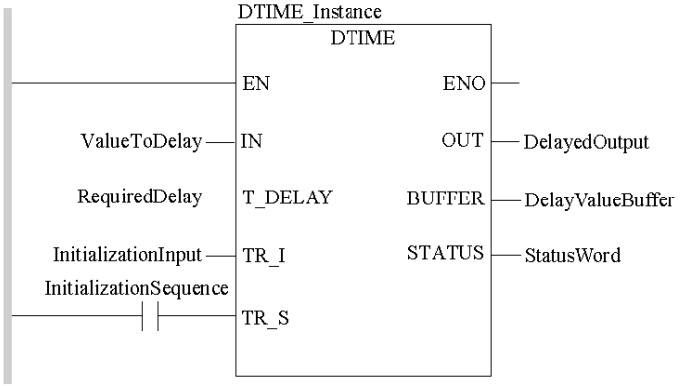
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL DTIME_Instance (IN:=ValueToDelay,
    T_DELAY:=RequiredDelay, TR_I:=InitializationInput,
    TR_S:=InitializationSequence, OUT=>DelayedOutput,
    BUFFER=>DelayValueBuffer, STATUS=>StatusWord)
```

Representation in ST

Representation:

```
DTIME_Instance (IN:=ValueToDelay,
    T_DELAY:=RequiredDelay, TR_I:=InitializationInput,
    TR_S:=InitializationSequence, OUT=>DelayedOutput,
    BUFFER=>DelayValueBuffer, STATUS=>StatusWord) ;
```

Parameter description

Input parameter description:

Parameter	Data type	Meaning
IN	REAL	Numerical value to delay
T_DELAY	TIME	Desired delay
TR_I	REAL	Initialization input
TR_S	BOOL	Initialization command

Output parameter description:

Parameter	Data type	Meaning
OUT	REAL	Delayed output
BUFFER	ANY*	Memory for the purpose of storing delayed values.
STATUS	WORD	Status word

*) It is essential for this to be linked to a variable (see"*Parametering, page 43*").

Parametering

Saving the input values (BUFFER output)

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

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

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

The following applies here

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

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

Data type of the buffer output

The BUFFER output is of the ANY type. This means any variable type can be assigned to it. It is better to define an ARRAY (table) with REAL elements. This ARRAY can contain up to 100 floating point values. With this variable type it is possible to attain a delay, which corresponds to 100 times the sampling interval of the DTIME function block.

NOTE: DTIME does only work, when the size of the actual parameter assigned to the BUFFER output is at least 4 bytes.
Valid data types are, for example, REAL, ARRAY[1..4] OF BYTE.
Invalid data types are, for example, BYTE, INT, ARRAY[1..3] OF BYTE.
If an invalid data type is connected, DTIME will not be processed and a runtime error (STATUS=2) is generated.

Procedure for large delay times

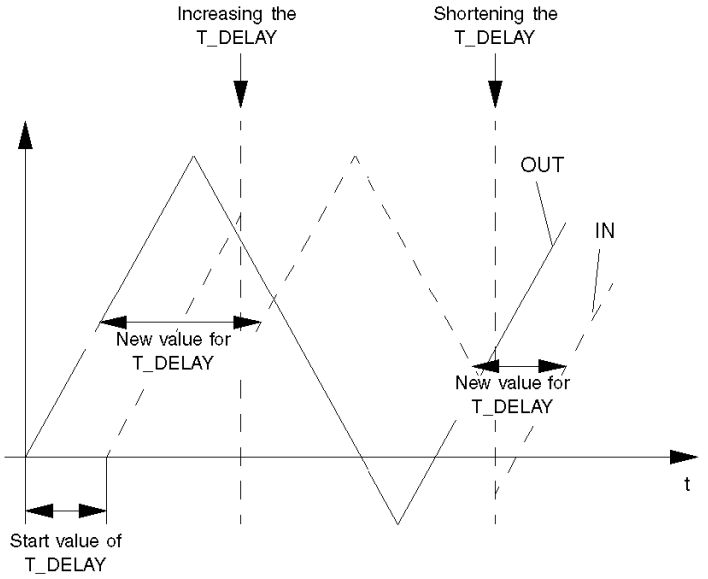
To attain delay values, which are equivalent to over 100 times the sampling interval of the function block, a larger variable must be assigned to the BUFFER parameter:

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

Dynamic modification of the T_DELAY delay

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

Representation of the dynamic modification to T_DELAY



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

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

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

Initialization and Operating modes

Initialization and Operating modes

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

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

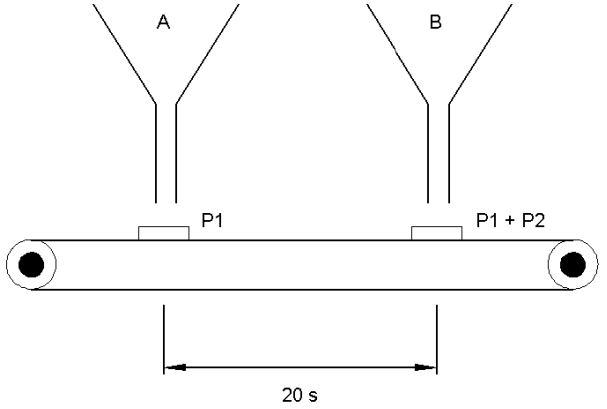
Example for measuring a rate of flow

Measuring a rate of flow

The DTIME function block can be used for example to model a process delay, which can be configured to measure flow rates or the number of revolutions of drive systems.

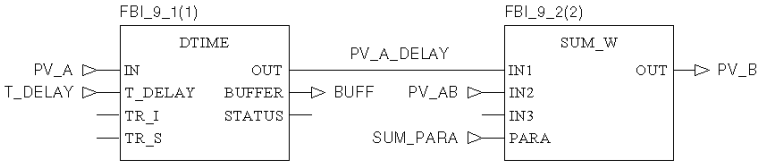
In the following example two products, A and B, are poured into a container one after the other and mixed. First, the container is placed under the dosing device for product A, to give the amount P1. Then it is moved on a conveyor belt to the dosing device for product B to give the amount P2. The time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

Measuring flow rates



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

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



Values of the data structure elements of the SUM_PARA variables:

Element of SUM_PARA	Value
SUM_PARA.K1	1
SUM_PARA.K2	1

Runtime error

Status word

The following messages are displayed in the Status word:

Bit	Value in Dec.	Value in Hex.	ENO Status	Description
Bit 0 = 1	1	0x0001	False	Error in a floating point value calculation
Bit 1 = 1	2	0x0002	False	Invalid value recorded at one of the floating point inputs
Bit 2 = 1	4	0x0004	False	Division by zero during a floating point value calculation
Bit 3 = 1	8	0x0008	False	Capacity overflow during a floating point value calculation
Bit 8 = 1	256	0x0100	True	`T_DELAY` exceeds the maximum value that can be reached on the `BUFFER` output

For a list of other possible floating point error codes, see *Common Floating Point Errors, page 365*.

Error message

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

NOTE: For a list of all block error codes and values, see *Conditioning, page 358*.

Alarm

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

Chapter 5
INTEGRATOR: Integrator with limit

Introduction

This chapter describes the INTEGRATOR block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	50
Detailed description	54

Description

Function description

The function block replicates a limited integrator.

The function block has the following properties:

- Tracking and automatic operating modes
- Manipulated variable limiting in automatic mode

NOTE: This function block performs an internal initialization in the first program cycle after a warm start or cold start (e.g. application download or power cycle) of the PLC program.
Due to this, you have to make sure that the function block is invoked in the first program cycle. In case of inkoving the function block in a later program cycle, the internal initialization will not be performed and the ouputs may deliver wrong values.

⚠ WARNING

UNEXPECTED OUTPUT BEHAVIOUR

Make sure that the function block is always invoked in the first program cycle.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

`EN` and `ENO` can be configured as additional parameters.

Formula

The transfer function is:

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

The formula for the output `OUT` is:

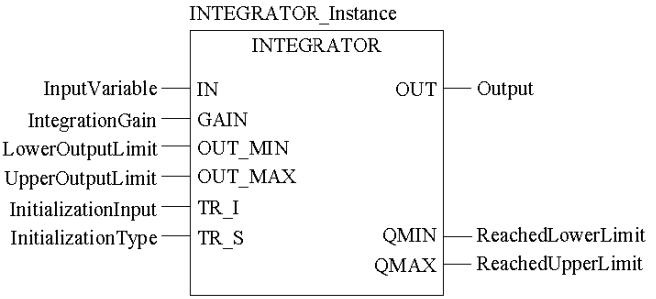
$$OUT = OUT_{(old)} + GAIN \times dt \times \frac{IN_{(new)} + IN_{(old)}}{2}$$

Meaning of the sizes

Variable	Description
$IN_{(new)}$	Current value of input `IN`
$IN_{(old)}$	Value of input `IN` from the previous cycle
$OUT_{(old)}$	Value of the output `OUT` from the previous cycle
dt	is the time differential between the current cycle and the previous cycle

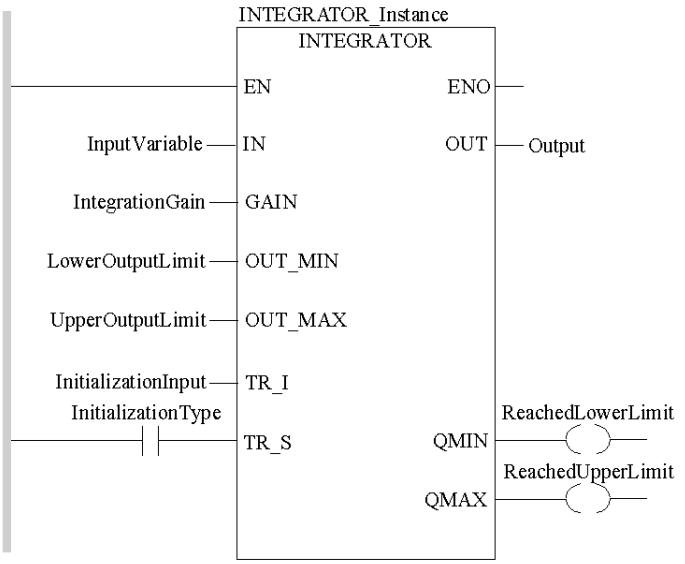
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL INTEGRATOR_Instance (IN:=InputVariable,
    GAIN:=IntegrationGain, OUT_MIN:=LowerOutputLimit,
    OUT_MAX:=UpperOutputLimit, TR_I:=InitializationInput,
    TR_S:=InitializationType, OUT=>Output,
    QMIN=>ReachedLowerLimit, QMAX=>ReachedUpperLimit)
```

Representation in ST

Representation:

```
INTEGRATOR_Instance (IN:=InputVariable,
    GAIN:=IntegrationGain, OUT_MIN:=LowerOutputLimit,
    OUT_MAX:=UpperOutputLimit, TR_I:=InitializationInput,
    TR_S:=InitializationType, OUT=>Output,
    QMIN=>ReachedLowerLimit, QMAX=>ReachedUpperLimit) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Description
IN	REAL	Input variable
GAIN	REAL	Integral gain
OUT_MIN	REAL	Lower limit
OUT_MAX	REAL	Upper limit
TR_I	REAL	Initialization input
TR_S	BOOL	Initialization type "1" = Operating mode Tracking "0" = Automatic operating mode

Description of output parameters:

Parameter	Data type	Description
OUT	REAL	Output
QMIN	BOOL	"1" = Output OUT has reached lower limit
QMAX	BOOL	"1" = Output OUT has reached upper limit

Error message

With OUT_MAX < OUT_MIN an error message appears.

NOTE: For a list of all block error codes and values, see *Conditioning, page 358*.

Detailed description

Parametering

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

The limits OUT_MAX and OUT_MIN limit the upper output as well as the lower output. Hence OUT_MIN ≤OUT ≤OUT_MAX.

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

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

Operating mode

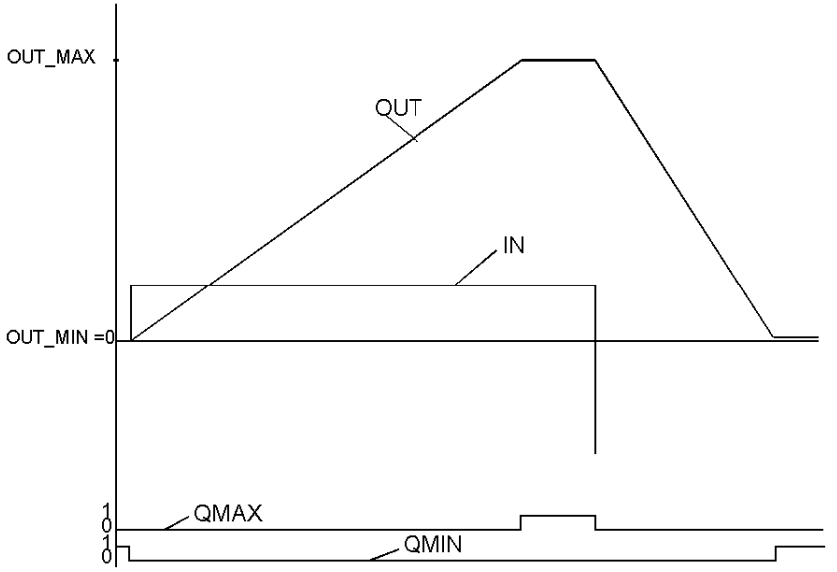
There are two operating mode selectable through the TR_S parameter input.

Operating mode	TR_S	Meaning
Automatic	0	The Function block will be handled as "Parametering" describes.
Tracking	1	The tracking value TR_I is transferred directly to the output OUT. The control output is, however, limited by OUT_MAX and OUT_MIN.

Example

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

Representation of the integrator step response:



Chapter 6
LAG_FILTER: Time lag device: 1st order

Introduction

This chapter describes the LAG_FILTER block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	58
Detailed description	61

Description

Function description

The function block represents a delay element 1st order.

The function block contains the following operating mode:

- Tracking
- Automatic

NOTE: This function block performs an internal initialization in the first program cycle after a warm start or cold start (e.g. application download or power cycle) of the PLC program.
Due to this, you have to make sure that the function block is invoked in the first program cycle. In case of inkoving the function block in a later program cycle, the internal initialization will not be performed and the ouput may deliver a wrong value.

⚠ WARNING

UNEXPECTED OUTPUT BEHAVIOUR

Make sure that the function block is always invoked in the first program cycle.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

`EN` and `ENO` can be configured as additional parameters.

Formula

The transfer function is:

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

The formula of calculation is:

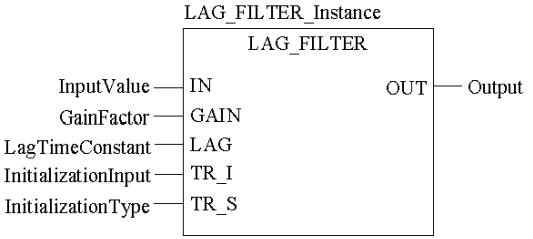
$$OUT = OUT_{(old)} + \frac{dt}{LAG + dt} \times \left(GAIN \times \frac{IN_{(old)} + IN_{(new)}}{2} - OUT_{(old)} \right)$$

Meaning of the sizes

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

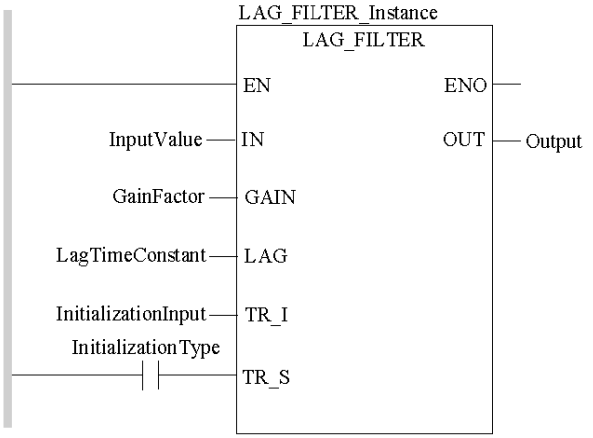
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL LAG_FILTER_Instance (IN:=InputValue, GAIN:=GainFactor,
    LAG:=LagTimeConstant,  TR_I:=InitializationInput,
    TR_S:=InitializationType, OUT=>Output)
```

Representation in ST

Representation:

```
LAG_FILTER_Instance (IN:=InputValue, GAIN:=GainFactor,
    LAG:=LagTimeConstant,  TR_I:=InitializationInput,
    TR_S:=InitializationType, OUT=>Output) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Description
IN	REAL	Input value
GAIN	REAL	Gain factor
LAG	TIME	Delayed time constants
TR_I	REAL	Initialization input
TR_S	BOOL	Initialization type "1" = Operating mode Tracking "0" = Automatic operating mode

Description of output parameters:

Parameter	Data type	Description
OUT	REAL	Output

Runtime error

For a list of all block error codes and values, see *Conditioning, page 358*.

Detailed description

Parametering

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

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

$\exp(\ t/LAG)$

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

Operating mode

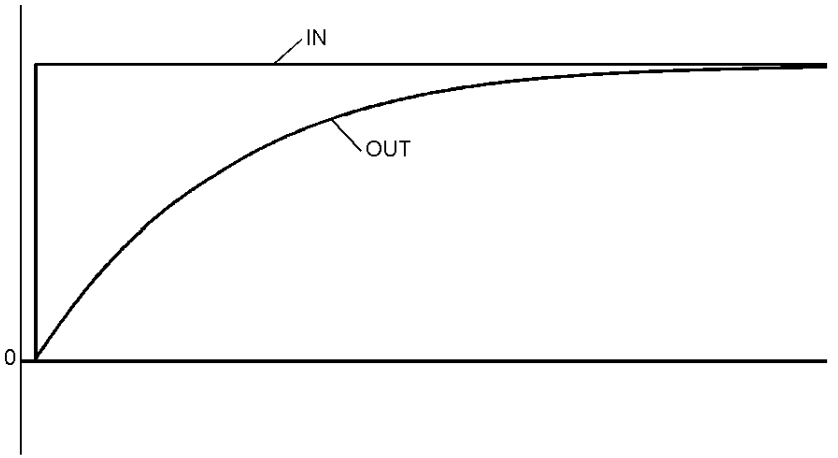
Two operating modes can be selected through the TR_S parameter input.

Operating mode	TR_S	Meaning
Automatic	0	The Function block will be handled as "Parametering" describes.
Tracking	1	The tracking value TR_I is transferred directly to the output OUT.

Example

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

Jump response of the function block LAG_FILTER when GAIN = 1



Chapter 7
LDLG: PD device with smoothing

Introduction

This chapter describes the LDLG block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	64
Detailed description	67
Examples of function block LDLG	68

Description

Function description

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

The function block has the following properties:

- Definable delay of the D-component
- Tracking and automatic modes

NOTE: This function block performs an internal initialization in the first program cycle after a warm start or cold start (e.g. application download or power cycle) of the PLC program.
Due to this, you have to make sure that the function block is invoked in the first program cycle. In case of inkoving the function block in a later program cycle, the internal initialization will not be performed and the ouput may deliver a wrong value.

⚠ WARNING

UNEXPECTED OUTPUT BEHAVIOUR

Make sure that the function block is always invoked in the first program cycle.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

`EN` and `ENO` can be configured as additional parameters.

Formula

The transfer function is:

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

The formula of calculation is:

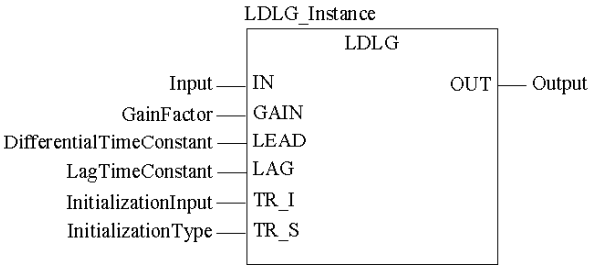
$$OUT = \frac{LAG \times OUT_{(old)} + GAIN \times ((LEAD + dt) \times IN - LEAD \times IN_{(old)})}{LAG + dt}$$

Meaning of the sizes

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

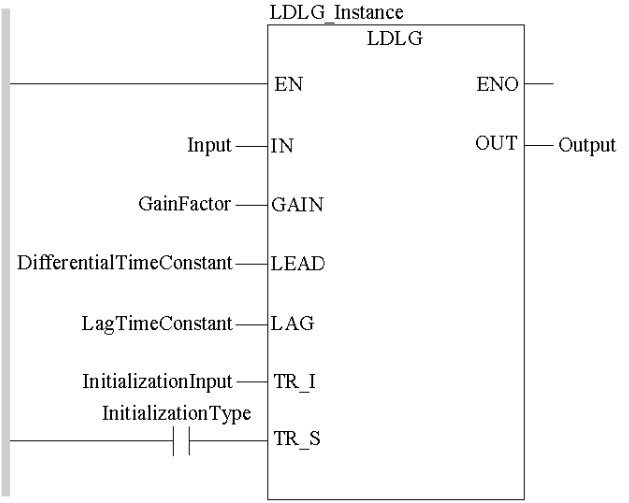
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL LDLG_Instance (IN:=Input, GAIN:=GainFactor,
    LEAD:=DifferentialTimeConstant, LAG:=LagTimeConstant,
    TR_I:=InitializationInput, TR_S:=InitializationType,
    OUT=>Output)
```

Representation in ST

Representation:

```
LDLG_Instance (IN:=Input, GAIN:=GainFactor,
    LEAD:=DifferentialTimeConstant, LAG:=LagTimeConstant,
    TR_I:=InitializationInput, TR_S:=InitializationType,
    OUT=>Output) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Description
IN	REAL	Input
GAIN	REAL	Gain factor
LEAD	TIME	Derivative time constant
LAG	TIME	Delayed time constants
TR_I	REAL	Initialization input
TR_S	BOOL	Initialization type "1" = Operating mode Tracking "0" = Automatic operating mode

Description of output parameters:

Parameter	Data type	Description
OUT	REAL	Output

Runtime error

For a list of all block error codes and values, see *Conditioning, page 358*.

Detailed description

Parametering

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

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

Operating mode

Two operating modes can be selected through the TR_S parameter input.

Operating mode	TR_S	Meaning
Automatic	0	The Function block will be handled as "Parametering" describes.
Tracking	1	The tracking value TR_I is transferred directly to the output OUT.

Examples of function block LDLG

Example overview

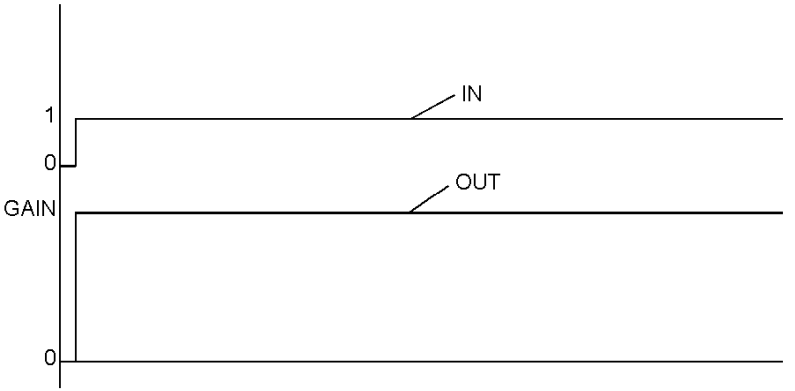
The following examples are presented in the following diagrams:

- LEAD = LAG
- LEAD/LAG = 0.5, GAIN = 1
- LEAD/LAG = 2, GAIN = 1

LEAD = LAG

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

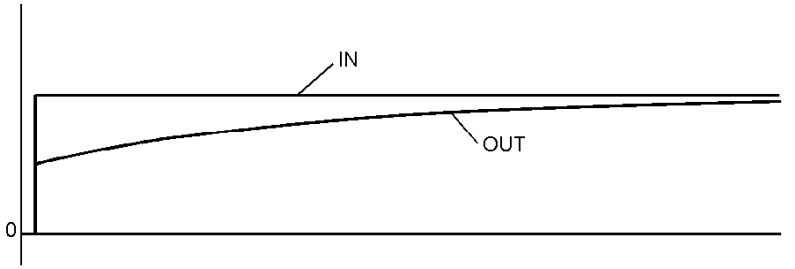
Function block LDLG with LEAD = LAG



LEAD/LAG = 0.5, GAIN = 1

In this case the output OUT will jump to half the accumulated value in order to make the transition to the final value (GAIN * IN) with the delay time constant LAG.

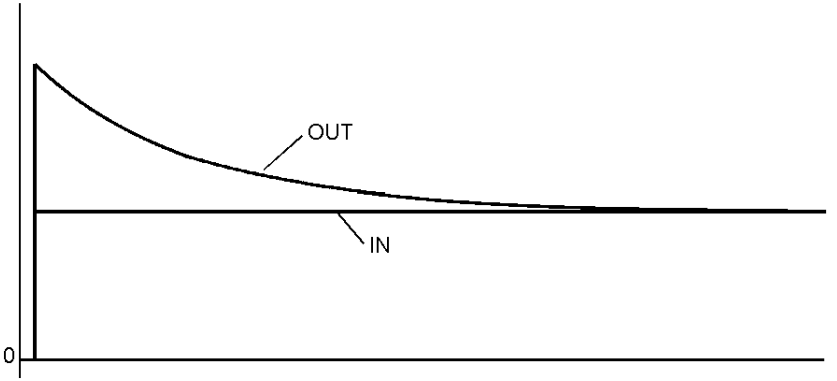
Function block LDLG with LEAD/LAG = 0.5 and GAIN = 1



LEAD/LAG = 2, GAIN = 1

In this case the output OUT will jump to double the accumulated value in order to make the transition to the final value (GAIN * IN) with the delay time constant LAG.

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



Chapter 8
LEAD: Differentiator with smoothing

Introduction

This chapter describes the LEAD block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	72
Detailed description	75

Description

Function description

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

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

NOTE: This function block performs an internal initialization in the first program cycle after a warm start or cold start (e.g. application download or power cycle) of the PLC program.
Due to this, you have to make sure that the function block is invoked in the first program cycle. In case of inkoving the function block in a later program cycle, the internal initialization will not be performed and the ouput may deliver a wrong value.

⚠ WARNING

UNEXPECTED OUTPUT BEHAVIOUR

Make sure that the function block is always invoked in the first program cycle.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

EN and ENO can be configured as additional parameters.

Formula

The transfer function for OUT is:

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

The formula of calculation is:

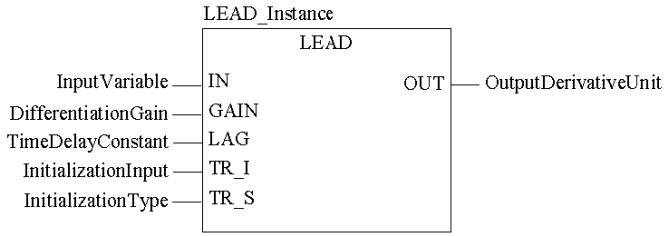
$$OUT = \frac{LAG}{dt + LAG} \times (OUT_{(old)} + GAIN \times (IN_{(new)} - IN_{(old)}))$$

Meaning of the sizes

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

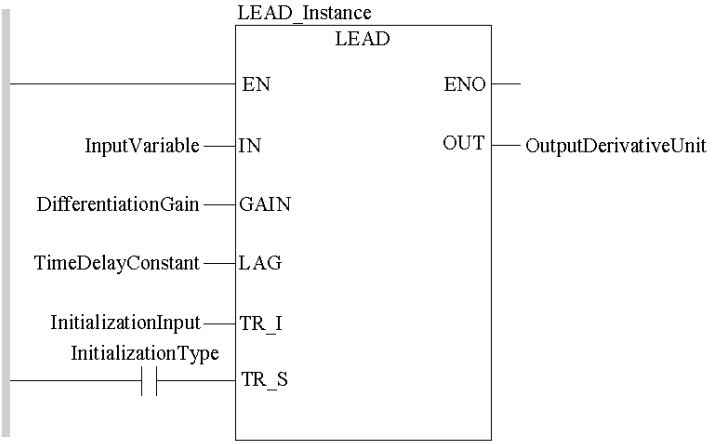
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL LEAD_Instance (IN:=InputVariable,
    GAIN:=DifferentiationGain,  LAG:=TimeDelayConstant,
    TR_I:=InitializationInput, TR_S:=InitializationType,
    OUT=>OutputDerivativeUnit)
```

Representation in ST

Representation:

```
LEAD_Instance (IN:=InputVariable,
    GAIN:=DifferentiationGain,  LAG:=TimeDelayConstant,
    TR_I:=InitializationInput, TR_S:=InitializationType,
    OUT=>OutputDerivativeUnit) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Description
IN	REAL	Input value
GAIN	REAL	Gain of the differentiation
LAG	TIME	Delayed time constants
TR_I	REAL	Initialization input
TR_S	BOOL	Initialization type "1" = Operating mode Tracking "0" = Automatic operating mode

Description of output parameters:

Parameter	Data type	Description
OUT	REAL	Output derivative unit with smoothing

Runtime error

For a list of all block error codes and values, see *Conditioning, page 358*.

Detailed description

Parametering

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

For very small sample times and the unit jump to the IN input (jump in at IN input from 0 to 1.0), the OUT output jumps to the GAIN value (theoretical value - actual slightly smaller due to the not infinitely small sample times), in order to return the LAG time constant to 0.

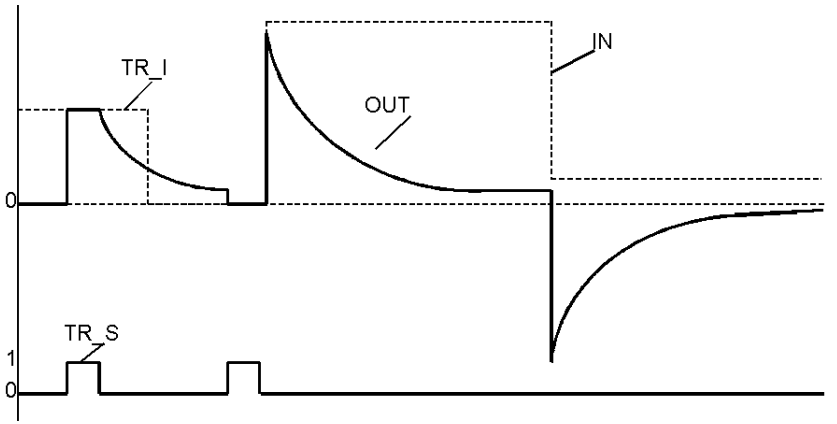
Operating mode

Two operating modes can be selected through the TR_S parameter input.

Operating mode	TR_S	Meaning
Automatic	0	The Function block will be handled as "Parametering" describes.
Tracking	1	The tracking value TR_I is transferred directly to the output OUT.

Example

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



Chapter 9
MFLOW: Mass flow block

Introduction

This chapter describes the MFLOW block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	78
Detailed description	81
Runtime error	83

Description

Function description

The function block MFLOW calculates the mass flow of a gas in a throttle device resulting from the differential pressure and the temperature and pressure conditions of the gas.

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

EN and ENO can be configured as additional parameters.

Formula

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

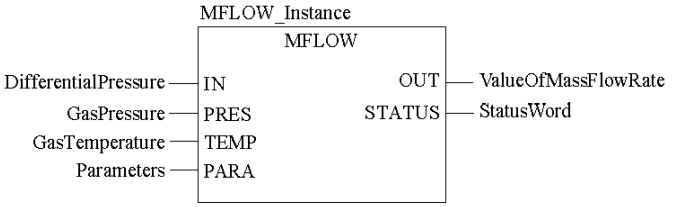
$$OUT = k \times \sqrt{\frac{IN \times PA}{TA}}$$

Meaning of the sizes

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

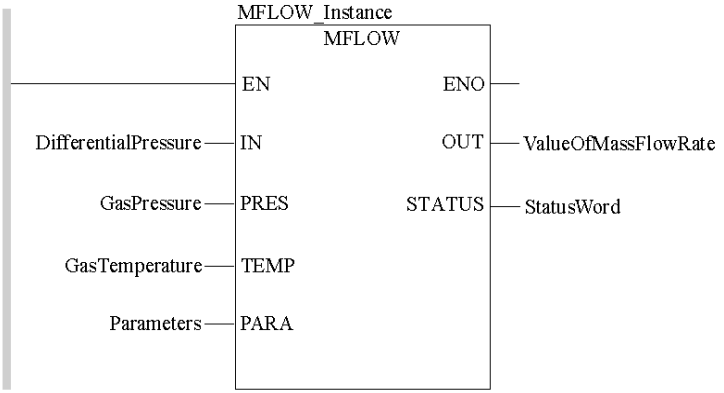
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL MFLOW_Instance (IN:=DifferentialPressure,
    PRES:=GasPressure, TEMP:=GasTemperature,
    PARA:=Parameters, OUT=>ValueOfMassFlowRate,
    STATUS=>StatusWord)
```

Representation in ST

Representation:

```
MFLOW_Instance (IN:=DifferentialPressure,
    PRES:=GasPressure, TEMP:=GasTemperature,
    PARA:=Parameters, OUT=>ValueOfMassFlowRate,
    STATUS=>StatusWord) ;
```

Parameter description MFLOW

Input parameter description:

Parameter	Data type	Differential pressure (or other measure)
IN	REAL	Input
PRES	REAL	Absolute or relative gas pressure
TEMP	REAL	Gas temperature printed out in °C or °F
PARA	Para_MFLOW	Parameter

Output parameter description:

Parameter	Data type	Differential pressure (or other measure)
OUT	REAL	Value of the mass flow, with temperature and pressure correction
STATUS	WORD	Status word

Parameter description Para_MFLOW

Data structure description

Element	Data type	Meaning
k	REAL	Calculating constants (see *Calculation of the constant k, page 81*
en_pres	BOOL	"1": Activate the pressure correction
pr_pa	BOOL	"1": PRES is an absolute pressure "0": PRES is a relative pressure
pu	REAL	Value, which in the used pressure unit 1 displays atmosphere
en_temp	BOOL	"1": Activate the temperature correction
tc_tf	BOOL	"1": TEMP will be expressed in degrees Fahrenheit "0": TEMP will be expressed in degrees Celsius
en_sqrt	BOOL	"1": Calculation with Square Root

Detailed description

Calculation of the constant k

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

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

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

When the input IN **is not a Differential pressure** the equation says as follows:

$$k = MF\_REF$$

Specification of the calculation

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

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

Temperature unit

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

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

Pressure unit

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

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

Runtime error

Status word

The bits of the status words have the following meaning:

Bit	Value in Dec.	Value in Hex.	ENO Status	Description
Bit 0 = 1	1	0x0001	False	Error in a calculation in floating point values
Bit 1 = 1	2	0x0002	False	Recording of an invalid value of a floating point value input
Bit 2 = 1	4	0x0004	False	Division by zero with calculation in floating point values
Bit 3 = 1	8	0x0008	False	Capacity overflow during floating point value calculation
Bit 4 = 1	16	0x0010	True	One of the following variables is negative: `IN`, `pu`, PA, TA. For calculation, the function block uses the value 0.

For the list of other possible floating point error codes, see *Common Floating Point Errors, page 365*.

Error message

In the following cases an error will be reported:

- An invalid value will be recorded at one of the floating point inputs
- Division by zero with calculation in floating point values
- Capacity overflow during floating point value calculation

The output `OUT` will not be altered.

NOTE: For a list of all block error codes and values, see*Conditioning, page 358*.

Warning

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

Chapter 10
QDTIME: Deadtime device

Introduction

This chapter describes the QDTIME block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	86
Detailed description	89

Description

Function description

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

The function block delays the signal IN by the deadtime T_DELAY, before it is transmitted to OUT again.

The function block has a delay puffer for 128 elements (IN values), i.e. 128 IN values can be saved during the T_DELAY time. The buffer is used in such a way that it corresponds with the operating mode.

Whether the system is started cold or warm, the value of OUT remains unchanged. The internal values are set to the value of IN.

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

The function block has both a tracking and automatic mode.

NOTE: This function block performs an internal initialization in the first program cycle after a warm start or cold start (e.g. application download or power cycle) of the PLC program.
Due to this, you have to make sure that the function block is invoked in the first program cycle. In case of inkoving the function block in a later program cycle, the internal initialization will not be performed and the ouputs may deliver wrong values.

⚠ WARNING

UNEXPECTED OUTPUT BEHAVIOUR

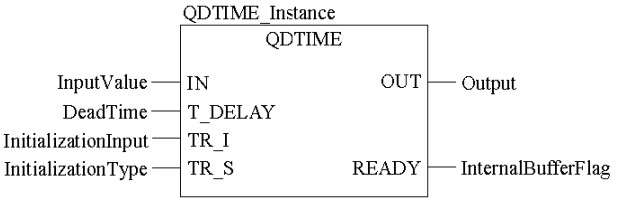
Make sure that the function block is always invoked in the first program cycle.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

EN and ENO can be configured as additional parameters.

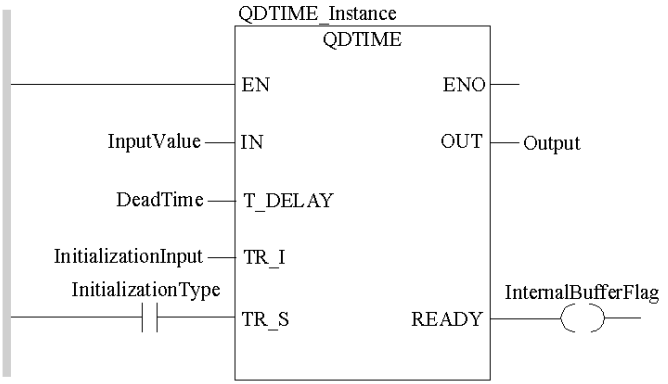
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL QDTIME_Instance (IN:=InputValue, T_DELAY:=DeadTime,
    TR_I:=InitializationInput, TR_S:=InitializationType,
    OUT=>Output, READY=>InternalBufferFlag)
```

Representation in ST

Representation:

```
QDTIME_Instance (IN:=InputValue, T_DELAY:=DeadTime,
    TR_I:=InitializationInput, TR_S:=InitializationType,
    OUT=>Output, READY=>InternalBufferFlag) ;
```

QDTIME parameter description

Description of input parameters:

Parameter	Data type	Description
IN	REAL	Input value
T_DELAY	TIME	Deadtime
TR_I	REAL	Initialization input
TR_S	BOOL	Initialization type: "1" = Operating mode Tracking "0" = Automatic operating mode

Description of output parameters:

Parameter	Data type	Description
OUT	REAL	Output
READY	BOOL	"1" = internal buffer is full "0" = internal buffer is not full (e.g. after warm/cold start or alteration to deadtime)

Runtime error

For a list of all block error codes and values, see *Conditioning, page 358*.

Detailed description

Selecting the operating modes

Two operating modes can be selected through the TR_S parameter input.

Operating mode	TR_S
Automatic	0
Tracking	1

Automatic operating mode

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

If	Then
Cycle time > T_DELAY/128	If the current IN value is transferred to the buffer, the oldest IN value will be displayed on the output OUT. In this case the solution is smaller than 128 and there is a systematic error, i.e. some IN values are saved twice (see also example).
Cycle time < T_DELAY/128	not all IN values can be contained in the buffer. In this case the IN value is not saved in some cycles and OUT remains unchanged in this cycle.

Example of cycle time > 128

The following values are accepted:

Cycle time = 100 ms

T_DELAY = 10 s

tin = T_DELAY / 128 = 78 ms

As tin (reading time) is shorter than the cycle time, every IN value is accepted in the buffer. On the fourth performance of the function block (after 400 ms) the IN value will be saved twice rather than once (because 3 x 78 = 312 and 4 x 78 = 390).

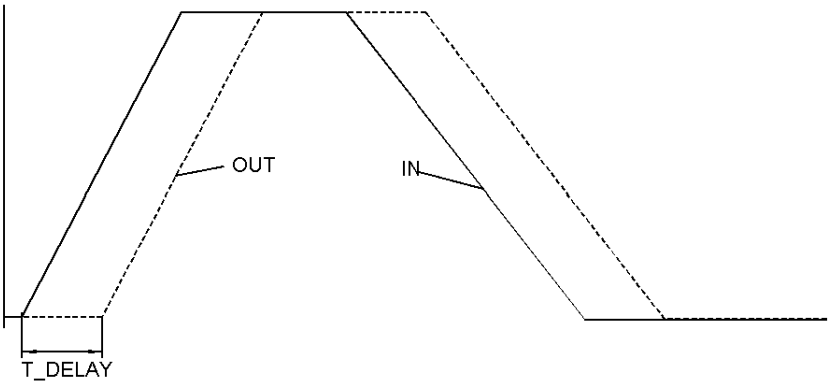
Tracking mode

In the tracking mode, the tracking value TR_I is transmitted permanently to the output OUT. The internal buffer is filled with the tracking value TR_1. The buffer is marked as charged (READY =1).

Example of the behavior of the QDTIME

The diagram shows an example of the behavior of the function block. The input IN changes, in the form of a ramp, from one value to a new value and the output OUT follows the input IN, delayed by the deadtime T_DELAY.

Diagram of the QDTIME function block.



Chapter 11
SCALING: Scaling

Introduction

This chapter describes the SCALING block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	92
Parametering	94
Runtime error	95

Description

Function description

This function block can be used to change the value range of a numerical variable.

EN and ENO can be configured as additional parameters.

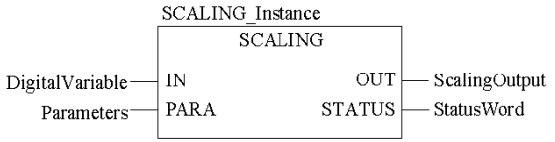
Formula

The function block carries out the following calculation:

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

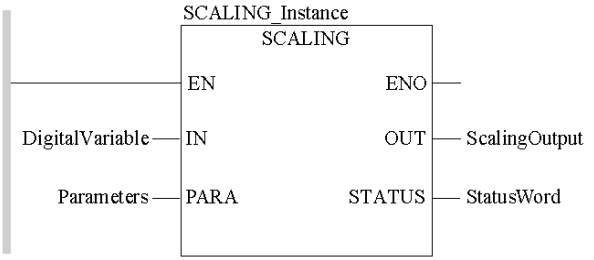
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL SCALING_Instance (IN:=DigitalVariable,
    PARA:=Parameters, OUT=>ScalingOutput,
    STATUS=>StatusWord)
```

Representation in ST

Representation:

```
SCALING_Instance (IN:=DigitalVariable, PARA:=Parameters,
    OUT=>ScalingOutput, STATUS=>StatusWord) ;
```

Parameter description SCALING

Input parameter description:

Parameter	Data type	Meaning
IN	REAL	Numerical variable to be scaled
PARA	Para_SCALING	Parameter

Output parameter description:

Parameter	Data type	Meaning
OUT	REAL	Scaled output value
STATUS	WORD	Status word

Parameter description Para_SCALING

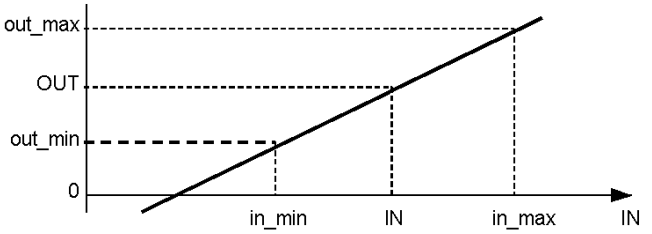
Data structure description

Element	Data type	Meaning
in_min	REAL	Lower limit of the input scale
in_max	REAL	Upper limit of the input scale
out_min	REAL	Lower limit of the output scale
out_max	REAL	Upper limit of the output scale
clip	BOOL	"1": the value of the OUT output is limited by out_min and out_max.

Parametering

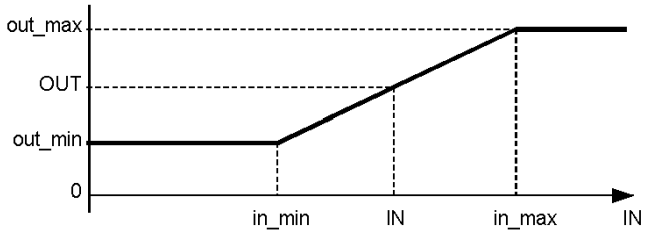
Without output limiting (clip = 0)

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



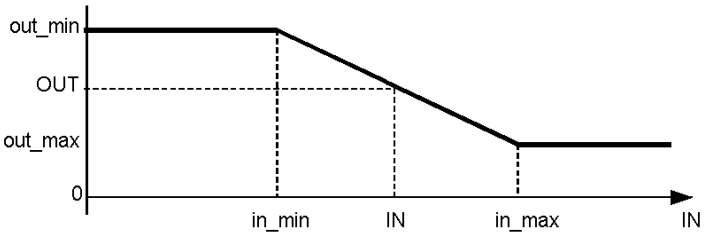
With output limiting (clip =1)

If the clip parameter is set to 1, then the scaling takes place within the range [in_min, in_max]. Outside this range, the output will be limited by the values out_min and out_max.



Modifying the rise direction

It is possible to alter the rise direction of the numerical input variables, by setting out_max to a lower value than out_min.



Runtime error

Status word

The following messages are displayed in the Status word:

Bit	Value in Dec.	Value in Hex.	ENO Status	Description
Bit 0 = 1	1	0x0001	False	Error in a floating point value calculation
Bit 1 = 1	2	0x0002	False	Recording of an invalid value on one of the floating point value inputs
Bit 2 = 1	4	0x0004	False	Division by zero during a floating point value calculation
Bit 3 = 1	8	0x0008	False	Capacity overflow for a calculation with floating point values
Bit 4 = 1	16	0x0010	True	The clip parameter is set to 1 and the input `IN` is outside this range [`in_min, in_max`]: for calculation the function block requires the values `in_min` or `in_max`.
Bit 7 = 1	128	0x0080	True	The parameter `in_min` is equal to `in_max`

For the list of other possible floating point error codes, see *Common Floating Point Errors, page 365*.

Error message

An error appears in the following cases:

- A non-floating value is on an input.
- A problem has occurred during a floating point value calculation.
- If `in_min` = `in_max`

In these cases, the `OUT` output remains unchanged.

NOTE: For a list of all block error codes and values, see*Conditioning, page 358*.

Chapter 12
TOTALIZER: Integrator

Introduction

This chapter describes the TOTALIZER block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	98
Formulas	102
Detailed description	103
Runtime error	107

Description

Function Description

This function block integrates the value of the IN input (typically a flow volume) over time, until an adjustable limit is reached (typically a volume).

EN and ENO can be configured as additional parameters.

NOTE: When using the EN enable input the following must be taken into account:
If the block has not been called for a long time because the EN enable input is set to FALSE, the totalizer block runtime is extended until the next call. If the watchdog timeout is exceeded this can lead to a PLC stop.
To remedy this, the enable input should not be used or set permanently to TRUE, so that the block is processed during every cycle.

NOTE: This function block performs an internal initialization in the first program cycle after a warm start or cold start (e.g. application download or power cycle) of the PLC program.
Due to this, you have to make sure that the function block is invoked in the first program cycle. In case of invoking the function block in a later program cycle, the internal initialization will not be performed and the outputs may deliver wrong values.

⚠ WARNING

UNEXPECTED OUTPUT BEHAVIOUR

Make sure that the function block is always invoked in the first program cycle.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

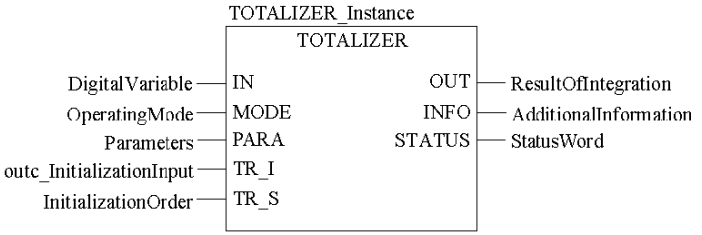
Properties

The function block has the following properties

- The integration can be temporarily paused and newly installed
- Equipment that can also consider very small input values
- Division whereby the low limit of the values of IN will no longer be considered
- Use in the mode "Reverse of the integral summation": the output OUT decreases from threshold value to zero (inc_dec = 1)

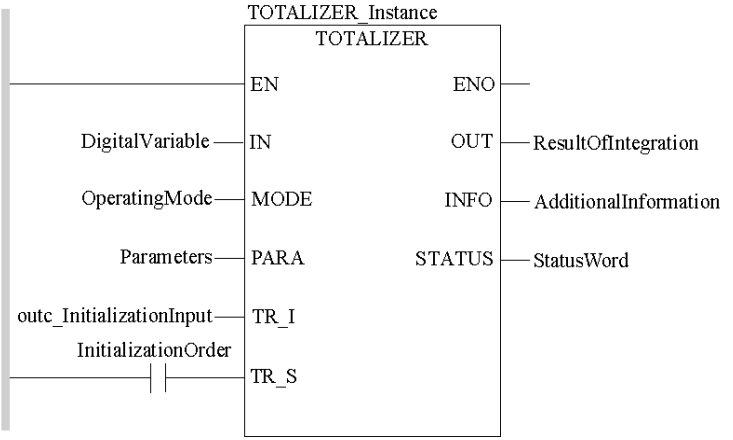
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL TOTALIZER_Instance (IN:=DigitalVariable,
    MODE:=OperatingMode, PARA:=Parameters,
    TR_I:=outc_InitializationInput,
    TR_S:=InitializationOrder, OUT=>ResultOfIntegration,
    INFO=>AdditionalInformation, STATUS=>StatusWord)
```

Representation in ST

Representation:

```
TOTALIZER_Instance (IN:=DigitalVariable,
    MODE:=OperatingMode, PARA:=Parameters,
    TR_I:=outc_InitializationInput,
    TR_S:=InitializationOrder, OUT=>ResultOfIntegration,
    INFO=>AdditionalInformation, STATUS=>StatusWord);
```

TOTALIZER Parameter Description

Description of input parameters:

Parameter	Data type	Description
IN	REAL	To integrated numerical sizes (only when > 0)
MODE	Mode_TOTALIZER	Operating mode
PARA	Para_TOTALIZER	Parameter
TR_I	REAL	Initialization input from outc
TR_S	BOOL	Initialization command

Description of output parameters:

Parameter	Data type	Description
OUT	REAL	Result of the integration of IN (limited to thld)
INFO	Info_TOTALIZER	additional information generated by function block
STATUS	WORD	Status word

Parameter Description Mode_TOTALIZER

Data structure description

Element	Data type	Description
hold	BOOL	"1": Stopping the integration
rst	BOOL	"1": Resetting the function block

Parameter Description Para_TOTALIZER

Data structure description

Element	Data type	Description
thld	REAL	Integral threshold of IN
cutoff	REAL	Division (≥ 0)
inc_dec	BOOL	"1": Reverse of integration "0": Normal mode

Parameter Description Info_TOTALIZER

Data structure description

Element	Data type	Description
outc	REAL	Total result of the integration of IN
cter	UINT	Counter for integral calculation
done	BOOL	"1": output OUT achieves integral threshold thld

Formulas

Calculating the output OUT

With each execution the output OUT is calculated with the following formula:

$$OUT(new) = OUT(old) + IN \times \Delta T$$

If OUT exceeds the threshold value thld:

- the counter cter will be incremented: $cter = cter + 1$
- the threshold value thld will be deducted from the output: $OUT = OUT \quad thld$

Explanation of formula variables

Meaning of the variables in the formulas above:

Variable	Meaning
ΔT	time elapsed since last block execution
$OUT(old)$	Value of the output OUT at the end of the previous execution of the controller

Output of the integral results

In consideration of this principle, the function block can issue three integral results:

Result	Explanation
Partial collective index OUT	indicates the integral result of input IN from the last threshold value overflow.
cter	Frequency of achieving the threshold value
Collective register (outc)	corresponds to the integral result of the input IN since the beginning of the integral invoice This counter will be updated at every execution via the following formula: $outc = thld \times cter + OUT$

Detailed description

Setting the integral threshold thld

The integral threshold value corresponds in general to a process property, which is simple to determine (e.g. the content of a tank).

The function block can also be used for the integral calculation of smaller input values, as well as when the result of the integral invoice is very large. In this case there is the risk that the integral values will become so strongly reduced in relation to the total values that they will no longer be considered. The solution offered by TOTALIZER is in the limit of the collective index OUT on the threshold value thld, so that the integral value is never insignificant in relation to the partial collective index. The result of the integral total (outc) is also calculated: the controller saves the frequency of achieving the threshold value thld on the collective index OUT.

When the threshold value thld corresponds to the value 0, the integral value will not be calculated, the outputs remain blocked.

Further properties

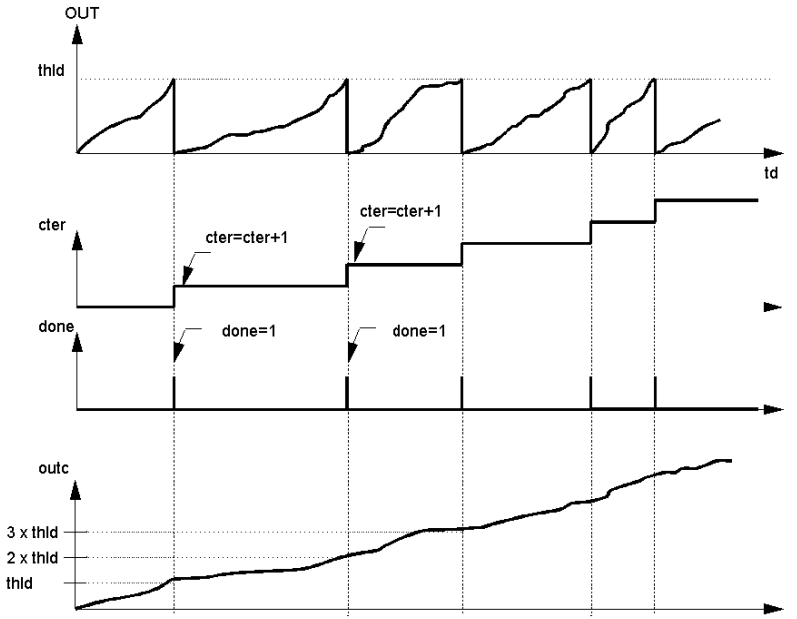
As soon as the output OUT exceeds the threshold value thld, the output done is set to 1. During the execution of the function block they are set to zero again.

When the counter cter achieves its maximum value (65535), this counter no longer changes. The outputs OUT continues to function when the threshold value thld is included. However, the output outc and the counter cter can no longer be used.

Negative values of the input IN are never considered, because they always lie below the division cutoff.

Timing diagram

Timing diagram of the TOTALIZER block



td Time span

Operating mode

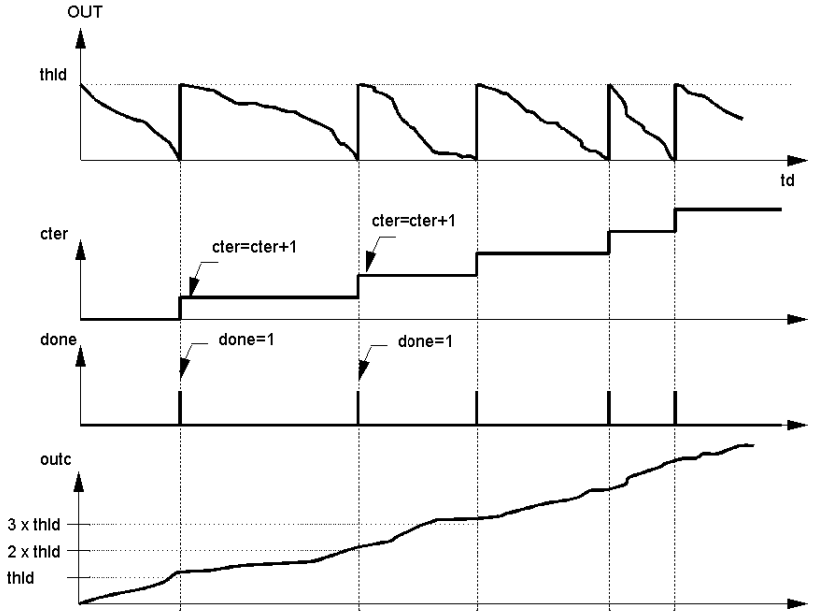
There are 3 individual operating modes for the TOTALIZER function block: Tracking, Reset and Halt:

Operating mode	Parameter	Meaning
Tracking	TR_S = 1	The parameter TR_I will be run on outc and the parameter OUT and cter will be set so that the following equation applies: outc = thld x cter + OUT. The tracking mode enables renewed synchronization of the controller outputs with the control process (e.g. as a consequence of a sensor failure).
Reset	rst = 1	The outputs OUT, outc, cter and done are set to zero. The reset via rst allows a new start from the zero reference point (for example after phase change in production).
Halt	hold = 1	Integration is paused. The outputs keep their previous values.

NOTE: By simultaneous activation of the inputs TR_S, rst and hold, the tracking mode has priority over the other operating modes and the reset operating mode has priority over halt.

Reverse integral summation (`inc_dec` = 1)

Display of the function principle



td Time span

In tracking mode (`TR_S` = 1) the parameter `TR_I` will be run on `outc` and the parameter `OUT` and `cter` will be set so that the following equation applies:

`outc` = `thld` x `cter` + (`thld` - `OUT`).

`outc` is calculated using the following formula: `outc` = `thld` x `cter` + (`thld` - `OUT`)

Function principle of the reverse of the integral summation

The following function principle applies:

Step	Action
1	At the first execution or positive on edge on `rst` the output `OUT` will be initiated by `thld`.
2	Thereafter with each execution the output `OUT` is calculated with the following formula: $OUT(new) = OUT(old) - IN \times \Delta T$
3	As soon as the output `OUT` becomes negative, the following happens: ● The counter `cter` will be incremented: `cter` = `cter` + 1 ● The threshold value `thld` will be added on to the output `OUT`: `OUT` = `OUT` + `thld` ● `done` is set to 1

Runtime error

Status word

The following messages are displayed in the status word:

Bit	Value in Dec.	Value in Hex.	ENO Status	Description
Bit 0 = 1	1	0x0001	False	Error in a floating point value calculation
Bit 1= 1	2	0x0002	False	Recording of an invalid value on one of the floating point value inputs
Bit 2= 1	4	0x0004	False	Division by zero during a floating point value calculation
Bit 3 = 1	8	0x0008	False	Capacity overflow during floating point value calculation
Bit 4 = 1	16	0x0010	True	The input `TR_I` or one of the parameters `thld` or `cutoff` are negative: For calculation, the function block uses the value 0
Bit 6 = 1	64	0x0060	True	The count register `cter` has reached its maximum value (65535): `cter` is locked at this value and the output `outc` no longer has any meaning. The `OUT` and `done` outputs can however continue to be used.

For the list of other possible floating point error codes, see *Common Floating Point Errors, page 365*.

Error message

A runtime error is signaled if a non floating point value is recorded or if there is a problem with a floating point calculation. In this case the `OUT, outc, cter` and `done` outputs remain unmodified.

NOTE: For a list of all block error codes and values, see *Conditioning, page 358*.

Warning Message

In the following cases a warning message is given:

If...	Then...
`thld` < 0	For calculation, the controller uses the value 0
`cutoff` < 0	For calculation, the controller uses the value 0
`cter` = 65535	`cter` is blocked at this value and the output `outc` no longer has any meaning. The `OUT` and `done` outputs can however continue to be used.

Chapter 13
VEL_LIM: Velocity limiter

Introduction

This chapter describes the VEL_LIM block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	110
Detailed description	113

Description

Function description

The function block creates a velocity limiter with manipulated variable limiting.

The gradient of the input variable IN is limited to a predefinable RATE value. It also limits the output OUT to within OUT_MAX and OUT_MIN. This allows the function block to adjust signals to the technologically limited pace and limits from controlling elements.

NOTE: This function block performs an internal initialization in the first program cycle after a warm start or cold start (e.g. application download or power cycle) of the PLC program.
Due to this, you have to make sure that the function block is invoked in the first program cycle. In case of inkoving the function block in a later program cycle, the internal initialization will not be performed and the ouputs may deliver wrong values.

⚠ WARNING

UNEXPECTED OUTPUT BEHAVIOUR

Make sure that the function block is always invoked in the first program cycle.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

EN and ENO can be configured as additional parameters.

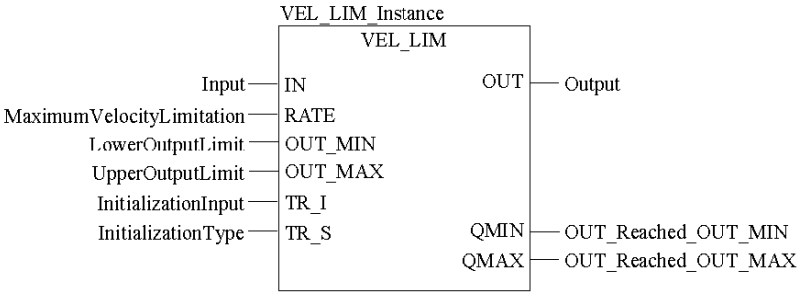
Properties

The function block has the following properties:

- Tracking and automatic operating modes
- Manipulated variable limiting in automatic mode

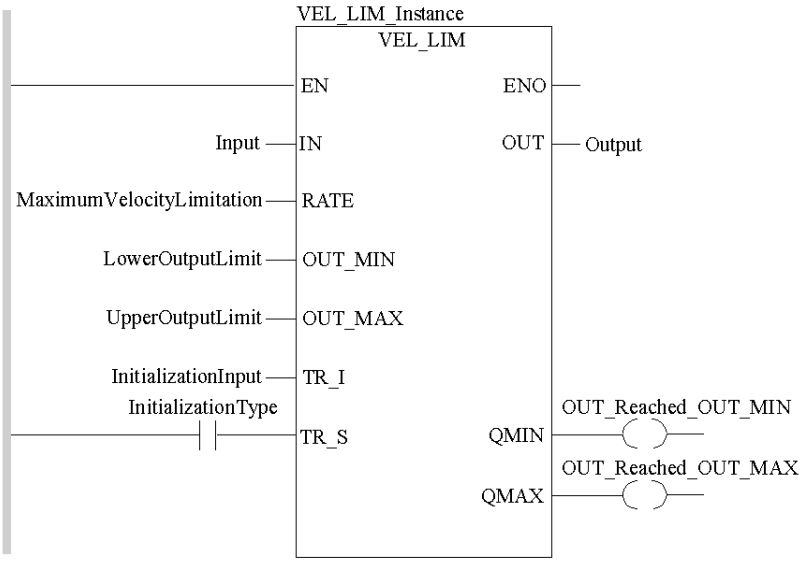
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL VEL_LIM_Instance (IN:=Input,
    RATE:=MaximumVelocityLimitation,
    OUT_MIN:=LowerOutputLimit, OUT_MAX:=UpperOutputLimit,
    TR_I:=InitializationInput, TR_S:=InitializationType,
    OUT=>Output, QMIN=>OUT_Reached_OUT_MIN,
    QMAX=>OUT_Reached_OUT_MAX)
```

Representation in ST

Representation:

```
VEL_LIM_Instance (IN:=Input,
    RATE:=MaximumVelocityLimitation,
    OUT_MIN:=LowerOutputLimit, OUT_MAX:=UpperOutputLimit,
    TR_I:=InitializationInput, TR_S:=InitializationType,
    OUT=>Output, QMIN=>OUT_Reached_OUT_MIN,
    QMAX=>OUT_Reached_OUT_MAX) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Description
IN	REAL	Input
RATE	REAL	Maximum velocity limiting
OUT_MIN	REAL	Lower limit
OUT_MAX	REAL	Upper limit
TR_I	REAL	Initialization input
TR_S	BOOL	Initialization type "1" = Operating mode tracking "0" = Automatic operating mode

Description of output parameters:

Parameter	Data type	Description
OUT	REAL	Output
QMIN	BOOL	"1" = Output OUT has reached lower limit
QMAX	BOOL	"1" = Output OUT has reached upper limit

Runtime error

With OUT_MAX < OUT_MIN an error message appears.

NOTE: For a list of all block error codes and values, see *Conditioning, page 358*.

Detailed description

Parametering

Parameter assignment for the function block is accomplished by specifying the maximum rising velocity RATE and the limiting values OUT_MAX and OUT_MIN for the output OUT. The maximum velocity rate indicates by how much the output may change within one second.

Actual RATE = 0, becomes OUT = IN.

The limits OUT_MAX and OUT_MIN limit the upper output as well as the lower output. Hence OUT_MIN ≤OUT ≤OUT_MAX.

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

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

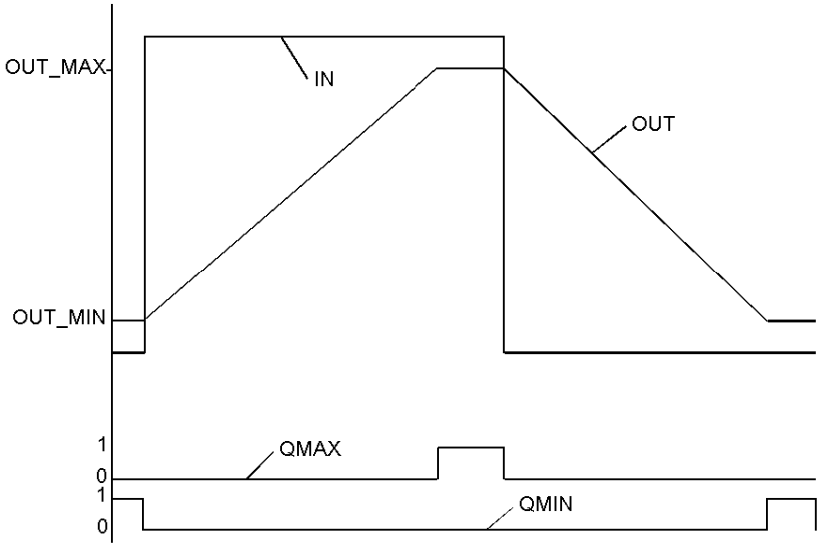
Operating mode

Two operating modes can be selected through the TR_S parameter input.

Operating mode	TR_S	Meaning
Automatic	0	The current value for OUT will be constantly calculated and displayed.
Tracking	1	The tracking value TR_I is transferred directly to the output OUT. The control output is, however, limited by OUT_MAX and OUT_MIN.

Example

Explanation of the dynamic behavior of the VEL_LIM function block.



The function block follows the transition at the input IN at its maximum velocity change rate. It can also be clearly seen that the output OUT is limited by OUT_MAX and OUT_MIN with the associated QMAX and QMIN signals.

Part III
Controller

Overview

This section describes the elementary functions and elementary function blocks of the `Controller` family.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
14	AUTOTUNE: Automatic tuner setting	117
15	IMC: Model corrector	141
16	PI_B: Simple PI controller	153
17	PIDFF: Complete PID controller	167
18	SAMPLETM: Sample time	195
19	STEP2: Two point controller	197
20	STEP3: Three point controller	205

Chapter 14
AUTOTUNE: Automatic tuner setting

Introduction

This chapter describes the AUTOTUNE block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	118
Principle of autotuning	123
Identification principle	125
Parametering	126
Controller coupling	129
Diagnosis	131
Causes of autotuning termination	133
Generating a test after stopping the autotuning	135
Runtime error	139

Description

Function description

This Function block enables the autotuning of the PID controller (*PIDFF: Complete PID controller, page 167*, *PI_B: Simple PI controller, page 153*).

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

`EN` and `ENO` can be configured as additional parameters.

Algorithm

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

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

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

The algorithm is applied to the following process types :

- Processes with only one input / output
- Processes with natural stability or integral components
- Asymmetric processes within the limits authorized by the algorithm of the PID controller
- Processes controlled via pulse width modulation output (`PWM`).

Important characteristics

The block has the following characteristics

- Pre-estimation of the control for the types `PIDFF` and/or `PI_B`
- Diagnostic function
- Parametering of the control dynamic
- Recovery of previous control settings

Operating mode

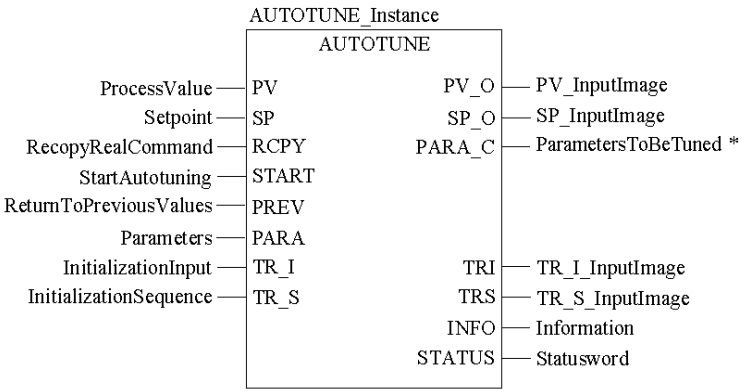
The various operating modes of the autotuning and their priorities in descending order of validity are shown in the following table:

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

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

Representation in FBD

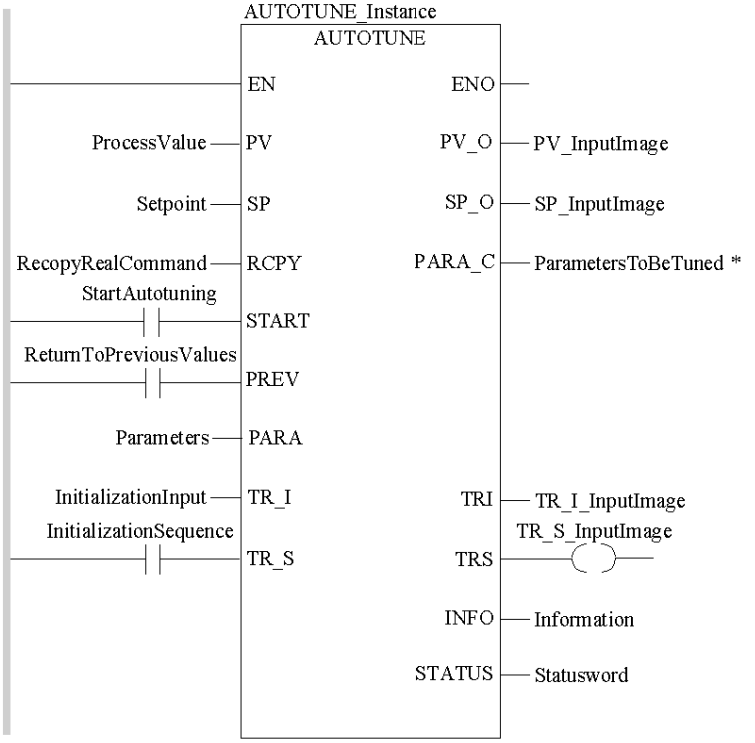
Representation:



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

Representation in LD

Representation:



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

Representation in IL

Representation:

```
CAL AUTOTUNE_Instance (PV:=ProcessValue, SP:=Setpoint,
    RCPY:=RecopyRealCommand, START:=StartAutotuning,
    PREV:=ReturnToPreviousValues, PARA:=Parameters,
    TR_I:=InitializationInput, TR_S:=InitializationSequence,
    PV_O=>PV_InputImage, SP_O=>SP_InputImage,
    PARA_C=>ParametersToBeTuned, TRI=>TR_I_InputImage,
    TRS=>TR_S_InputImage, INFO=>Information,
    STATUS=>Statusword)
```

Representation in ST

Representation:

```
AUTOTUNE_Instance (PV:=ProcessValue, SP:=Setpoint,
    RCPY:=RecopyRealCommand, START:=StartAutotuning,
    PREV:=ReturnToPreviousValues, PARA:=Parameters,
    TR_I:=InitializationInput, TR_S:=InitializationSequence,
    PV_O=>PV_InputImage, SP_O=>SP_InputImage,
    PARA_C=>ParametersToBeTuned, TRI=>TR_I_InputImage,
    TRS=>TR_S_InputImage, INFO=>Information,
    STATUS=>Statusword) ;
```

Parameter description AUTOTUNE

Input parameter description:

Parameter	Data type	Meaning
PV	REAL	Process value
SP	REAL	Setpoint
RCPY	REAL	Copy of the actual manipulated variable
START	BOOL	"0 →1" : Starting the autotune
PREV	BOOL	Reverting to the previous controller settings
PARA	Para_AUTOTUNE	Parameter
TR_I	REAL	Start input
TR_S	BOOL	Start command

Output parameter description:

Parameter	Data type	Meaning
PV_O	REAL	Copy of the actual value PV
SP_O	REAL	Copy of the SP input
PARA_C	Parameters of the autotunable controller (Para_PIDFF or Para_PI_B)	Control parameters
TRI	REAL	Copy of the TR_I input
TRS	BOOL	Copy of the TR_S input
INFO	Info_AUTOTUNE	Information
STATUS	WORD	Status word

Parameter description Para_AUTOTUNE

Data structure description

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

Info_AUTOTUNE parameter description

Data structure description

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

Principle of autotuning

Two kinds of autotuning

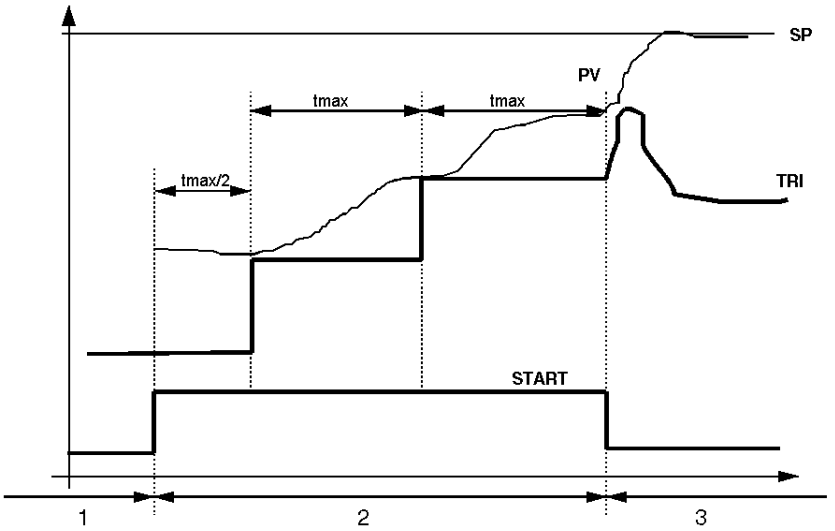
Two kinds of autotuning are possible: autotuning at a warm and cold system start

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

Autotuning at a cold start

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

Autotuning at a cold start

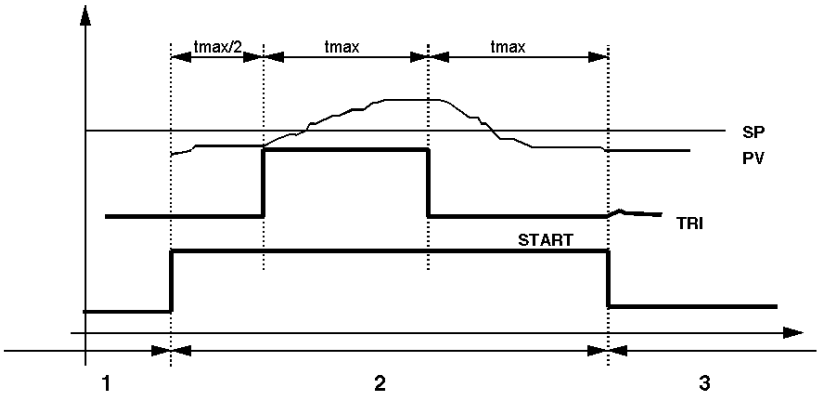


1 Automatic or manual mode
2 Autotune mode
3 Automatic or manual mode

Autotuning at a warm start

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

Autotuning at a warm start



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

Identification principle

Identification process

The identification process consists of 3 stages:

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

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

Control principle

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

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

Parametering

Parametering actuating pulse

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

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

The check occurs when autotune is started.

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

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

Performance index: perf

The controller can be modulated for each value in the performance index.

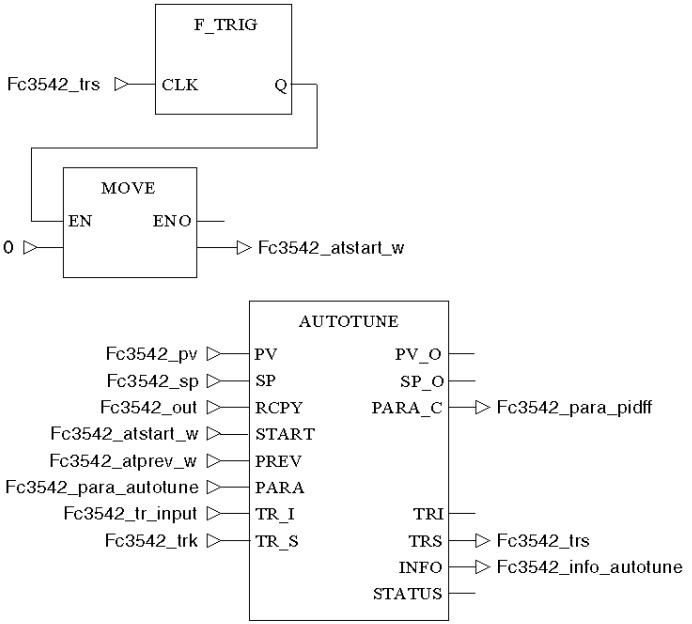
The ERROR: - An internal tag " " was added during translation.perf parameter varies between 0 and 1, which enables the perfparameter to stabilize close to 0 or perf to be set close to 1 to achieve a more dynamic setting (and therefore optimize the response time of disturbance variables).

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

Starting the autotune: START

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

Example for starting the autotuning



Reverting to the previous setting: PREV

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

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

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

The following `Info_AUTOTUNE` structural parameters are valid for `PI_B` type controllers:

Element of the data structure	Description
`p1_prev`	`KP`
`p2_prev`	`TI`

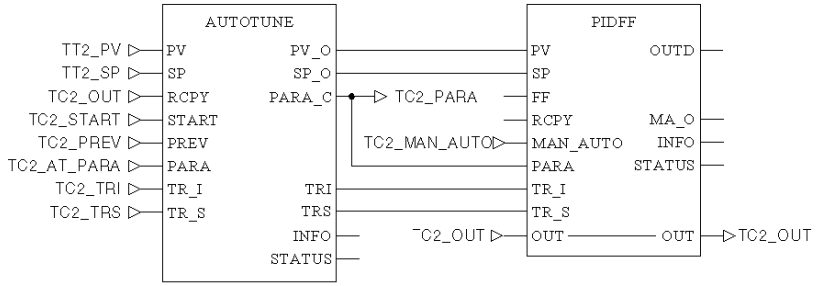
Diagnosis during autotuning: `diag`

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

Controller coupling

Application example with a PIDFF controller type EFB

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



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

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

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

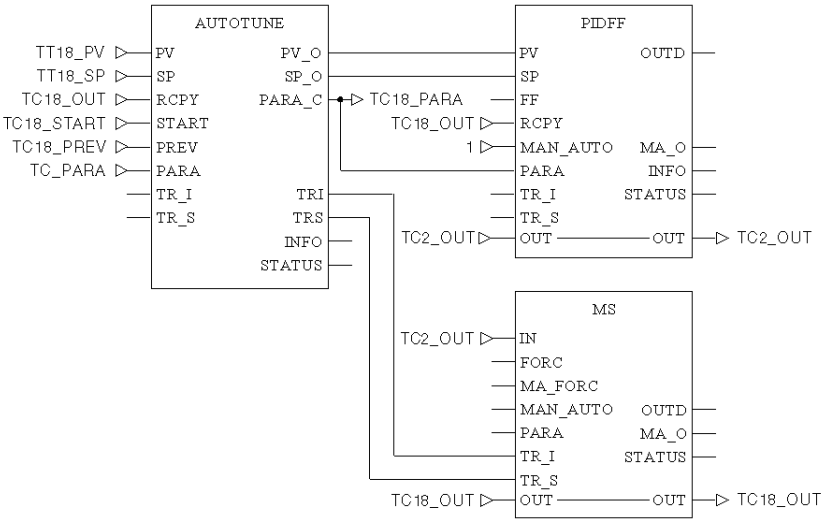
Example for connection: Servoloops with a simple PID controller

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

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

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

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



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

Diagnosis

Overview of the diagnosis

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

This word contains the current operating mode of the autotuning.

The following cases are explained:

- *Status of the autotuning,* *page 132*
- *Causes of a faulty start,* *page 132*
- *Causes of autotuning termination,* *page 133*
- *Generating a test after stopping the autotuning,* *page 135*

Diagnostic word

This table contains the meaning of the `diag` elements of the data structure `Info_AUTOTUNE`

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

Status of the autotuning

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

Bit	Meaning
0 *(see page 132)*	1 = Autotuning running.
1 *(see page 132)*	1 = Autotuning stopped

Bit 0 of the diag element

This Bit indicates that the autotuning is running. On quitting autotuning or terminating using the START Bit, this is set to zero.

Bit 1 of the diag element

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

Causes of a faulty start

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

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

Bit 2 of the diag element

The following causes can lead to a faulty start:

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

Bit 7 of the diag element

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

Causes of autotuning termination

Overview

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

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

Bit 3 of the `diag` element

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

Bit 4 of the `diag` element

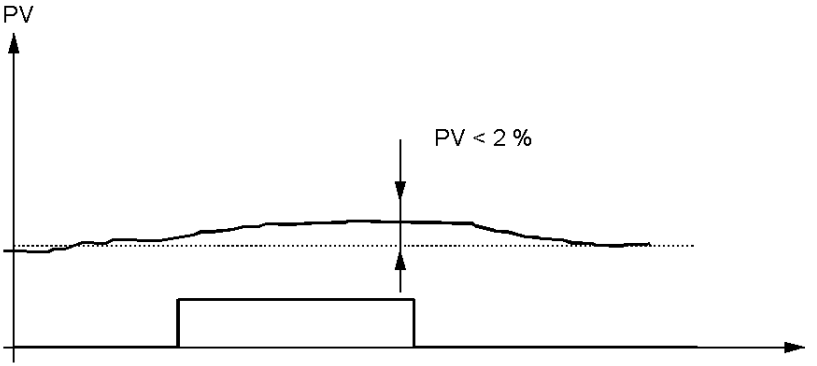
The autotuning will be cancelled if the PLC experiences a system error that prevents the completion of the chain. For example, the function will automatically stop should a voltage return occur.

Bit 5 of the `diag` element

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

Bit 6 of the `diag` element

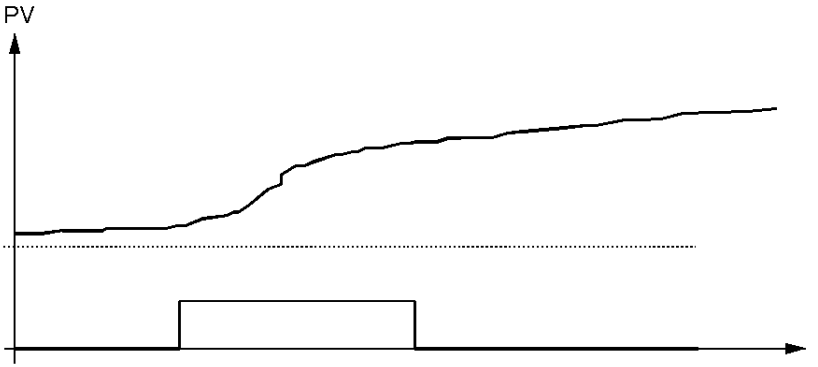
This picture shows the behavior when the ascent is too small:



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

Bit 8 of the `diag` element

This picture shows the behavior during an illogical reaction.



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

Generating a test after stopping the autotuning

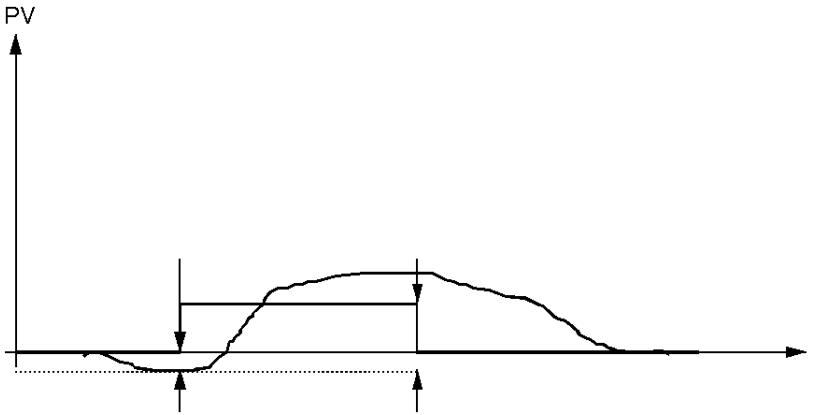
Overview

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

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

Bit 9 of the `diag` element

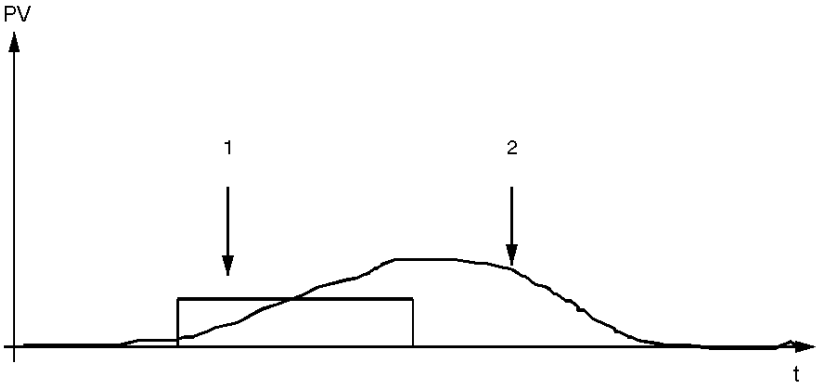
This image illustrates behavior when measurements are not initially stabilized:



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

Bit 10 of the `diag` element

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



1 Actuating pulse test
2 Process reaction

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

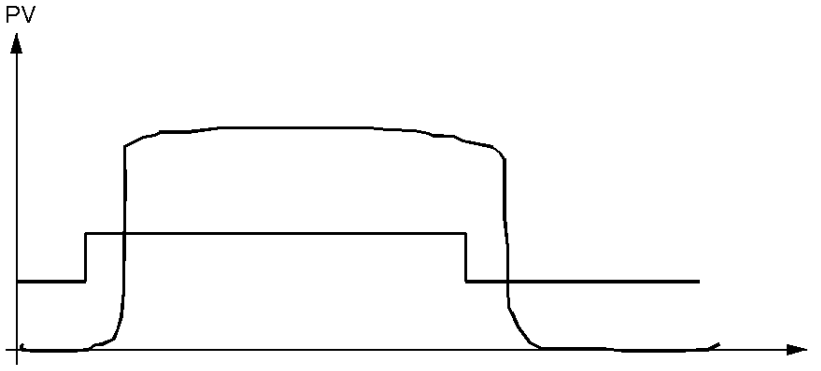
Bit 11 of the `diag` element

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

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

Bit 12 of the `diag` element

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



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

Bit 13 of the `diag` element

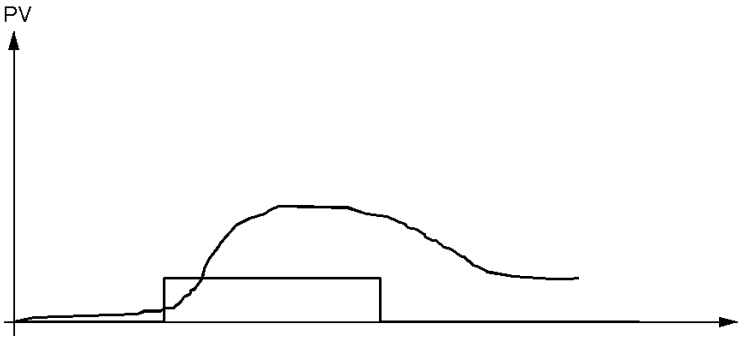
This bit is used when the reaction to an actuating pulse significantly exceeds (overshoots) the measured value (i.e. by more than 10%). The process does not conform to the models used by the algorithms.

Bit 14 of the `diag` element

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

Bit 15 of the `diag` element

This image illustrates the behavior when the process is asymmetrical.



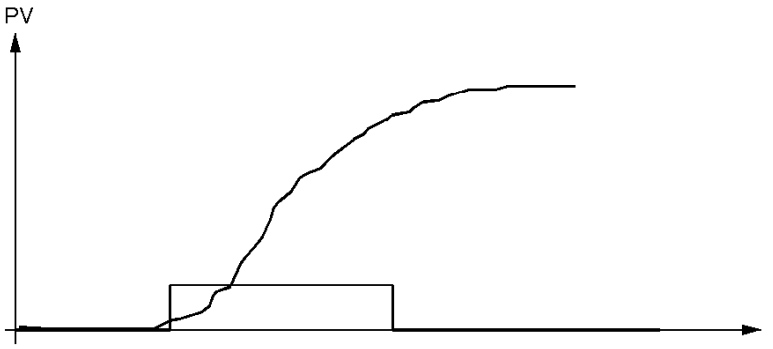
The reaction of the process is asymmetrical.

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

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

Bit 16 of the `diag` element

This image illustrates the behavior during an integration process.



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

Runtime error

Status word

The bits of the status words have the following meaning:

Bit	Description
Bit 0 = 1	Error in a floating point value calculation
Bit 1 = 1	Invalid value recorded at one of the floating point inputs
Bit 2 = 1	Division by zero with calculation in floating point values
Bit 3 = 1	Capacity overflow during floating point value calculation
Bit 4 = 1	The parameter `perf` is outside the [0.1] range: the function block uses the value 0 or 1 for calculations.
Bit 7 = 1	The thresholds `pv_inf` and `pv_sup` of the controller to be set are identical
Bit 8 = 1	The `PARA_C` output is not connected to the parameters of an autotunable controller
Bit 9 = 1	Autotuning failed
Bit 10 = 1	The last autotune was successful

For the list of other possible floating point error codes, see *Common Floating Point Errors, page 365*.

Error message

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

NOTE: For a list of all block error codes and values, see *Controller, page 359*.

Warning

A warning is issued, if the parameter `perf` is outside the [0.1] range. In this case, the block can use either the value 0 or 1 for the purpose of calculations.

Chapter 15
IMC: Model corrector

Aim of this section

This section describes the IMC function block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	142
Delay management	149
Block diagram of the IMC controller	150
Execution Error	151

Description

Description of the Function

The model corrector allows you to deal with serious delays, if any, compared with the main time constant of the process; this case cannot be satisfactorily resolved by standard PID process control. The model corrector is also important for regulating a non-linear process.

The model is first order + delay. However, this corrector may deal with any stable and aperiodic process, in any order whatsoever.

The parameters to be supplied are:

- The static gain (ratio between delta measurement/delta command in open loop).
- The pure delay value of the process (estimated value).
- The equivalent time constant (response time / 3).

The ratio between time constant in open loop/time constant in closed loop.

NOTE: This function block performs an internal initialization in the first program cycle after a warm start or cold start (e.g. application download or power cycle) of the PLC program.
Due to this, you have to make sure that the function block is invoked in the first program cycle. In case of inkoving the function block in a later program cycle, the internal initialization will not be performed and the ouputs may deliver wrong values.

⚠ WARNING

UNEXPECTED OUTPUT BEHAVIOUR

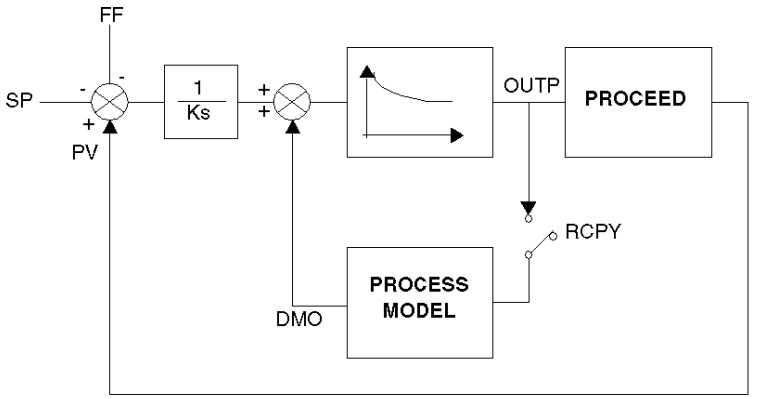
Make sure that the function block iss always invoked in the first program cycle.

For a correct behaviour you must synchronize the model corrector with the `SAMPLE_TM` function and adjust the `INTERVAL` variable to the same value as `T_ECH` of `IMC`

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Block diagram

The block diagram of the algorithm of the model corrector is:



Installation of the Corrector

The installation of a model corrector is similar to that of a PID corrector. The adjustment of the parameters KP, TI and TD of the PID having been replaced by the adjustment of the gain, the time constant, the pure delay of the process model and the ratio between the time constants in open loop and closed loop.

The model corrector uses the same inputs/outputs as a PID (PV, RSP, FF, OUTP). It also uses the RCPY optional input (external input of the model), which makes it possible to make the model's input the real process input (for example the baud rate measured at the output of a valve).

NOTE: The DMO output of the model is not directly comparable to the PV measurement. The model does not take into account at this level the Ks static gain and the possible existence of an equalizisation (BIAS).

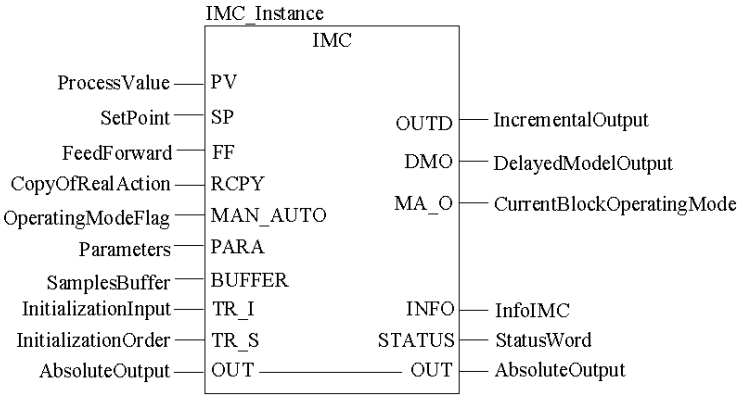
Functionalities

The functionalities other than the control calculation are identical to those of the PID:

- Direct or reverse action.
- Action Feed forward for the equalization of perturbations.
- Dead band on deviation.
- High and low output signal limiter.
- Output ramp limitation.
- High and low alarm on deviation with hysterisis.
- Selection of the automatic/manual operating mode.
- Mode Tracking.
- External output of the model.

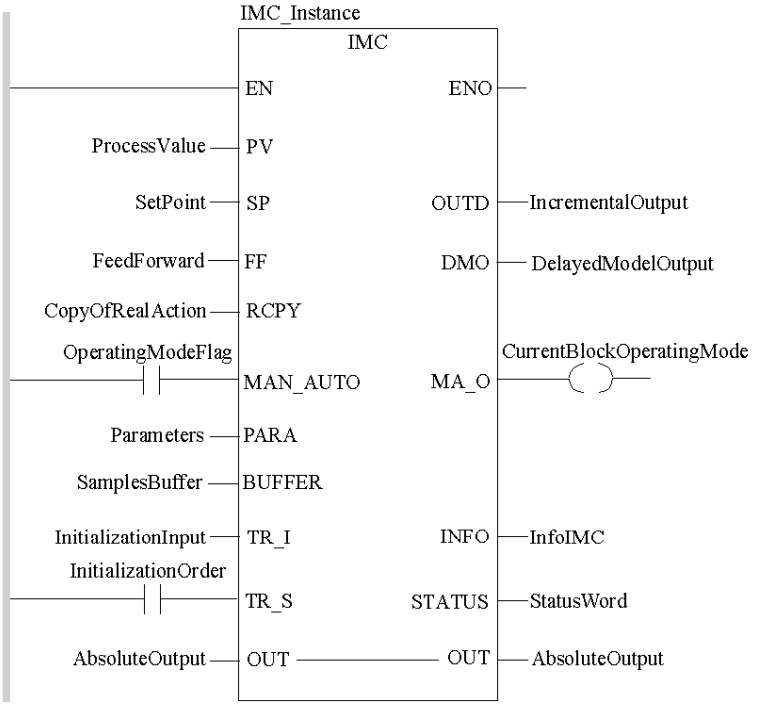
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL IMC_Instance(PV:=ProcessValue,SP:=SetPoint,
    FF:=FeedForward, RCPY:=CopyOfRealAction,
    MAN_AUTO:=OperatingModeFlag,PARA:=Parameters,
    BUFFER:=SamplesBuffer, TR_I:=InitializationInput,
    TR_S:=InitializationOrder, OUT:=AbsoluteOutput,
    OUTD=>IncrementalOutput,DMO=>DelayedModelOutput,
    MA_O=>CurrentBlockOperatingMode, INFO=>InfoIMC,
    STATUS=>StatusWord)
```

Representation in ST

Representation:

```
IMC_Instance (PV:=ProcessValue, SP:=SetPoint,
    FF:=FeedForward, RCPY:=CopyOfRealAction,
    MAN_AUTO:=OperatingModeFlag,PARA:=Parameters,
    BUFFER:=SamplesBuffer, TR_I:=InitializationInput,
    TR_S:=InitializationOrder, OUT:=AbsoluteOutput,
    OUTD=>IncrementalOutput,DMO=>DelayedModelOutput,
    MA_O=>CurrentBlockOperatingMode, INFO=>InfoIMC,
    STATUS=>StatusWord);
```

Description of the input/output parameters IMC

⚠ CAUTION

UNEXPECTED BEHAVIOR OF APPLICATION

Do not set the datatype of the PARA input parameter to constant, in the general attributes tab of the Data Properties window.

Failure to follow these instructions can result in injury or equipment damage.

Description of the input parameters:

Parameter	Type	Meaning
PV	REAL	Measurement (Process Value)
SP	REAL	Set point
FF	REAL	Feed Forward input (FeedForward)
RCPY	REAL	External input of the model
MAN_AUTO	BOOL	Operating mode of the corrector: "1": automatic mode "0": manual mode
PARA	Para_IMC	Internal parameters
BUFFER	ARRAY [n..m] OF REAL	Floating point table containing the input value to be delayed.
TR_I	REAL	Initialization input (tracking)
TR_S	BOOL	Initialization command

Description of the input/output parameters:

Parameter	Type	Meaning
OUT	REAL	Analog output of the corrector

Description of the output parameters:

Parameter	Type	Meaning
OUTD	REAL	Differential output: difference between the current cycle output and that of the previous cycle
DMO	REAL	Module output, including delay.
MA_O	BOOL	Current function block operating mode: "1": automatic mode "0": other mode (i.e. manual or tracking)
Info_IMC	REAL	Information
STATUS	WORD	Status word

Description of the internal parameters IMC

Description of the internal parameters:

Parameter	Type	Meaning
KS (1)	REAL	Static gain of the process in open loop Default value: 1.0, limits: 0.0/3E38.
OL_TIME(1)	REAL	Time constant of the process in open loop Default value: 1.0, limits: 0.0/3E38.
CL_PERF	REAL	Relationship of the natural time constants (open loop)/required (closed loop) Default value: 1.0, limits: 0.0/3E38.
T_DELAY	REAL	Current pure delay time. Default value: 0.0, limits: 0.0/3E38. Note: If this value is not a whole sampling period multiple, then it is automatically replaced by the whole sampling period multiple which is immediately below.
DBAND	REAL	Dead band around the deviation. Default value: 0, limits: 0.0/3E38.
T_ECH	REAL	Sampling period. Default value: 0,3, limits: 0.0/3E38.
REV_DIR	BOOL	Direction of action: "0": direct "1": reverse (initial value)
En_rcpy	BOOL	"1"= RCPY used (initial value = 0)
Id	UINT	Reserved for the automatic control of the controller.

Parameter	Type	Meaning
PV_INF	REAL	Low scale of PV PV (default value: 0.0)
PV_SUP	REAL	High scale of PV PV (default value: 100.0)
OUT_INF	REAL	Low scale of output OUT (default value: 0.0)
OUT_SUP	REAL	High scale of output OUT (default value: 100.0)
OUT_MIN	REAL	Low limit of output OUT (default value: 0.0)
OUT_MAX	REAL	High limit of output OUT (default value: 100.0)
OUTRATE	REAL	Velocity limit OUT (default value: 0.0)
FF_INF	REAL	Low scale of Feedforward input (default value: 0.0)
FF_SUP	REAL	High scale of Feedforward input (default value: 100.0)
OTFF_INF	REAL	Low limit of Feedforward input (default value: 0.0)
OTFF_SUP	REAL	High limit of Feedforward input (default value: 100.0)
DEV	REAL	PV-PS deviation
OUT_FF	REAL	Output value of the FF action

(1) KS and OL_TIME can not take the value 0 (inconsistent value). They will be forced to the value 1.0.

Delay management

Description

In the processes to which the controller addresses this controller, the delay is:

- Variable (For example, transfer of matter depending of the baud rate in a circuit, speed of the carrier base)
- Very great.

Principle

These two cases are dealt with using a register (buffer) of a size that may be parametrized. Depending on the size of this register, it will be possible to sample either all of the sample periods, or one period out of two, or one period out of three, etc.

It is possible to increase or reduce the delay T_DELAY during the execution of the program. The new delay is applied immediately, as long as it is compatible with the size of the register. The delay sampling period remains unchanged.

If the value T_DELAY becomes too great in relation to the size of the register, it becomes impossible to adequately store the input values to reach the required delay, if the sampling is carried out during the same period. The delay sampling period is therefore recalculated and the output is only valid after a period equal to a new delay. In order to avoid this problem, we advise you to size the register, taking into account any increases in the delay T_DELAY.

If the delay decreases, by default the sampling does not change. However, it is possible to command a new sampling calculation if necessary.

In the case of a dynamic modification of the time of the task or the sampling period, the output is only valid after a period equal to the delay.

Any dynamic modification T_DELAY between 0 s and 30 s is immediately taken into account without changing the sampling of the register.

Example

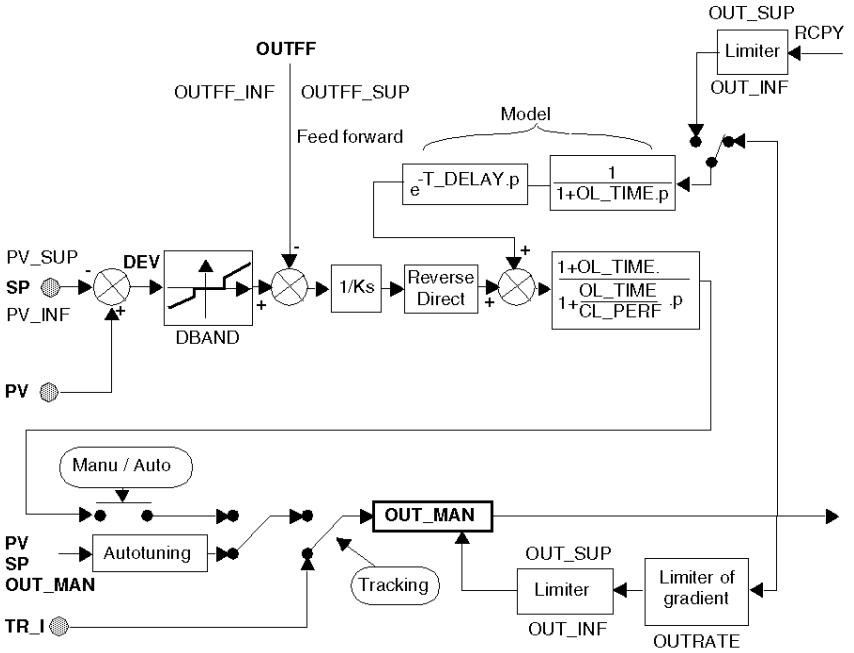
Calculation example

Sampling period	T_ECH = 300 ms
Size of the delay register	50
Delay	T_DELAY = 25 s
The delay register is therefore sampled every 2 T_ECH	50 x 2 x 0.3 = 30 s > 25 s

Block diagram of the IMC controller

Block diagram

The block diagram of the module controller is:



Execution Error

Execution Monitoring

An execution error is signaled in the following cases:

- A non-live input data is detected on one of the parameters.
- A problem appears in a floating point calculation.
- The output scale is inconsistent at the time of the cold start of the controller (OUT_INF >= OUT_SUP).

In all cases, the error is considered to be serious. The loop output is frozen and the errors are signaled in the status words.

Status Word

The status word displays the following messages:

Bit	Meaning
Bit 0 =1	Error at the time of a calculation with the floating point values
Bit 1 = 1	Non-admissible value detected on one of the floating point inputs
Bit 2 = 1	Division by 0 on a floating point calculation
Bit 4 = 1	The following behaviors are signaled: ● The SP input overflows from the zone [pv_inf, pv_sup]: the function block uses the value pv_inf or pv_sup for the calculation. ● The parameter kp or dband is negative: the block function uses the value 0 instead of the incorrect parameter value. ● The outbias parameter comes from the zone [(out_min - out_max), (out_max - out_min)]. The function block uses the value (out_inf - out_sup) or (out_sup - out_inf) for the calculation
Bit 5 = 1	The OUT output has reached the lower limit out_min
Bit 6 = 1	The OUT output has reached the upper limit out_max
Bit 7 = 1	The limit values pv_inf and pv_sup are the same
Bit 8 = 1	The pure delay buffer does not exist

NOTE: When a floating point calculation error occurs, the status output returns an error code *(see page 365)*.

Chapter 16
PI_B: Simple PI controller

Introduction

This chapter describes the `PI_B` block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	154
Formulas	159
Parametering	160
Detailed equations	163
Runtime error	165

Description

Function description

The function block PI_B depicts a PI-algorithm with a mixed structure (series/parallel). Its functions derive from function block PIDFF *(see page 167)*. These functions enable the function block to perform most classical control applications, without compromising user friendliness or using too many system resources. However, for difficult control tasks requiring extended control functions, the PIDFF block should be used.

EN and ENO can be configured as additional parameters.

Functions

The most important functions of function block PI_B are as follows:

- Calculation of the proportional and integral component in incremental form
- Actual value, setpoint value, and default value in physical units
- Direct or inverse action
- Possibility of upgrading a block-external I component (RCPY input)
- Dead zone on deviation
- Incremental value and absolute value default
- Upper and lower limit value of the default signal
- Output offset
- Selecting manual/automatic mode
- Tracking mode
- Upper and lower limit of the setpoint value

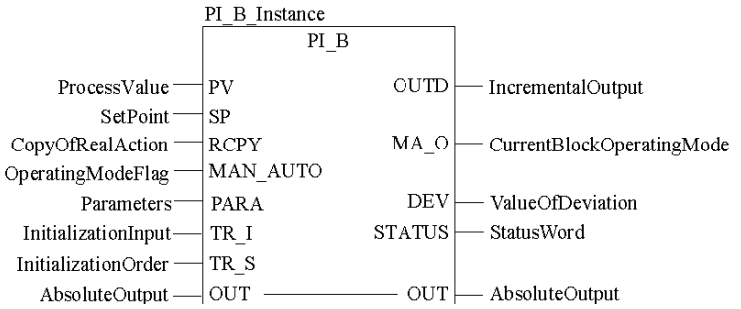
Extended functions

As is the case with PIDFF these functions can be extended by using various additional function blocks:

- Automatic control setting via the block AUTOTUNE
- Internal or external setpoint value selection via the block SP_SEL
- Controlling manual operation of the sampled servoloops *(see page 32)* using the function block MS

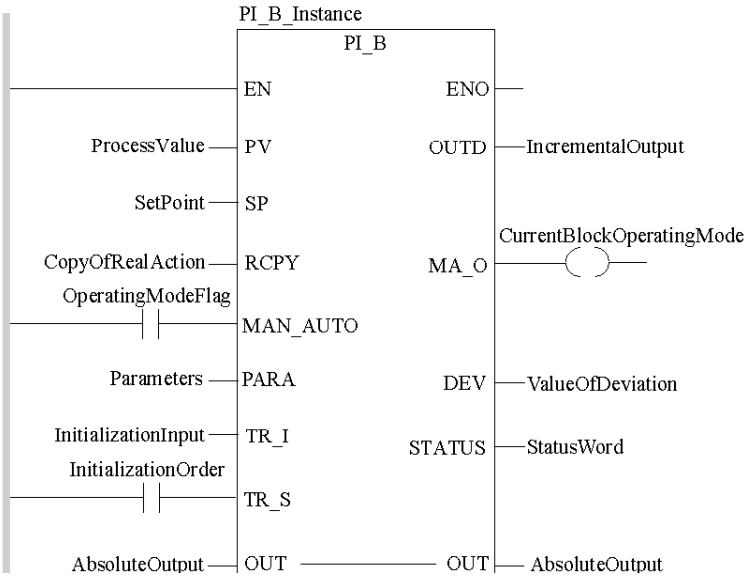
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL PI_B_Instance (PV:=ProcessValue, SP:=SetPoint,
    RCPY:=CopyOfRealAction, MAN_AUTO:=OperatingModeFlag,
    PARA:=Parameters, TR_I:=InitializationInput,
    TR_S:=InitializationOrder, OUT:=AbsoluteOutput,
    OUTD=>IncrementalOutput,
    MA_O=>CurrentBlockOperatingMode, DEV=>ValueOfDeviation,
    STATUS=>StatusWord)
```

Representation in ST

Representation:

```
PI_B_Instance (PV:=ProcessValue, SP:=SetPoint,
    RCPY:=CopyOfRealAction, MAN_AUTO:=OperatingModeFlag,
    PARA:=Parameters, TR_I:=InitializationInput,
    TR_S:=InitializationOrder, OUT:=AbsoluteOutput,
    OUTD=>IncrementalOutput,
    MA_O=>CurrentBlockOperatingMode, DEV=>ValueOfDeviation,
    STATUS=>StatusWord) ;
```

Parameter description PI_B

⚠ CAUTION

UNEXPECTED BEHAVIOR OF APPLICATION

Do not set the datatype of the PARA input parameter to constant, in the general attributes tab of the Data Properties window.

Failure to follow these instructions can result in injury or equipment damage.

Input parameter description:

Parameter	Data type	Meaning
PV	REAL	Process value
SP	REAL	Setpoint
RCPY	REAL	Copy of the effective actuator position
MAN_AUTO	BOOL	Controller operating mode: "1" : Automatic mode "0" : Manual mode
PARA	Para_PI_B	Parameter
TR_I	REAL	Initialization input
TR_S	BOOL	Initialization command

Input / output parameter description:

Parameter	Data type	Meaning
OUT	REAL	Actuator output

Output parameter description:

Parameter	Data type	Meaning
OUTD	REAL	Differential output Difference between the output of the current and previous execution
MA_O	BOOL	Current operating mode of the function block: "1": Automatic operating mode "0": other operation mode (i.e. manual or tracking mode)
DEV	REAL	Deviation value (PV - SP)
STATUS	WORD	Status word

Parameter description Para_PI_B

Data structure description

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

Formulas

Transfer function

The transfer function is:

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

Calculation formulas

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

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

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

Explanation of formula variables

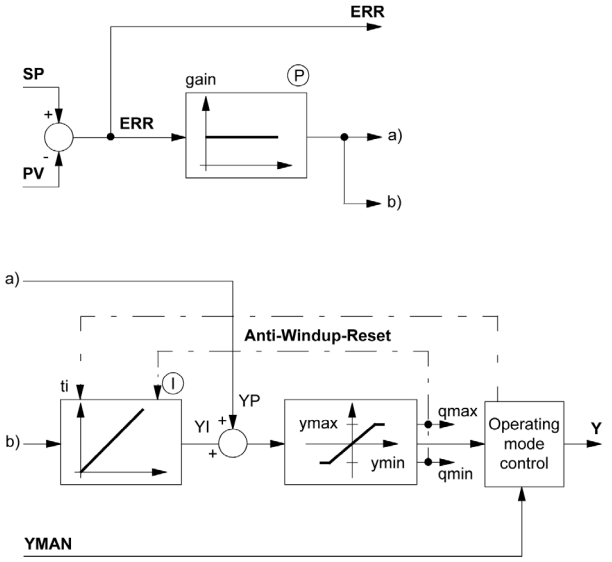
The meaning of the formula sizes is given in the following table:

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

Parametering

Structure display of PI_B controller

Structure of PI_B controller:



Absolute algorithm

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

Incremental algorithms

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

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

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

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

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

to

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

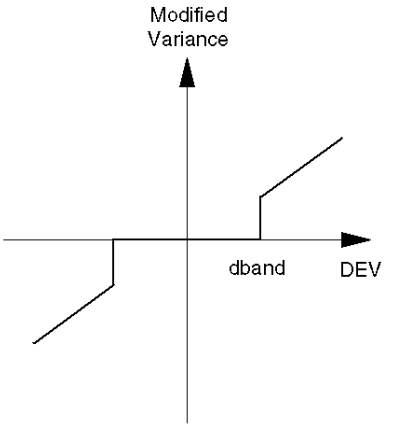
This is particularly useful for cascades or cascade-like controls.

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

Dead zone on deviation (dband)

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

Display of dead zone on deviation (dband)



Further properties

The block contains the following properties:

- The use of the parameter `outbias` allows for a precise setting of the work point when no integral component is available (`ti` = 0).
- The output `OUT` is limited to the area between `out_inf` and `out_sup` for all operation modes. If a value calculated by the function block (or a written value entered by the user in manual mode) exceeds these limits, the value of `OUT` is capped. The incremental output `OUTD`, however, does not take this cut into consideration. This enables the `PI_B` to control a `SERVO` function block without having to revert the position of the control element (continuous control).
- The choice between direct/inverse action (parameter `rev_dir`) allows for the adjustment of the control direction of the link control element/measuring process.
- Limiting the setpoint between `pv_inf` and `pv_sup`.
- The function block can operate in a purely integral mode (with `kp` = 0).

Operating mode

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

The operating modes are selected via the inputs `MAN_AUTO` and `TR_S`.

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

Switching operating modes

Manual switching →automatic or tracking →automatic is carried out as follows:

- The changeover is smooth for the incremental algorithm (`ti` > 0).
- The changeover is bumpy for the absolute algorithm (`ti`=0).

Detailed equations

Convention

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

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

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

Absolute algorithm

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

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

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

Incremental algorithm

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

$$OUTD = TermP + TermI$$

$$OUT = limiter(OUT)$$

If `en_rcpy` = 0, then:

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

If `en_rcpy` = 1, then:

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

Value of the proportional component TermP

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

Value of the integral component TermI, if `kp` > 0:

$$TermI = sense \times kp \times \frac{dt}{ti} \times DEV$$

Value of the integral component TermI if `kp` = 0 (pure integral operation):

$$TermI = sense \times \frac{out\_sup - out\_inf}{pv\_sup - pv\_inf} \times \frac{dt}{ti} \times DEV$$

Runtime error

Status word

The following messages are displayed in the status word:

Bit	Description
Bit 4 = 1	The following behavior is displayed: ● The SP input lies outside the area [pv_inf, pv_sup]: for calculation the function block requires the values pv_inf or pv_sup. ● The kp or dband parameter is negative. the function block uses the value 0 outside the incorrect parameter value. ● The outbias parameter lies outside the area [(out_inf - out_sup), (out_sup - out_inf)]. For calculation, the function block uses the value (out_inf - out_sup) or (out_sup - out_inf).
Bit 5 = 1	The output OUT has reached the lower threshold out_min (see Note)
Bit 6 = 1	The output OUT has reached the upper threshold out_max (see Note)
Bit 7 = 1	The thresholds pv_inf and pv_sup are identical.

If an error occurs during floating point processing, it will also be displayed on the Statusoutput. For the list of possible floating point error codes, see *Common Floating Point Errors, page 365*.

Note on output OUT

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

Error message

An error is displayed when a non-floating point is recorded at an input, when a problem occurs during a calculation with floating points or when the limit values pv_inf and pv_sup are identical. The outputs OUT, OUTD, MA_O and DEV remain unchanged.

NOTE: For a list of all block error codes and values, see *Controller, page 359*.

Warning Message

In the following cases a warning message is given:

- One of the kp or dband parameters are negative. the function block uses the value 0 instead of the incorrect parameter value.
- The outbias parameter lies outside the area [(out_inf - out_sup), (out_sup - out_inf)]. For calculation, the function block uses the value (out_inf - out_sup) or (out_sup - out_inf).

Chapter 17
PIDFF: Complete PID controller

Introduction

This chapter describes the PIDFF block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	168
Formulas	173
Structure diagram of the PIDFF controller	175
Parametering	177
Operating mode	181
Detailed equations	182
Detailed equations: Incremental algorithm PID controller	184
Detailed equations: Incremental algorithms in integral mode	186
Example for the PIDFF block	188
Runtime error	193

Description

Function description

The PIDFF function block is based on a PID algorithm with parallel or mixed structure (series / parallel).

EN and ENO can be configured as additional parameters.

Functions

It displays numerous functions:

- Calculating the proportional, integral and differential component in its incremental form
- 2 antiwindup measures
- Actual value, setpoint and output in physical units
- Direct or inverse action
- Differential component to process value or deviation
- Parametering the transfer gain of the differential component
- Weight of the setpoint in the proportional component (reducing the overrun)
- Possibility of upgrading a block external integral component (RCPY input)
- Feed forward component for disturbance compensation (FF input)
- Dead zone on deviation
- Incremental value and absolute value output
- Upper and lower limit on the output signal (according to operating mode)
- Gradient limitation of the output signal
- Output offset
- Selecting manual/automatic mode
- Tracking mode
- Upper and lower setpoint limit

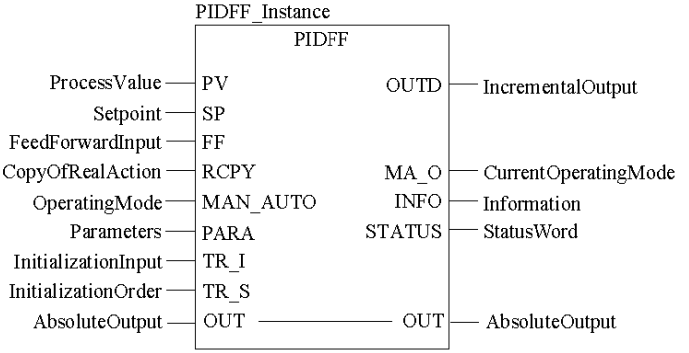
Complementary functions

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

- Autotuning using the AUTOTUNE function block
- Selecting an internal or external setpoint via the function block SP_SEL
- Controlling manual operation of the sampled control loops *(see page 32)* using the function block MS or MS_DB *(see page 285)*.

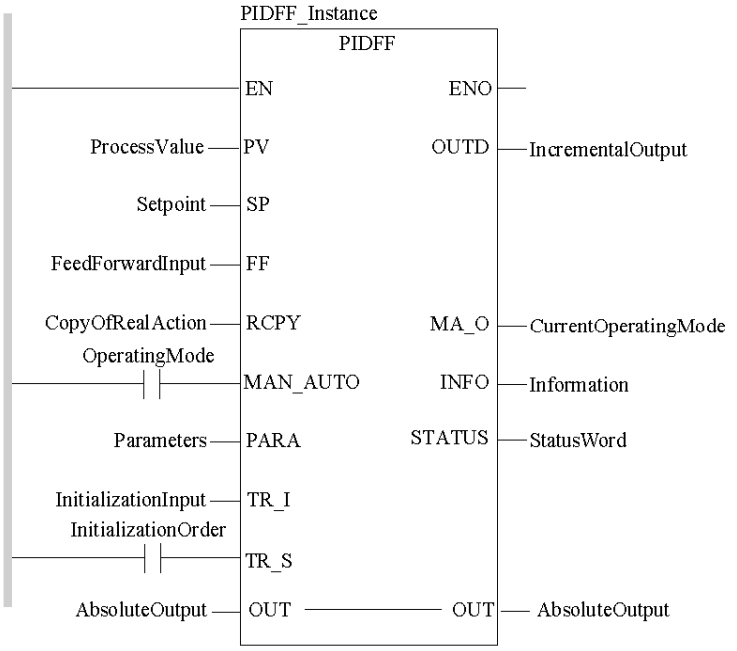
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL PIDFF_Instance (PV:=ProcessVariable, SP:=Setpoint,
    FF:=FeedForwardInput, RCPY:=CopyOfRealAction,
    MAN_AUTO:=OperatingMode, PARA:=Parameters,
    TR_I:=InitializationInput, TR_S:=InitializationOrder,
    OUT:=AbsoluteOutput, OUTD=>IncrementalOutput,
    MA_O=>CurrentOperatingMode, INFO=>Information,
    STATUS=>StatusWord)
```

Representation in ST

Representation:

```
PIDFF_Instance (PV:=ProcessVariable, SP:=Setpoint,
    FF:=FeedForwardInput, RCPY:=CopyOfRealAction,
    MAN_AUTO:=OperatingMode, PARA:=Parameters,
    TR_I:=InitializationInput, TR_S:=InitializationOrder,
    OUT:=AbsoluteOutput, OUTD=>IncrementalOutput,
    MA_O=>CurrentOperatingMode, INFO=>Information,
    STATUS=>StatusWord) ;
```

PIDFF parameter description

⚠ CAUTION

UNEXPECTED BEHAVIOR OF APPLICATION

Do not set the datatype of the PARA input parameter to constant, in the general attributes tab of the Data Properties window

Failure to follow these instructions can result in injury or equipment damage.

Description of input parameters:

Parameter	Data type	Description
PV	REAL	Process value
SP	REAL	Setpoint
FF	REAL	Disturbance input
RCPY	REAL	Copy of the current manipulated variable
MAN_AUTO	BOOL	Controller operating mode: "1": Automatic mode "0": Manual mode
PARA	Para_PIDFF	Parameter
TR_I	REAL	Initialization input
TR_S	BOOL	Initialization command

Description of input / output parameters:

Parameter	Data type	Description
OUT	REAL	Absolute value

Description of output parameters:

Parameter	Data type	Description
OUTD	REAL	Incremental value output: Difference between the output of the current and previous cycle
MA_O	BOOL	Current operating mode of the function block: "1": Automatic operating mode "0": other operating mode (i.e. manual or tracking mode)
INFO	Info_PIDFF	Information
STATUS	WORD	Status word

Parameter description Para_PIDFF

Data structure description

Element	Data type	Description
id	UINT	Reserved for autotuning
pv_inf	REAL	Lower limit of the process value range
pv_sup	REAL	Upper limit of the process value range
out_inf	REAL	Lower limit of the output value range
out_sup	REAL	Upper limit of the output value range

Element	Data type	Description
rev_dir	BOOL	"0": opposite action of the PID controller "1": direct action of the PID controller
mix_par	BOOL	"1": PID controller with parallel structure "0": PID controller with mixed structure
aw_type	BOOL	"1": Anti-windup halt is filtered
en_rcpy	BOOL	"1": the RCPY input is used
kp	REAL	Proportional action coefficient (gain)
ti	TIME	Integral time
td	TIME	Derivative time
kd	REAL	Differential gain
pv_dev	BOOL	Type of differential contribution: "1": Differential contribution in relation to system deviation "0": Differential contribution in relation to regulating variable (process value)
bump	BOOL	"1": Transition to automatic mode with bump "0": Bumpless transition to automatic mode
dband	REAL	Dead zone on deviation
gain_kp	REAL	Reducing the proportional contribution within the dead zone dband
ovs_att	REAL	Reducing the overrun
outbias	REAL	Manual compensation for the static deviation
out_min	REAL	Lower limit of the output
out_max	REAL	Upper limit of the output
outrate	REAL	Limit for output modification in units per second (≥ 0)
ff_inf	REAL	Lower limit of the FF range
ff_sup	REAL	Upper limit of the FF range
otff_inf	REAL	Lower limit of the out_ff range
otff_sup	REAL	Upper limit of the out_ff range

Parameter description Info_PIDFF

Data structure description

Element	Data type	Description
dev	REAL	Deviation value (PV - SP)
out_ff	REAL	Value of the feed forward contribution

Formulas

Transfer function

Depending on whether the mixed or parallel structure is being used, the transfer function is as follows:

Structure	Formulas
Mixed	$OUT = kp \times \left[1 + \frac{1}{ti \times p} + \frac{td \times p}{1 + \left(\frac{td}{kd}\right) \times p} \right] \times IN$
Parallel	$OUT = \left[kp + \alpha \times \frac{1}{ti \times p} + \alpha \times \frac{td \times p}{1 + \left(\frac{td}{kd}\right) \times p} \right] \times IN$
with α = scaling factor	$OUT = \frac{out\_sup - out\_\inf}{pv\_sup - pv\_\inf}$

Calculation formulas

The formulas actually used vary depending whether the function block uses the incremental or absolute form of the algorithm.

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

Algorithm	`ti`	Formulas
Absolute	0	$OUT = TermP + TermD + TermFF + outbias$ $OUTD = OUT(new) - OUT(old)$
Incremental	>0	$OUTD = TermP + TermI + TermD + TermFF$ $OUT = OUT(old) + OUTD(new)$

Explanation of formula variables

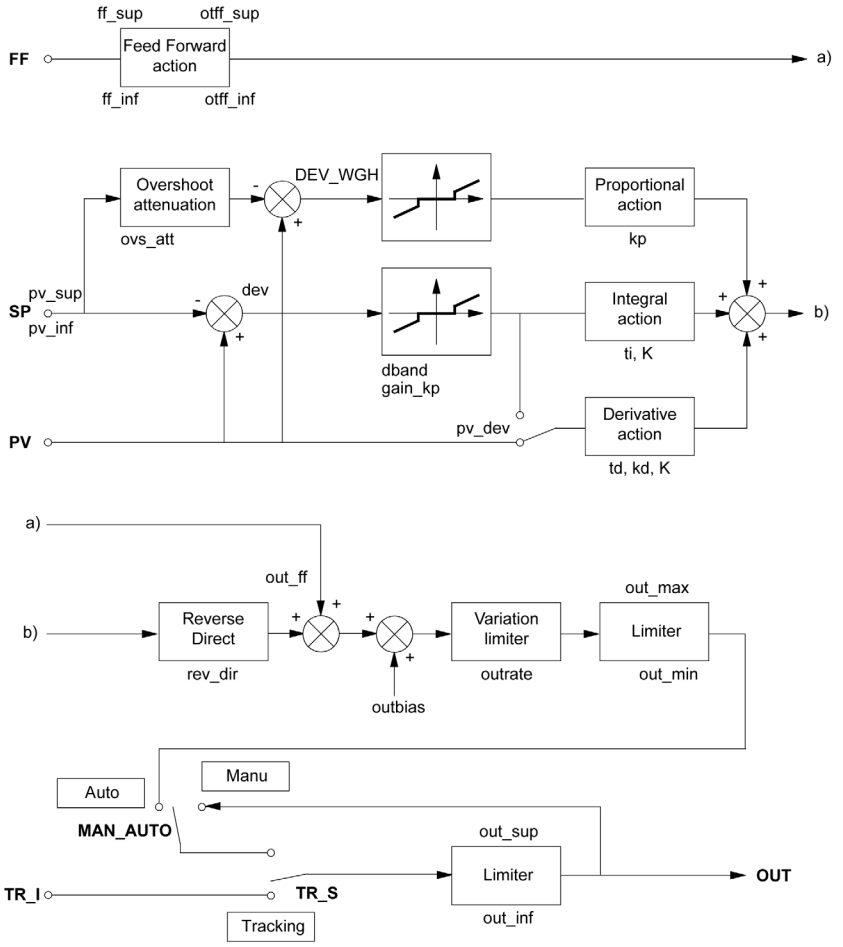
The meaning of the formula sizes is given in the following table:

Variable	Meaning
(new)	Value which is calculated on current execution of the function block
(old)	Value which is calculated on previous execution of the function block
OUT	Absolute value output
OUTD	Incremental value output
TermD	Value of the differential component
TermFF	Value of the feed forward component (disturbance compensation)
TermI	Value of the integral component
TermP	Value of the proportional component

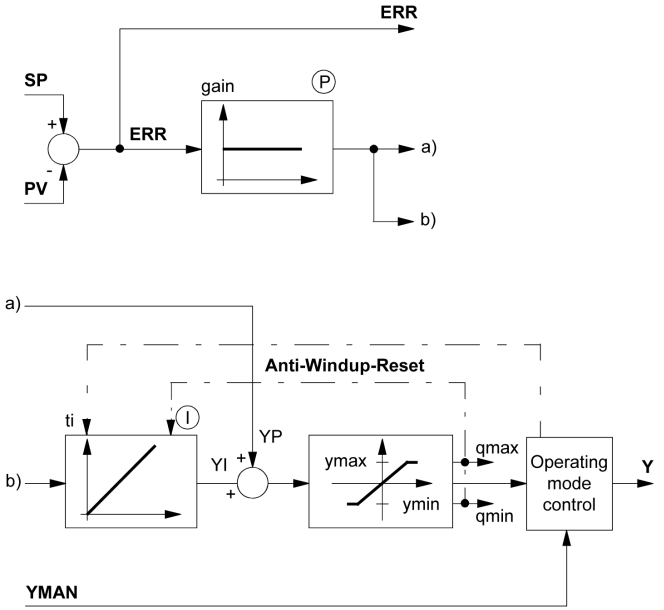
Structure diagram of the PIDFF controller

Structure diagram

Parallel structure of the PIDFF controller:



Mixed structure of the `PIDFF` controller:



Parametering

Mixed/parallel structure (mix_par)

Structure selection takes place via the mix_par parameter:

If	Then
mix_par = 0	there is a mixed structure, i.e. the proportional component is set up in the connection to the integral and differential component. The gain K set up for the components (see *Structure diagram, page 175*) corresponds to kp.
mix_par = 1	the structure is parallel, i.e. the proportional coefficient is set up parallel to the integral and differential coefficient. In this case, the gain kp does not related to the integral and differential component. In this case, gain K corresponds to the relationship between the output zone and the range.

Absolute algorithms (ti = 0)

Absolute algorithms are used when no integral component is set up (ti = 0). In this case the output OUT is calculated first, and then the output alteration is deducted.

Incremental algorithms (ti > 0)

Incremental algorithms are used when an integral component is present (i.e. when ti > 0). The special feature of this algorithm is that the output alteration OUTD is calculated first and then an absolute value output is determined according to the following formula:

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

This algorithm form makes it possible to switch a SERVO function block to the controller and thus to attain astatic control.

The incremental form also offers the following possibilities:

Possibility	Explanation
External block integral component (with en_rcpy = 1)	If the real component deviates from the value calculated by the controller (with an open servoloop), the real value should be used as the basis for the calculation. If this value is available, it should be assigned to the RCPY input and the parameter en_rcpy must be switched to 1. In calculations done by the function block, the equation OUT(new) = OUT (old) + OUTD to OUT (new) = RCPY+ OUTD This is particularly beneficial for cascades or cascade-like controls. **Note**: In this case the OUT output is not limited.

Possibility	Explanation
Expanded anti-windup measure	The incremental form of the PID controller offers as standard an anti-windup measure taken into account in the algorithm. This type is the basis when `aw_type` = 0. In this case the output can be saturated and suddenly leave its threshold, even if the sign of the deviation does not change (e.g. if it is affected by a brief disturbance during measuring). It is possible to use a second anti-windup measure (`aw_type` = 1) which prevents the output from exceeding its threshold as long as the deviation does not alter the sign.

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

If an integral component is present (`ti` > 0), the `ovs_att` parameter makes the weight of the proportional component possible the calculation of the proportional component is based on the weighted deviation ($PV \quad (1 \quad ovs\_att) \times SP$).

This could have an influence in the case of an overrun, as can occur with setpoint modifications. The aim is to retain a control-intensive proportional component and therefore a dynamic response to disturbances without an overrun occurring during control.

The parameter `ovs_att` can fluctuate continually between:

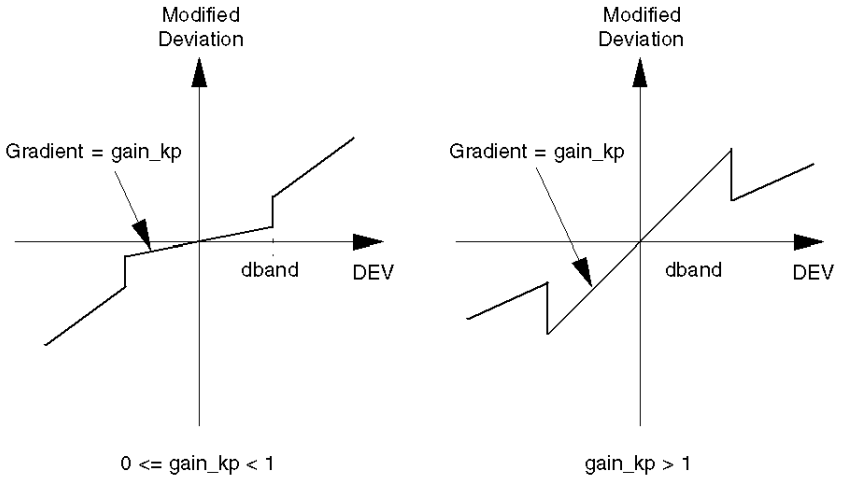
Value	Meaning
0	to the proportional component (classic case) assigned to the deviation (system deviation)
1	for the proportional component (with sensitive processes or processes with an integral effect) assigned to the measurement (controlled variable).

Dead zone on deviation (`dband`)

When the operating point is reached the dead zone can limit smaller values to the actuator's value. as long as the deviation lies below `dband`, the calculation of the function block is based on the value zero.

The extended parameter `gain_kp` can be used to modify the deviation inside the dead zone. This is better than deleting it. The modified deviation (multiplied by `gain_kp`) is used to calculate the proportional and integral components.

Representation of the alteration of the deviation



Transfer gain with the differential component

The `PIDFF` function block contains a filter of the first order for the differential component. The filter gain `kd` can be configured so that processes where the differential component must be very strongly filtered can be processed as well as processes where the filtering of the differential component can be removed because the signal is "pure" enough.

Feed forward component for disturbance compensation (`FF` input)

With classic PID control, the controller reacts to output modifications of the control process (closed servoloop). In the case of a disturbance, the controller only reacts if the process value deviates from the setpoint value. The feed-forward-function means that a measurable disturbance can be compensated for as soon as it arises. This function, conceived as an open servoloop, removes the effects of the disturbance. in this case the term disturbance size update (Feed Forward) is used.

The component of the feed forward input is updated directly/inversely to the manipulated variable of the controller after the control direction has been included.

The calculation proceeds according to the following formula:

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

A specific user example of this function is given in the section "*Application example of the Feed Forward function, page 188*".

NOTE: If `ff_sup` = `ff_inf`, the calculation of the Feed-Forward component is ignored.

Further properties

The block contains the following properties:

- The outbias parameter makes precision at the operating point possible if the process contains no integral component (ti = 0).
- In automatic mode, the OUT output is limited to the range between out_min and out_max, and to the range between out_inf and out_sup in manual mode. If a value calculated by the function block (or a written value entered by the user in manual mode) exceeds one of these limits, the value of OUT is capped. The incremental output OUT_D, however, does not take this capping into consideration. This enables the PIDFF function block to control a SERVO function block without having to revert the position of the actuator (continuous control).
- The output speed increase is limited by the parameter outrate.
- The possibility of selecting between direct/inverse action (parameter rev_dir) allows for the adjustment of the control direction of the link actuator/ process.
- The differential component can affect both the process value (pv_dev = 0), and the deviation (pv_dev = 1)
- pv_inf and pv_sup correspond to the upper and lower thresholds of the setpoint value.
- The function block can also have an effect in pure integral mode (with kp = 0).

Operating mode

Selecting the operating modes

There are 3 operating modes for the PIDFF function block: Automatic, Manual and Tracking. As the following table shows, the tracking mode takes priority over the other operating modes.

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

Operating mode	TR_S	MAN_AUTO	Meaning
Automatic	0	1	The OUT and OUTD outputs correspond to the result of the calculations made by the function block. The thresholds for the OUT are out_min and out_max.
Manual	0	0	The output OUT is not set via the function block. Its value can be directly modified by the user. OUT remains limited however; this operating mode involves the thresholds out_inf and out_sup (instead of out_min and out_max in automatic mode).
Tracking	1	0 or 1	The input TR_1 is transferred to the output OUT. As in manual mode, OUT is between the thresholds out_inf and out_sup.

Switching from Manual -> Automatic or Tracking -> Automatic

The type of changeover depends on the bump:

If	Then
bump = 0	the changeover is bumpless. **Note:** If ti = 0, the outbias parameter is recalculated. The OUT values can thus re-start beginning with the last value of the previous operating mode.
bump = 1	the changeover has a bump.

Detailed equations

Overview

The detailed equations are shown for the following situations are shown in this section:

- Convention for the most important Interim variables and Functions used in the equations
- *Absolute algorithm, page 183*
- *Incremental algorithm PID controller, page 184*
  - Normal incremental algorithms (`aw_type` = 0)
  - With bumpless anti-windup measure (`aw_type` = 1)
- *Incremental algorithms in integral mode, page 186*
  - Normal incremental algorithms (`aw_type` = 0)
  - With bumpless anti-windup measure (`aw_type` = 1)

Convention

Various variables and functions are used in the following equations. The variables corresponding to the parameters of the function block are not newly described.

The most important Interim variables and the functions used are described in the following tables.

Explanation of the interim variables

An explanation of the most important interim variables can be found here.

Interim variable	Meaning
DEV_WGH	DEV_WGH = `PV` - (1 - `ovs_att`) * `SP`
dt	Time elapsed since the last function block execution.
`K`	Gain of the integral and differential components. The gain varies according to the structure of the function block (mixed or parallel) and depends on whether the proportional component is assigned or not. ● If `mix_par` = 0 (mixed structure) and `kp` <> 0, `K` = `kp` applies ● If `mix_par` = 1 (parallel structure) or `kp` – 0, the following applies:) $K = \alpha = Skalierfaktor = \frac{out\_sup - out\_\inf}{pv\ sup - pv\ \inf}$
(new)	Value which is calculated on current execution of the function block
(old)	Value which is calculated on previous execution of the function block
OUTc	Before limitation of calculated output value
sense	Control setting
TermAW	Value of the bumpless anti-windup measure
TermD	Value of the differential component
TermFF	Value of the feed forward component (disturbance compensation)
TermI	Value of the integral component

Interim variable	Meaning
TermP	Value of the proportional component
VAR	To calculate the variable used by the differential component. Its value depends on the `pv_dev` parameter : ● If `pv_dev` = 0, VAR = `PV` ● If `pv_dev` = 1, VAR = `dev`

Explanation of the functions

An explanation of the most important functions can be found here.

Function	Meaning
Control setting	The control setting has the following directions of action: ● +1 This is an opposite action (`rev_dir` = 1) i.e. a positive deviation (`PV` - `SP`) generates an increasing output value ● -1 This is a direct action (`rev_dir` = 0) i.e. a positive deviation (`PV` - `SP`) generates a reduction in the output value
Function Δ	$\Delta(x(t)) = x(t) - x(t-1)$
'Limit'	Limiting function for the function block output

Absolute algorithm

The following equations apply for PD controllers (`ti` = 0);

$$OUT = TermP + TermD + TermFF + outbias$$
$$OUTD = OUTP(new) \quad OUTP(old)$$
$$OUT = limiter(OUT)$$

Value of the proportional component TermP

$$TermP = sense \times kp \times dev$$

Value of the differential component TermD

$$TermD = sense \times \frac{td \times TermD_{(old)} + K \times td \times kd \times (VAR_{(new)} - VAR_{(old)})}{kd \times dt + td}$$

Value of the feed forward component TermFF

$$TermFF = \frac{(FF - ff\_inf) \times (otff\_sup - otff\_\mathrm{inf})}{ff\_\mathrm{sup} - ff\_inf} + otff\_inf$$

Detailed equations: Incremental algorithm PID controller

Incremental algorithm PID controller

For the PID controller (`ti` > 0), the equations are divided into the following categories, depending on the `aw_type` element.

Element	Meaning
`aw_type` = 0	Normal incremental algorithms
`aw_type` = 1	With bumpless anti-windup measures

PID controller: `aw_type` = 0

The following equations apply to normal incremental algorithms of PID controllers;

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

$$OUT = limiter(OUT)$$

If `en_rcpy` = 0, then:

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

If `en_rcpy` = 1, then:

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

Value of the proportional component TermP:

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

Value of the integral component TermI:

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

Value of the differential component TermD

$$TermD = \Delta\left[sense \times \frac{td \times TermD_{(old)} + K \times td \times kd \times (VAR_{(new)} - VAR_{(old)})}{kd \times dt + td}\right]$$

Value of the feed forward component TermFF

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

PID controller: `aw_type` = 1

The following equations apply to incremental algorithms of PID controllers with bumpless anti-windup measures;

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

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

$$OUT = limiter(OUTc)$$

Value of the proportional component TermP:

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

Value of the integral component TermI:

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

Value of the differential component TermD

$$TermD = \Delta\left[sense \times \frac{td \times TermD_{(old)} + K \times td \times kd \times (VAR_{(new)} - VAR_{(old)})}{kd \times dt + td}\right]$$

Value of the feed forward component TermFF

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

Value of the bumpless anti-windup measure TermAW

If `en_rcpy` = 0, then:

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

If `en_rcpy` = 1, then:

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

Detailed equations: Incremental algorithms in integral mode

Incremental algorithms in integral mode

The controller can be set to a purely integral mode (`kp` = 0).

Here too, the equations are divided into the following categories, depending on the `aw_type` element:

Element	Meaning
`aw_type` = 0	Normal incremental algorithms
`aw_type` = 1	With bumpless anti-windup measures

Integral mode: `aw_type` = 0

The following equations apply to normal incremental algorithms of controllers in integral mode;

$$OUTD = TermI + TermFF$$

$$OUT = limiter(OUT)$$

If `en_rcpy` = 0, then:

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

If `en_rcpy` = 1, then:

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

Value of the integral component TermI:

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

Value of the feed forward component TermFF

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

Integral mode: `aw_type` = 1

The following equations apply to incremental algorithms of integral controllers with bumpless antiwindup measures;

$$OUTD = TermI + TermFF + TermAW$$

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

$$OUT = limiter(OUTc)$$

Value of the integral component TermI:

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

Value of the feed forward component TermFF

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

Value of the bumpless anti-windup measure TermAW

If `en_rcpy` = 0, then:

If `en_rcpy` = 1, then:

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

Example for the PIDFF block

Example overview

This chapter contains the following examples:

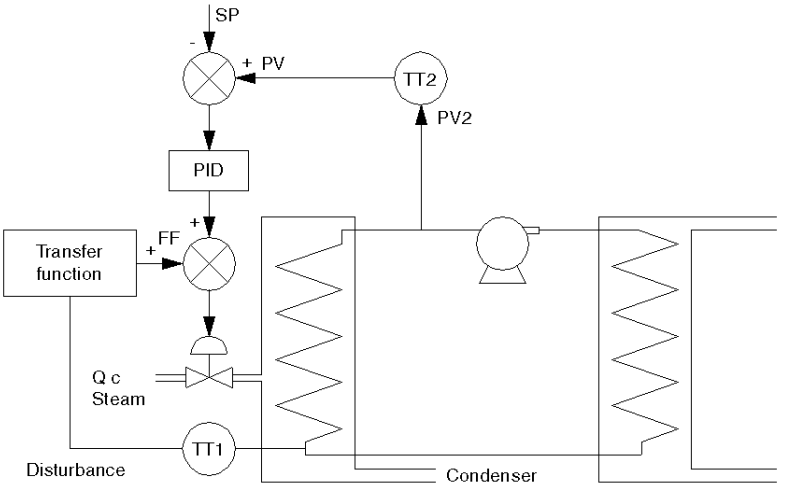
- *Application example of the Feed Forward function, page 188*
- Classic control examples programmed via the PIDFF function block:
  - *Example of the cascaded arrangement of two controllers, page 190*
  - *Example of cascade-like control, page 192*

Application example of the Feed Forward function

With a heat exchanger, the temperature PV2 should be regulated at the output of the secondary circulation. A PID controller controls the inflow valve for warm air depending on PV2 and the setpoint SP. The cold water temperature is regarded as a measurable disturbance variable in this control process.

The feed forward function means a reaction can occur as soon as the cold water temperature changes without waiting for PV2 to decrease.

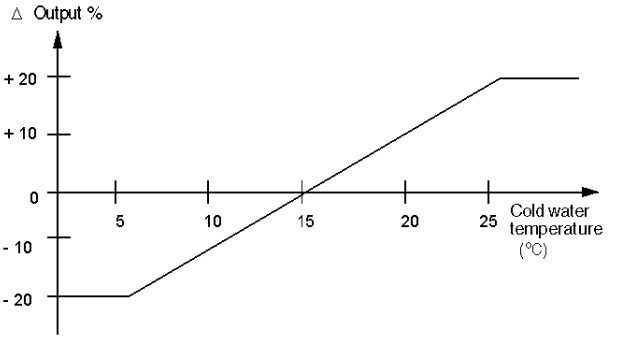
Presentation of the servo loop:



The following hypotheses are accepted:

- The condenser output temperature (cold water temperature) varies between 5 °C and 25 °C, with a mean value of 15 °C.
- A DT temperature change has a full effect on the output temperature of the heat exchanger.
- To compensate for a temperature increase (or decrease) by 5 C at the output of the heat exchanger, the steam control valve must be closed (or opened) by 10 %.

The feed forward input parameters should be adjusted so that the cold water temperature has the following effect on the steam control valve:

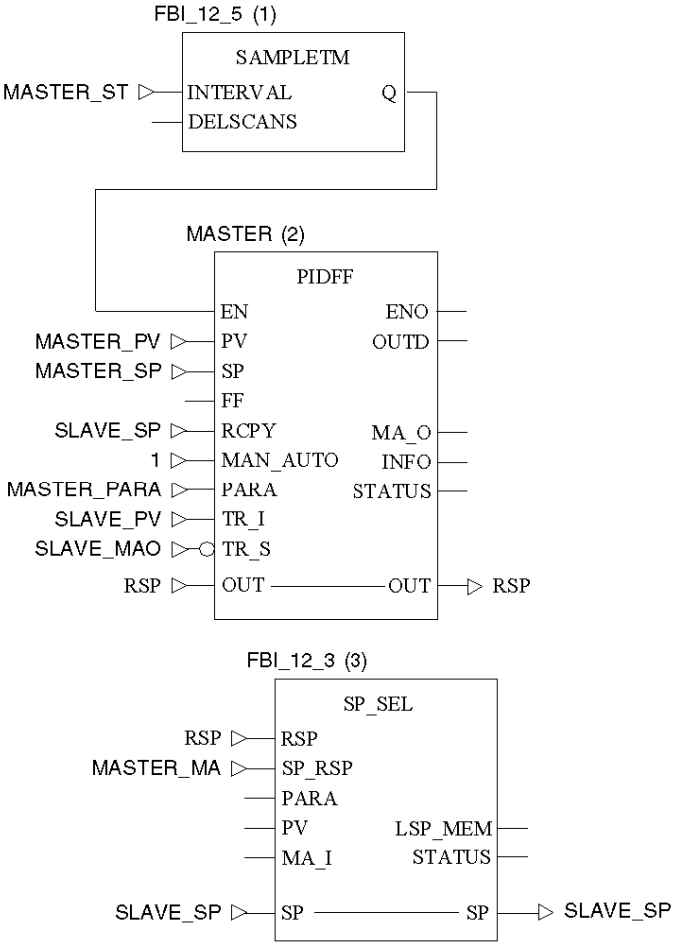
Temperature range	Effects
15 °C	no effect
10% per 5 °C between 5 and 25 °C	

Adjustments to be preset

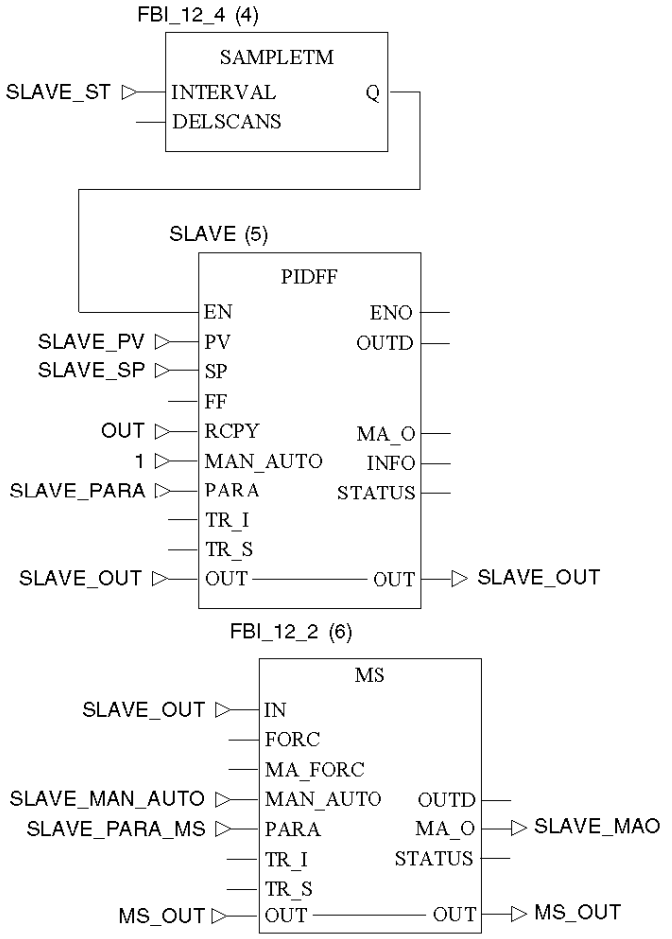
Element	Value
`ff_sup`	25 °C
`ff_inf`	5 °C
`otff_sup`	10 %
`otff_inf`	- 10 %

Example of the cascaded arrangement of two controllers

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

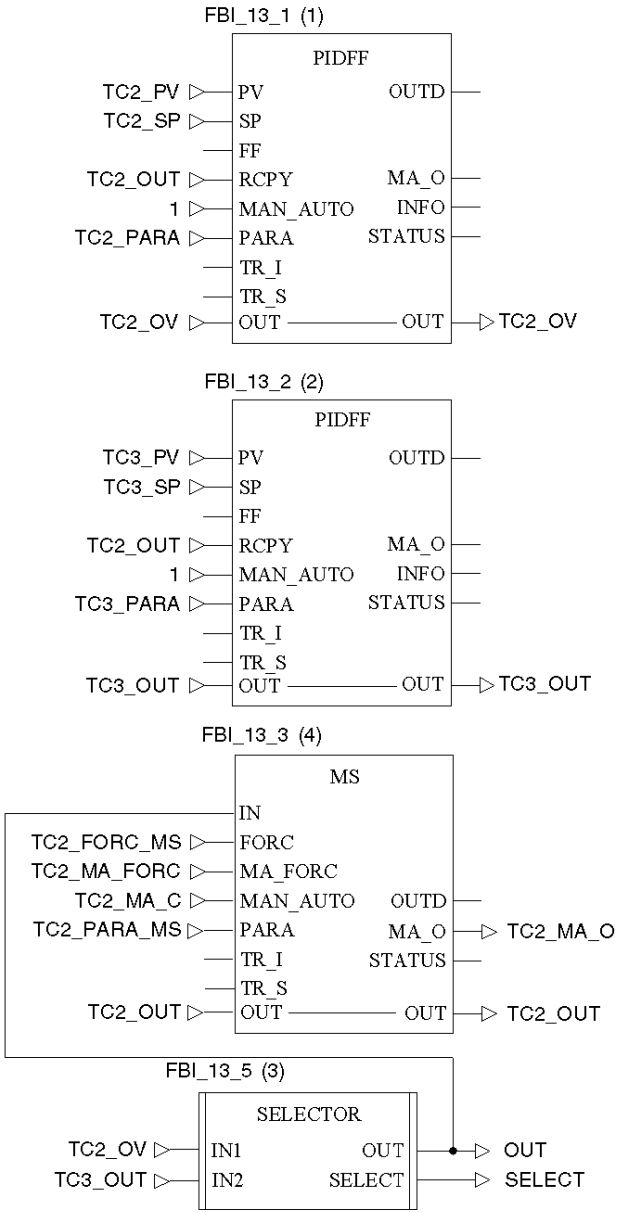


A representation of the function map, part 2, follows:



Example of cascade-like control

A representation of the function map follows:



Runtime error

Status word

The following messages are displayed in the status word:

Bit	Description
Bit 4 = 1	The following behavior is displayed: ● The SP input lies outside the area [pv_inf, pv_sup]: for calculation the function block requires the values pv_inf or pv_sup. ● One of the kp, dband, gain _kp parameters outrate is negative. the function block uses the value 0 outside the incorrect parameter value. ● kd < 1 (with td <> 0) : the function block uses the value 1 instead of the faulty value of kd. ● The parameter ovs_att is outside the [0.1] range: for calculation, the function block uses the value 0 or 1. ● One of the parameters out_min or out_max is outside the range [out_inf, out_sup]. For calculation, the function block uses the value out_inf or out_sup. ● One of the parameters outbias, otff_inf or otff_sup is outside the range [(out_min - out_max), (out_max - out_min)]. For calculation, the function block uses the value (out_min - out_max) or (out_max - out_min).
Bit 5 = 1	The output OUT has reached the lower threshold out_min (see Note)
Bit 6 = 1	The output OUT has reached the upper threshold out_max (see Note)
Bit 7 = 1	The thresholds pv_inf and pv_sup are identical.

If an error occurs during floating point processing, it will also be displayed on the Statusoutput. For the list of possible floating point error codes, see *Common Floating Point Errors, page 365*.

Note on output OUT

NOTE: In manual mode these bits stay at 1 for only one program cycle. When the user enters a value for OUT which exceeds one of the thresholds, the function block sets the Bit 5 or 6 to 1 and blocks them from the user entered value. During the next execution of the function block, the value of OUT no longer lies outside the range and bits 5 and 6 are set to zero again.

Error message

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

NOTE: For a list of all block error codes and values, see *Controller, page 359*.

Warning Message

In the following cases a warning message is given:

- One of the `kp, dband, gain _kp parameters outrate` is negative. The function block then uses the value 0 instead of the incorrect parameter value.
- `kd` < 1 (with `td` <> 0): the function block uses the value 1 instead of the faulty value of `kd`.
- The parameter `ovs_att` is outside the [0.1] range: for calculation, the function block uses the value 0 or 1.
- The parameter `out_min` or `out_max` is outside the range [`out_inf, out_sup`]. For calculations, the function block uses the value `out_inf` or `out_sup`.
- One of the parameters `outbias, otff_inf` or `otff_sup` is outside the range [(`out_min - out_max`), (`out_max - out_min`)]. For calculation, the function block uses the value (`out_min - out_max`) or (`out_max - out_min`).

Chapter 18
SAMPLETM: Sample time

Description

Function description

With this function block the function blocks of the control mechanism are released under time control.

To control, the Q output of the SAMPLETM function block is connected with the EN input of the function block to be controlled.

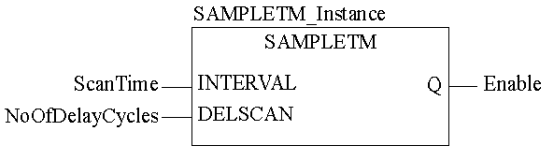
The Q output is activated for one program cycle after the stated time at the INTERVAL input has expired.

The DELSCAN input was created to prevent the simultaneous start of more than one sample time dependent FFB which are controlled by various SAMPLETM function blocks. At this input the number of cycles is stated according to which the activation of Q after a cold start is delayed. Therefore it is possible to release sample time dependent function blocks step by step to reduce the load on the CPU during the start cycle.

EN and ENO can be configured as additional parameters.

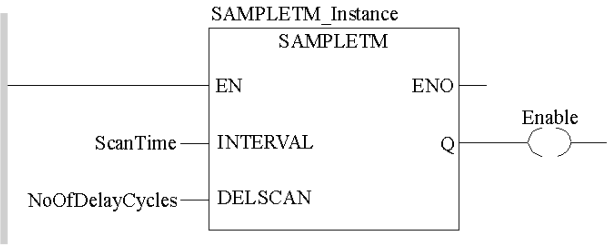
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL SAMPLETM_Instance (INTERVAL:=ScanTime,
    DELSCAN:=NoOfDelayCycles, Q=>Enable)
```

Representation in ST

Representation:

```
SAMPLETM_Instance (INTERVAL:=ScanTime,
    DELSCAN:=NoOfDelayCycles, Q=>Enable) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Description
INTERVAL	TIME	Sample time for connected control mechanism function block
DELSCAN	INT	Number of delay cycles after a cold start

Description of output parameters:

Parameter	Data type	Description
Q	BOOL	Release of control mechanism function block

Runtime error

For a list of all block error codes and values, see *Controller, page 359*.

Chapter 19
STEP2: Two point controller

Introduction

This chapter describes the STEP2 block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	198
Detailed description	201
Runtime error	203

Description

Function description

This function block is suitable for simple two point controls.

Control of the actuator proceeds according to the direction of the actual/setpoint value deviation in relation to the upper and lower threshold.

`EN` and `ENO` can be configured as additional parameters.

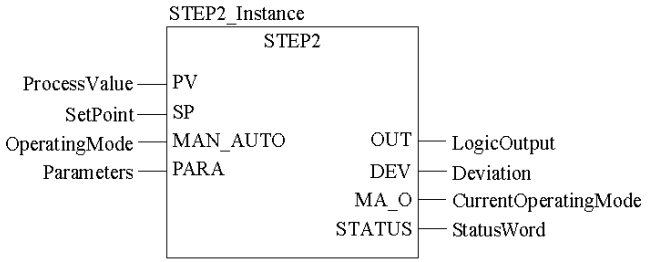
Properties

The control block has the following properties:

- Upper and lower limiting of the setpoint value between `pv_inf` and `pv_sup`
- The control input values (actual value, setpoint and associated parameters) are expressed in physical units.

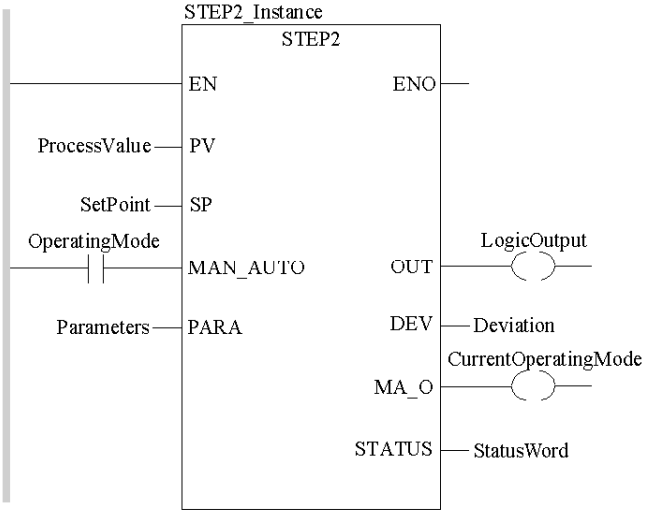
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL STEP2_Instance (PV:=ProcessValue, SP:=SetPoint,
    MAN_AUTO:=OperatingMode, PARA:=Parameters,
    OUT=>LogicOutput, DEV=>Deviation,
    MA_O=>CurrentOperatingMode, STATUS=>StatusWord)
```

Representation in ST

Representation:

```
STEP2_Instance (PV:=ProcessValue, SP:=SetPoint,
    MAN_AUTO:=OperatingMode, PARA:=Parameters,
    OUT=>LogicOutput, DEV=>Deviation,
    MA_O=>CurrentOperatingMode, STATUS=>StatusWord) ;
```

Parameter description STEP2

Input parameter description:

Parameter	Data type	Meaning
PV	REAL	Process value
SP	REAL	Setpoint
MAN_AUTO	BOOL	Controller operating mode: "1" : Automatic mode "0" : Halt mode
PARA	Para_STEP2	Parameter

Output parameter description:

Parameter	Data type	Meaning
OUT	BOOL	Logical output
DEV	REAL	Deviation (PV -SP)
MA_O	BOOL	Current operating mode of the function block (0: Halt, 1: Automatic)
STATUS	WORD	Status word

Parameter description Para_STEP2

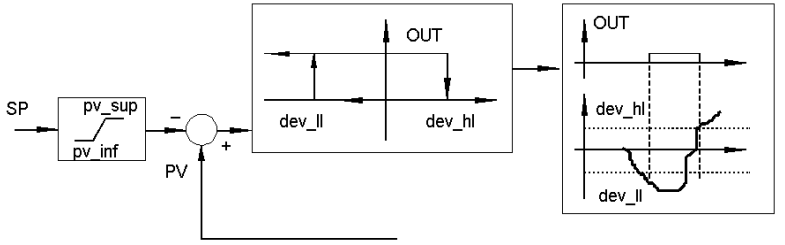
Data structure description

Element	Data type	Meaning
dev_ll	REAL	Lower deviation threshold (≤ 0)
dev_hl	REAL	Upper deviation threshold (≤ 0)
pv_inf	REAL	Lower limit of the process value range
pv_sup	REAL	Upper limit of the process value range

Detailed description

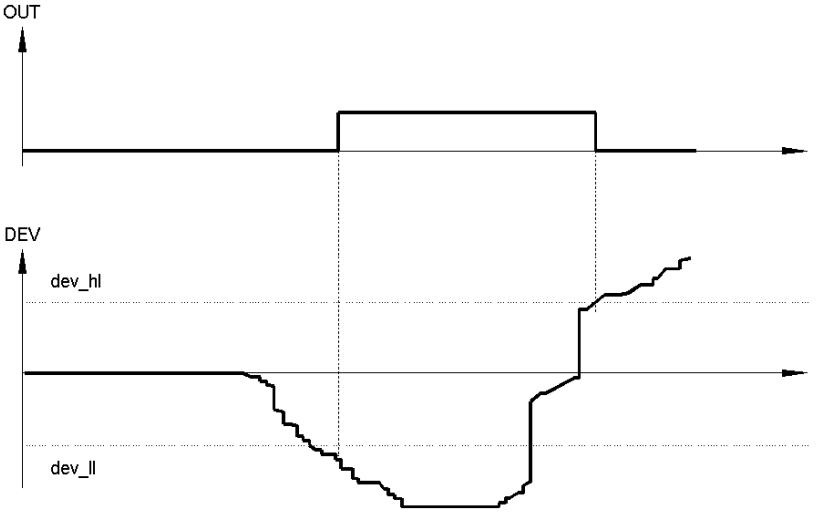
Structure diagram

The following is a structure diagram of the STEP2 block:



Behavior of the output

Behavior of the output OUT:



If the deviation (DEV = PV - SP) is less than the lower threshold dev_ll, the configured output OUT is set to 1. If however the deviation increases again, the output OUT is only set to zero if it exceeds dev_hl.

NOTE: To ensure that the block functions without errors, the output OUT should not be inverted.

Operating mode

The STEP2 function block has 2 operating modes available according to the value of the MAN_AUTO parameter :

Operating mode	MAN_AUTO	Meaning
Automatic	1	The output OUT is calculated by the controller block itself.
Halt	0	The output OUT will be held at the last calculated value.

Runtime error

Status word

The following messages are displayed in the status word:

Bit	Value in Dec.	Value in Hex.	ENO Status	Description
Bit 0 = 1	1	0x0001	False	Error in a floating point value calculation
Bit 1 = 1	2	0x0002	False	Recording of an invalid value on one of the floating point value inputs
Bit 2 = 1	4	0x0004	False	Division by zero during a floating point value calculation
Bit 3 = 1	8	0x0008	False	Capacity overflow during floating point value calculation
Bit 4 = 1	16	0x0010	True	The following behavior is displayed: ● `SP` lies outside the area [`pv_inf, pv_sup`]: `SP` is limited to `pv_inf` or `pv_sup` ● `dev_ll` > 0 or `dev_hl` < 0: the block uses the value 0

For the list of other possible floating point error codes, see *Common Floating Point Errors, page 365*.

Error message

An runtime error appears if a non floating point value is recorded or if there is a problem with a floating point calculation. The output `OUT` is then set to 0; the outputs `DEV` and `MA_O` remain unmodified.

NOTE: For a list of all block error codes and values, see *Controller, page 359*.

Warning

A warning is given if `dev_ll` > 0 is `dev_hl` < is 0 In this case the function block uses the value 0.

Chapter 20
STEP3: Three point controller

Introduction

This chapter describes the `STEP3` block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	206
Detailed description	209
Runtime error	211

Description

Function description

This function block is suitable for simple three-point step-action controls.

Control of the actuator proceeds according to the direction of the actual/setpoint value deviation in relation to the upper and lower threshold value. The control of the threshold value describes a configurable hysteresis.

This controller can also be used for temperature regulation. A traditional controller (such as a `PI_B` controller), which a function block such as the `PWM1` should be switched to is preferable for complex regulation.

`EN` and `ENO` can be configured as additional parameters.

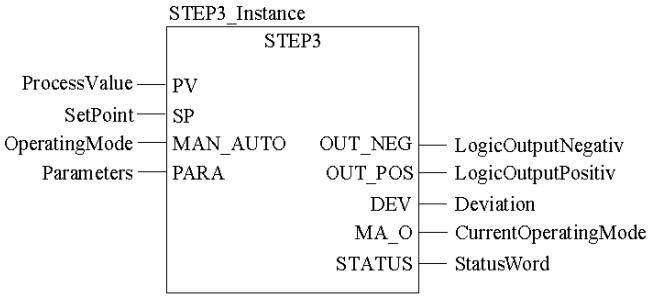
Properties

The control block has the following properties:

- Limiting the setpoint between `pv_inf` and `pv_sup`
- The control input values (actual value, setpoint, and corresponding parameters) are expressed in physical units.

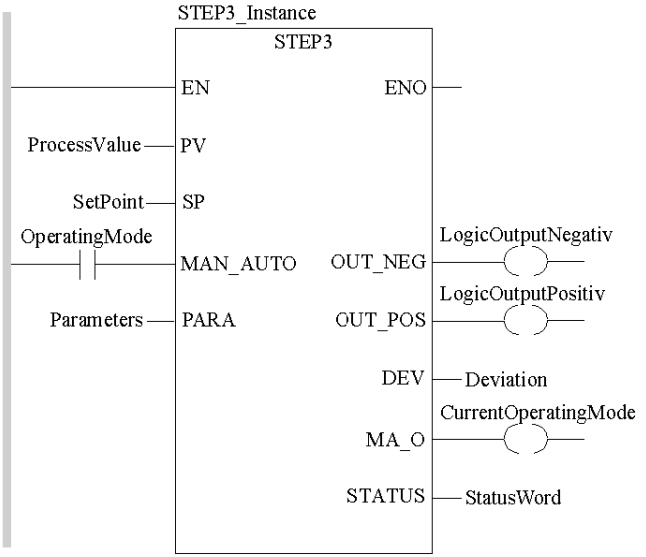
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL STEP3_Instance (PV:=ProcessValue, SP:=SetPoint,
    MAN_AUTO:=OperatingMode, PARA:=Parameters,
    OUT_NEG=>LogicOutputNegativ,
    OUT_POS=>LogicOutputPositiv, DEV=>Deviation,
    MA_O=>CurrentOperatingMode, STATUS=>StatusWord)
```

Representation in ST

Representation:

```
STEP3_Instance (PV:=ProcessValue, SP:=SetPoint,
    MAN_AUTO:=OperatingMode, PARA:=Parameters,
    OUT_NEG=>LogicOutputNegativ,
    OUT_POS=>LogicOutputPositiv, DEV=>Deviation,
    MA_O=>CurrentOperatingMode, STATUS=>StatusWord) ;
```

Parameter description STEP3

Input parameter description:

Parameter	Data type	Meaning
PV	REAL	Process value
SP	REAL	Setpoint
MAN_AUTO	BOOL	Controller operating mode: "1" : Automatic mode "0" : Halt mode
PARA	Para_STEP3	Parameter

Output parameter description:

Parameter	Data type	Meaning
OUT_NEG	BOOL	Logical output: is set to 1 for negative deviations
OUT_POS	BOOL	Logical output: is set to 1 for positive deviations
DEV	REAL	Deviation (PV-SP)
MA_O	BOOL	Current operating mode of the function block (0: Halt, 1: Automatic)
STATUS	WORD	Status word

Parameter description Para_STEP3

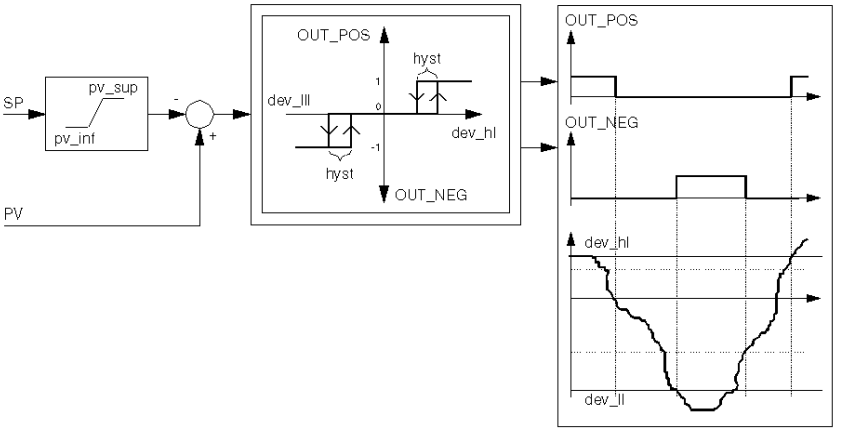
Data structure description

Element	Data type	Meaning
dev_ll	REAL	Lower deviation threshold (≤ 0)
dev_hl	REAL	Upper deviation threshold (≤ 0)
hyst	REAL	Hysteresis
pv_inf	REAL	Lower limit of the process value range
pv_sup	REAL	Upper limit of the process value range

Detailed description

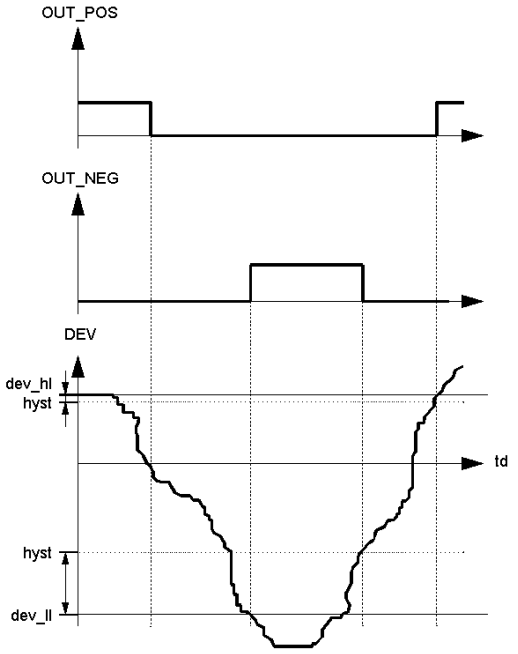
Structure diagram

The following is a structure diagram of the `STEP3` block:



Behavior of the outputs

Behavior of the OUT_POS and OUT_NEG blocks:



td Time span

If the deviation (DEV = PV - SP) climbs above dev_hl, the logical output OUT_POS is set to 1. If the deviation is less, OUT_POS is then only set to zero if the deviation is less than dev_hl – hyst.

If the deviation is less than dev_ll, the configured output OUT_NEG is set to 1. If the deviation increases again, OUT_NEG is only set to zero if the deviation exceeds dev_ll + hyst.

NOTE: To ensure that the block functions without errors, the outputs OUT_NEG and OUT_POS should not be inverted.

Operating mode

The STEP3 function block has 2 operating modes available according to the value of the MAN_AUTO parameter:

Operating mode	MAN_AUTO	Meaning
Automatic	1	The block calculates the outputs OUT_NEG and OUT_POS itself.
Halt	0	The outputs OUT_NEGandOUT_POS will be held at the last calculated value.

Runtime error

Status word

The following messages are displayed in the status word:

Bit	Value in Dec.	Value in Hex.	ENO Status	Description
Bit 0 = 1	1	0x0001	False	Error in a floating point value calculation
Bit 1 = 1	2	0x0002	False	Recording of an invalid value on one of the floating point value inputs
Bit 2 = 1	4	0x0004	False	Division by zero during a floating point value calculation
Bit 3 = 1	8	0x0008	False	Capacity overflow during floating point value calculation
Bit 4 = 1	16	0x0010	True	The following behavior is displayed: ● `SP` lies outside the area [`pv_inf, pv_sup`]: `SP` is limited to `pv_inf` or `pv_sup` in this case. ● `dev_ll` > 0 or `dev_hl` < 0: the block uses the value 0 ● `hyst` is outside the [0, Minimum (`dev_hl`, `-dev_ll`)] range: the block uses a value limited to zero or to minimum (`dev_hl`, `-dev_ll`)

For the list of other possible floating point error codes, see *Common Floating Point Errors, page 365*.

Error message

An runtime error appears if a non floating point value is recorded or if there is a problem with a floating point calculation. In this case the outputs `OUT_NEG` and `OUT_POS` are set to 0; the `DEV` and `MA_O` outputs remain unmodified.

NOTE: For a list of all block error codes and values, see *Controller, page 359*.

Warning

In the following cases a warning is given:

- `dev_ll` > 0 or `dev_hl` < 0: the block uses the value 0.
- `hyst` is outside the [0, Minimum (`dev_hl`, `-dev_ll`)] range: the block uses a limited value.

Part IV
Mathematics

Overview

This section describes the elementary functions and elementary function blocks of the `Mathematics` family.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
21	COMP_DB: Comparison	215
22	K_SQRT: Square root	221
23	MULDIV_W: Multiplication/Division	225
24	SUM_W: Summer	229

Chapter 21
COMP_DB: Comparison

Introduction

This chapter describes the COMP_DB block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	216
Detailed description	219

Description

Function description

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

Depending whether IN1 is greater than, equal to or less than IN2, the function blocks sets one of the outputs GREATER, EQUAL or LESS to 1.

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

EN and ENO can be configured as additional parameters.

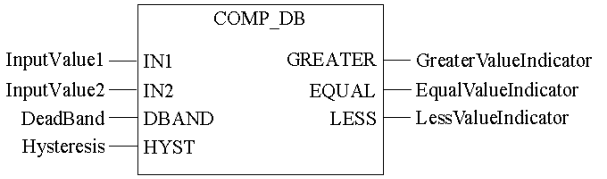
COMP_DB/COMP_DB_DFB

Creating new programs, always use the COMP_DB function block.

For technical reasons the COMP_DB_DFB may be implemented during converting legacy programs. But the functionality of COMP_DB and COMP_DB_DFB is exactly the same.

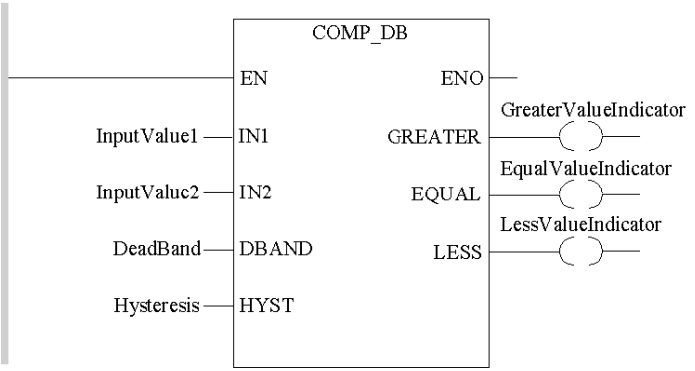
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL COMP_DB (IN1=:InputValue1, IN2:=InputValue2,
        DBAND=:DeadBand, HYST:=Hysteresis,
        GREATER=>GreaterValueIndicator,
        EQUAL=>EqualValueIndicator, LESS=>LessValueIndicator)
```

Representation in ST

Representation:

```
COMP_DB (IN1=:InputValue1, IN2:=InputValue2,
        DBAND=:DeadBand, HYST:=Hysteresis,
        GREATER=>GreaterValueIndicator,
        EQUAL=>EqualValueIndicator,
        LESS=>LessValueIndicator);
```

Parameter description

Description of input parameters:

Parameter	Data type	Description
IN1	REAL	Input No. 1
IN2	REAL	Input No. 2
DBAND	REAL	Dead zone
HYST	REAL	Hysteresis

Description of output parameters:

Parameter	Data type	Description
GREATER	BOOL	Greater-than marker
EQUAL	BOOL	Equals marker
LESS	BOOL	Less-than marker

Runtime error

An error is displayed in the Diagnostics display (**View →Diagnostics View**) if a non floating point value is determined at an input, or if a problem occurs when calculating a floating point value. In this case the GREATER, EQUAL and LESS outputs remain unchanged.

NOTE: For a list of all block error codes and values, see *Mathematics, page 361*.

Warning

A warning is displayed in the Diagnostics display (**View →Diagnostics Display**) if,

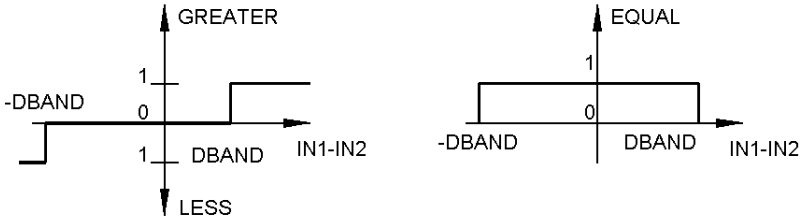
- the DBAND parameter is negative: the procedure then uses the value DeadBand = 0 for calculation.
- The HYST parameter is outside the [0, DeadBand] range: the procedure then uses the closest correct value for calculation, i.e. 0, if HYST is less than 0 and DBAND, if HYST is greater than DBAND.

Detailed description

Dead zone

The DBAND parameter enables a dead zone to be specified, within which deviation between IN1 and Inputvalue2 will be regarded as zero. If the deviation IN1 - IN2 remains within this zone, the EQUAL output is set to 1.

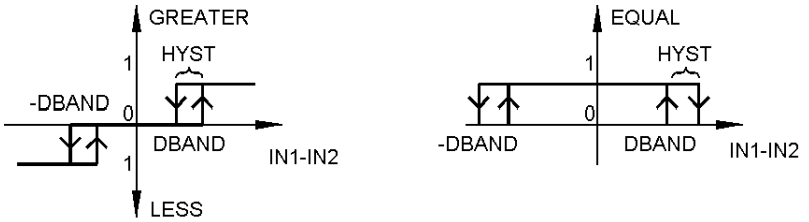
Dead zone specification



HYST

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

Generating a hysteresis effect

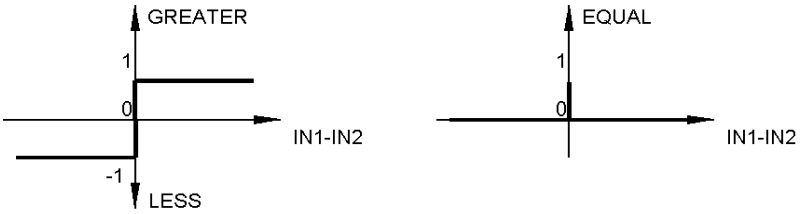


DBAND = 0 and HYST = 0

In this case, the block behaves like a classic comparison function:

- If IN1 is always greater than IN2, then GREATER = 1
- If IN1 is equal to IN2, then EQUAL = 1
- If IN1 is less than IN2, then LESS = 1

Classic comparison function (DBAND = 0 and HYST = 0)



Chapter 22
K_SQRT: Square root

Description

Function description

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

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

EN and ENO can be configured as additional parameters.

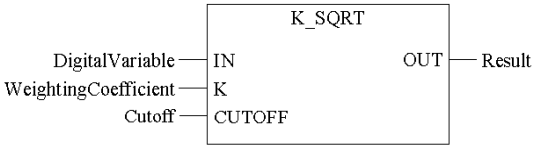
Formula

The function performs the following calculation:

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

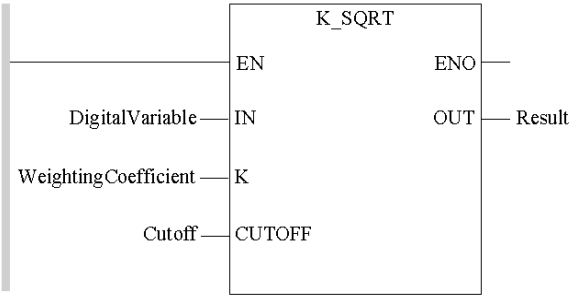
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD DigitalVariable
K_SQRT WeightingCoefficient, Cutoff
ST OUT
```

Representation in ST

Representation:

```
Result := K_SQRT (DigitalVariable, WeightingCoefficient,
                  Cutoff);
```

Parameter description

Description of input parameters:

Parameter	Data type	Description
DigitalVariable	REAL	Numerical value to process
WeightingCoefficient	REAL	Weighting coefficient
Cutoff	REAL	Division

Description of output parameters:

Parameter	Data type	Description
Result	REAL	Result of the calculation

Runtime error

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

NOTE: For a list of all block error codes and values, see *Mathematics, page 361*.

Warning

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

Chapter 23
MULDIV_W: Multiplication/Division

Description

Function description

The function `MULDIV_W` performs a weighted multiplication/division from 3 numerical input variables.

`EN` and `ENO` can be configured as additional parameters.

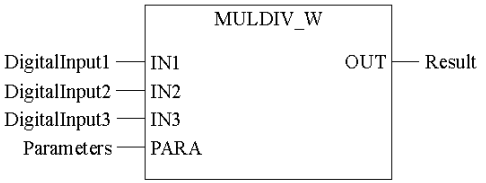
Formula

The equation says:

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

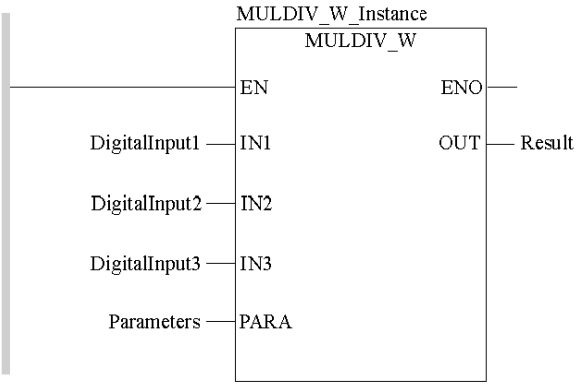
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD DigitalInput1
MULDIV_W DigitalInput2, DigitalInput3, Parameters
ST Result)
```

Representation in ST

Representation:

```
Result := MULDIV_W (DigitalInput1, DigitalInput2,
                    DigitalInput3, Parameters);
```

Parameter description `MULDIV_W`

Description of input parameters:

Parameter	Data type	Description
`DigitalInput1` to `DigitalInput3`	`REAL`	Numerical variables to be processed
`Parameters`	`Para_MULDIV_W`	Parameter

Description of output parameters:

Parameter	Data type	Description
`Result`	`REAL`	Result of the calculation

Parameter description PARA_MULDIV_W

Data structure description

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

Runtime error

This error will be signaled if a non floating point value is recorded or if there is a problem with a floating point calculation. In general, the output Result keeps its previous value, apart from with a division by 0, where the value corresponds to 1.#INF (infinite) depending on which sign the counter uses.

NOTE: For a list of all the block error messages and values, see *Common Floating Point Errors, page 365*.

Chapter 24
SUM_W: Summer

Description

Function description

The function performs the weighted summation of 3 numerical input variables according to the underlying formula.

EN and ENO can be configured as additional parameters.

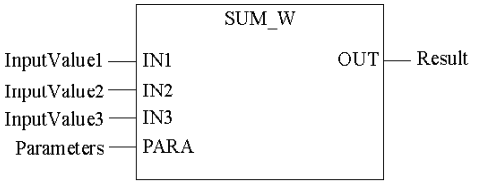
Formula

The function SUM_W operates as follows:

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

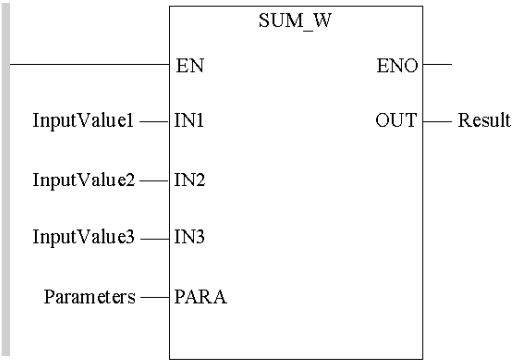
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD InputValue1
SUM_W InputValue2, InputValue3, Parameters
ST Result
```

Representation in ST

Representation:

```
Result := SUM_W (InputValue1, InputValue2, InputValue3,
                 Parameters);
```

Parameter description SUM_W

Description of input parameters:

Parameter	Data type	Description
`InputValue1` bis `InputValue3`	`REAL`	Numerical variables to be processed
`Parameters`	`Para_SUM_W`	Parameter

Description of output parameters:

Parameter	Data type	Description
`Result`	`REAL`	Result of the calculation

Parameter description `Para_SUM_W`

Data structure description

Element	Data type	Description
`k1` to `k3, c1`	`REAL`	Calculation coefficients

Runtime error

An runtime error appears if a non floating point value is recorded or if there is a problem with a floating point calculation. The output `Result` will not be altered.

NOTE: For a list of all the block error messages and values, see *Common Floating Point Errors, page 365*.

Part V
Measurement

Overview

This section describes the elementary functions and elementary function blocks of the `Measurement` family.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
25	AVGMV: Moving average with fixed window size	235
26	AVGMV_K: Moving average with frozen correction factor	241
27	DEAD_ZONE, DEAD_ZONE_REAL: Dead zone	247
28	LOOKUP_TABLE1: Polygon with interpolation of the 1st order	253
29	SAH: Detecting and holding a rising edge	259
30	HYST_***: Indicator signal for maximum value delimiters with hysteresis	261
31	INDLIM_***: Indicator signal for delimiters with hysteresis	265

Chapter 25
AVGMV: Moving average with fixed window size

Introduction

This chapter describes the AVGMV block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	236
Detailed description	239

Description

Function description

The function block creates a moving average from a fixed number of input values (input X). The output is the average of all values between the current X value and the oldest X value (N-1). Up to 50 input values can be stored (N).

The function block has both a manual and automatic mode.

NOTE: This function block performs an internal initialization in the first program cycle after a warm start or cold start (e.g. application download or power cycle) of the PLC program.
Due to this, you have to make sure that the function block is invoked in the first program cycle. In case of inkoving the function block in a later program cycle, the internal initialization will not be performed and the ouputs may deliver wrong values.

⚠ WARNING

UNEXPECTED OUTPUT BEHAVIOUR

Make sure that the function block is always invoked in the first program cycle.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

EN and ENO can be configured as additional parameters.

Formula

For RDY = 1:

$$Y(new) = \frac{\sum_{i=0}^{N-1} X_i}{N}$$

or

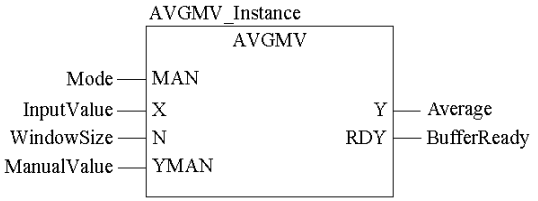
$$Y_{(new)} = Y_{(old)} + \frac{X}{N} - \frac{X_{(N-1)}}{N}$$

Explanation of the variables

Variable	Description
$Y_{(new)}$	Y Value in current program cycle
$Y_{(old)}$	Y Value from the last program cycle
N	Window size (number of values in the buffer)
$X_{(N-1)}$	oldest X value in the buffer

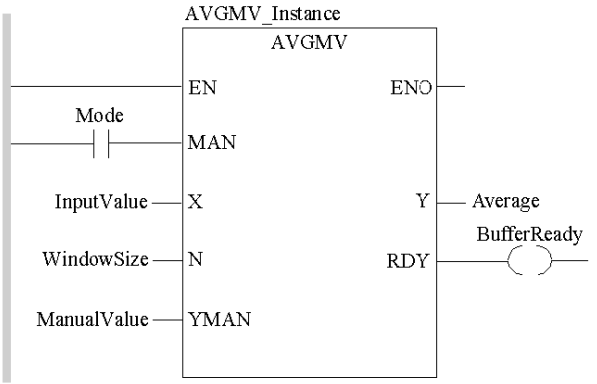
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL AVGMV_Instance (MAN:=Mode, X:=InputValue,
    N:=WindowSize, YMAN:=ManualValue, Y=>Average,
    RDY=>BufferReady)
```

Representation in ST

Representation:

```
AVGMV_Instance (MAN:=Mode, X:=InputValue,
    N:=WindowSize, YMAN:=ManualValue, Y=>Average,
    RDY=>BufferReady) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Description
MAN	BOOL	"0" = Automatic operating mode "1" = Manual operating mode
X	REAL	Input
N	INT	Window size (number of input values that are loaded in the buffer; max. 50)
YMAN	REAL	Manual value

Description of output parameters:

Parameter	Data type	Description
Y	REAL	Average value
RDY	BOOL	"1" = n Value in buffer, i.e. buffer is ready "0" = Buffer not ready

Runtime error

An error message is returned if

- N=0 or N>50

NOTE: For a list of all block error codes and values, see *Measurement, page 361*.

Detailed description

Automatic operating mode

In N program cycles, N X values are read into an internal buffer. The arithmetic mean of these values is calculated, and is delivered to the Y output. From the N+1 program cycle onwards, the oldest X value in the buffer is deleted and replaced with the current X value.

NOTE: As long as RDY = 0, the mean is not derived from N values but from the current updated read-in number (n < N).

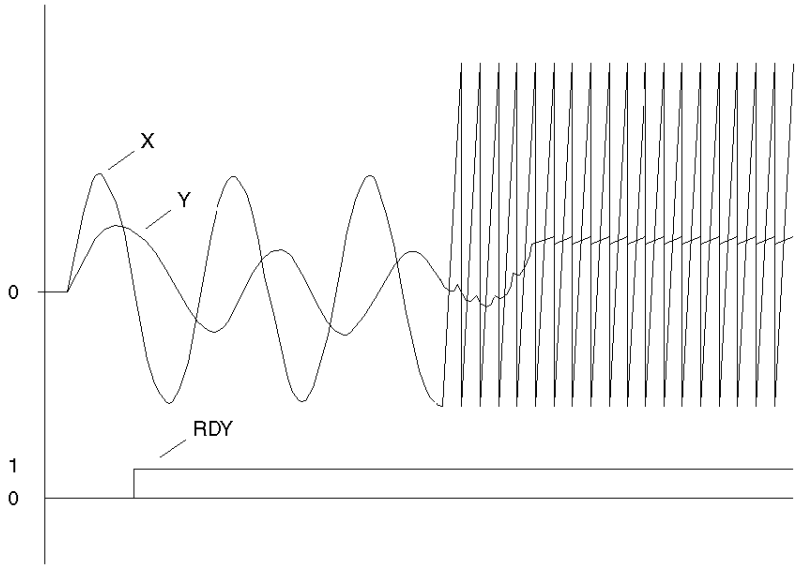
After a modification of the N value or after a cold/warm start, the internal buffer is deleted. The output is set to the input value X and RDY to "0". The buffer is filled during the next N cycles. The Y output contains a mean of the values accumulated so far. RDY remains "0" until the buffer is filled with correct X values after N program cycles then RDY becomes "1".

Manual operating mode

The value YMAN is transferred to the Y output. The buffer is completely filled with the value YMAN and marked as full (RDY = 1).

Diagram

Floating mean with limited memory of N = 50 values



Chapter 26
AVGMV_K: Moving average with frozen correction factor

Introduction

This chapter describes the AVGMV_K block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	242
Detailed description	245

Description

Function description

The function block creates a moving average value (with frozen correction factor) with up to 10,000 input values.

The function block has both a manual and automatic mode.

NOTE: This function block performs an internal initialization in the first program cycle after a warm start or cold start (e.g. application download or power cycle) of the PLC program.
Due to this, you have to make sure that the function block is invoked in the first program cycle. In case of invoking the function block in a later program cycle, the internal initialization will not be performed, the value Y(old) will not be sure and the output Y(new) may deliver a wrong value.

⚠ WARNING

UNEXPECTED OUTPUT BEHAVIOUR

Make sure that the function block is always invoked in the first program cycle.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

EN and ENO can be configured as additional parameters.

Formula

Block formula:

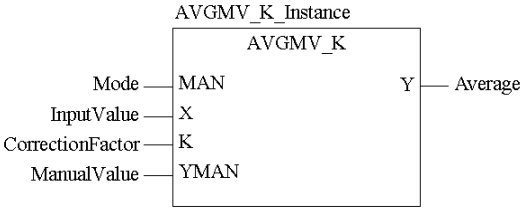
$$Y_{(new)} = Y_{(old)} + \frac{X - Y_{(old)}}{K}$$

Explanation of the variables

Variable	Description
Y (new)	Y Value in current program cycle
Y (old)	Y Value from the last program cycle
K	Correction factor
X	X Value in current program cycle

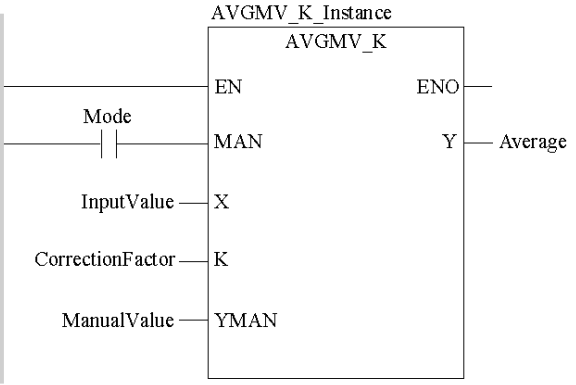
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL AVGMV_K_Instance (MAN:=Mode, X:=InputValue,
    K:=CorrectionFactor, YMAN:=ManualValue, Y=>Average)
```

Representation in ST

Representation:

```
AVGMV_K_Instance (MAN:=Mode, X:=InputValue,
    K:=CorrectionFactor, YMAN:=ManualValue, Y=>Average) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Description
MAN	BOOL	"0" = Automatic operating mode; "1" = Manual operating mode
X	REAL	Input
K	INT	Correction factor (max. 10,000)
YMAN	REAL	Manual value

Description of output parameters:

Parameter	Data type	Description
Y	REAL	Average value

Runtime error

For a list of all block error codes and values, see *Measurement, page 361*.

Detailed description

Automatic operating mode

One X word is read per program cycle. 1/K is deducted from the Y value of the last program cycle, and then 1/K of the current X value is added. The result is given at the Y output.

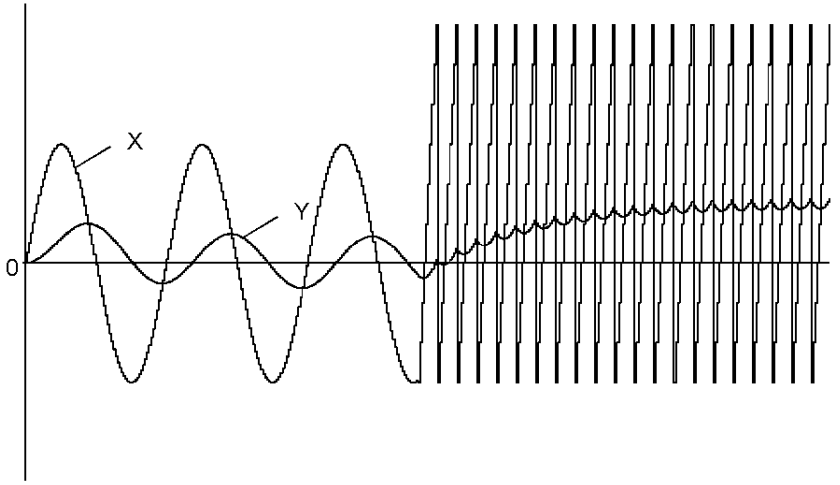
After a coldstart/warmstart, the value X is assigned to the Y output.

Manual operating mode

The value YMAN is transferred to the Y output.

Diagram

Moving average with frozen correction factor (K = 50)



Chapter 27
DEAD_ZONE, DEAD_ZONE_REAL: Dead zone

Introduction

This chapter describes the blocks DEAD_ZONE and DEAD_ZONE_REAL.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	248
Detailed description	250

Description

Function description

These functions are used to specify the dead zone for controlled variables.

EN and ENO can be configured as additional parameters.

Formula

Block formula:

Assuming: DZ ≥ 0

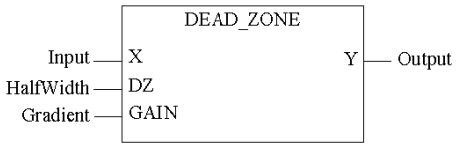
Y = GAIN x X for −DZ ≤X ≤DZ

Y = (X − DZ) + GAIN x DZ for X > DZ

Y = (X + DZ) - GAIN x DZ for X < −DZ

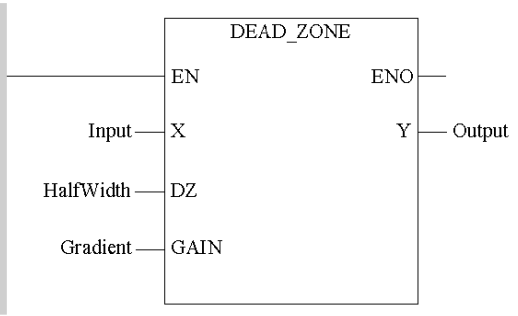
Appearance in FBD

Block Presentation DEAD_ZONE:



Appearance in LD

Block Presentation DEAD_ZONE:



Appearance in IL

Block Presentation `DEAD_ZONE`:

```
LD Input
DEAD_ZONE HalfWidth, Gradient
ST Output
```

Appearance in ST

Block Presentation `DEAD_ZONE`:

```
Output := DEAD_ZONE (Input, HalfWidth, Gradient) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Description
`X`	`REAL`	Input variable
`DZ`	`REAL`	Half width of the dead zone
`GAIN`	`REAL`	Gradient within the dead zone

Description of output parameters:

Parameter	Data type	Description
`Y`	`REAL`	Output variable

Runtime error

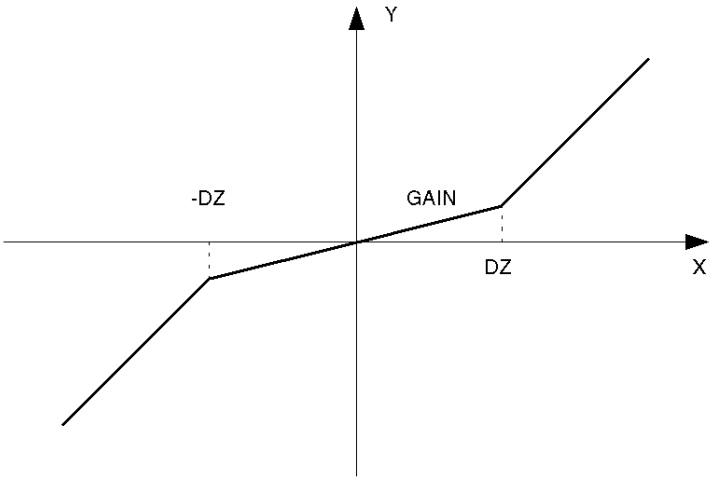
For a list of all block error codes and values, see *Measurement, page 361*.

Detailed description

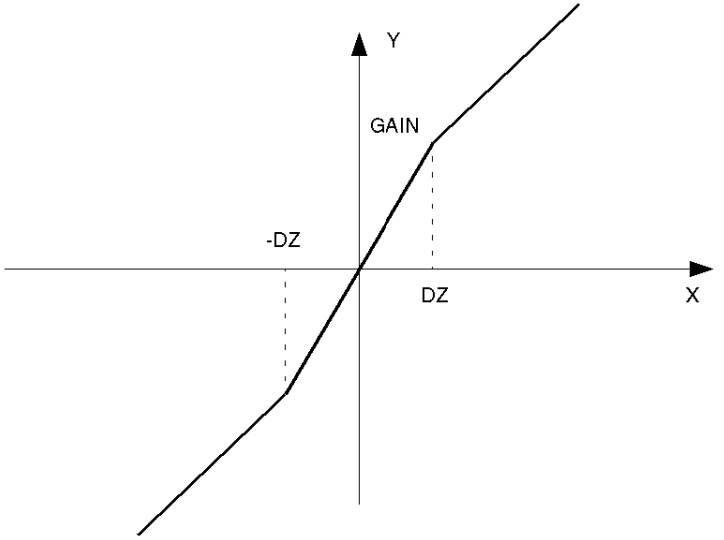
Characteristic Curves

The function block has the following characteristic curve:

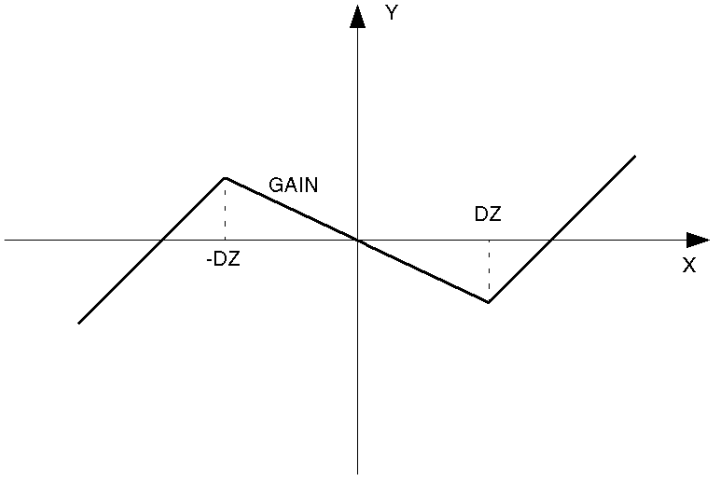
Dead zone with 0 < GAIN < 1



Dead zone with GAIN > 1



Dead zone with GAIN < 0



NOTE: A gradient of 1 has been specified outside the dead zone.

Chapter 28
LOOKUP_TABLE1: Polygon with interpolation of the 1st order

Introduction

This chapter describes the `LOOKUP_TABLE1` block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	254
Detailed description	256

Description

Function description

This procedure linearizes characteristic curves using interpolation. The procedure works with variable support point widths.

The number of `X/Y_Coord_SupportPoint_n` inputs can be increased from 2 to a maximum of 30 by vertically resizing the block frame. This corresponds to a maximum of 15 support points.

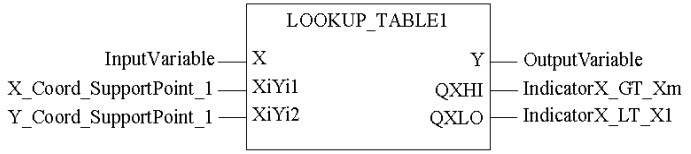
The number of inputs must be even.

The `InputVariable` values must be in ascending order.

`EN` and `ENO` can be configured as additional parameters. It is recommended to use the **ENO** output parameter as an error indicator.

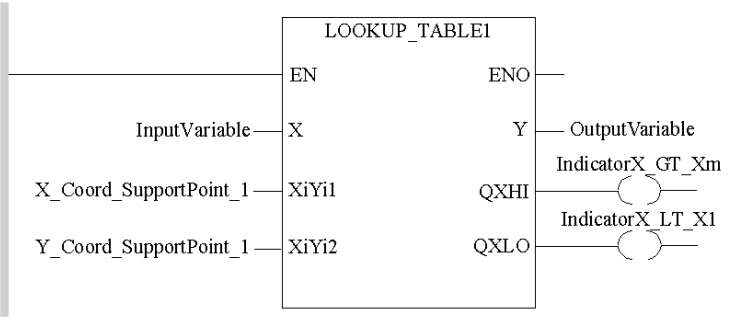
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD InputVariable
LOOKUP_TABLE1 X_Coord_SupportPoint_1,
              Y_Coord_SupportPoint_1, OutputVariable,
              IndicatorX_GT_Xm, IndicatorX_LT_X1
```

Representation in ST

Representation:

```
LOOKUP_TABLE1 (InputVariable, X_Coord_SupportPoint_1,
               Y_Coord_SupportPoint_1, OutputVariable,
               IndicatorX_GT_Xm, IndicatorX_LT_X1);
```

Parameter description

Description of input parameters:

Parameter	Data type	Description
InputVariable	REAL	Input variable
X_Coord_SupportPoint_1	REAL	X coordinate 1. Support point
Y_Coord_SupportPoint_1	REAL	Y coordinate 1. Support point
X_Coord_SupportPoint_1(n-1)	REAL	X coordinate $\frac{n}{2}$. Support point n=max 30
Y_Coord_SupportPoint_(n)	REAL	Y coordinate $\frac{n}{2}$. Support point n=max 30

Description of output parameters:

Parameter	Data type	Description
OutputVariable	REAL	Output variable
IndicatorX_GT_Xm	BOOL	Display: X > Xm
IndicatorX_LT_X1	BOOL	Indicate X < X1

Runtime error

NOTE: For a list of all block error codes and values, see Tables of Error Codes for the CONT_CLC Library *(see page 361)*.

Detailed description

Parameter description

Two inputs in sequence (XiYi) represent a support point pair. The first input XiYi corresponds to X1, the nextY1, the nextX2 etc.

For any X input value falling between these support points, the corresponding Y output is linearly interpolated.

For $X < X_1$ is $Y = Y_1$

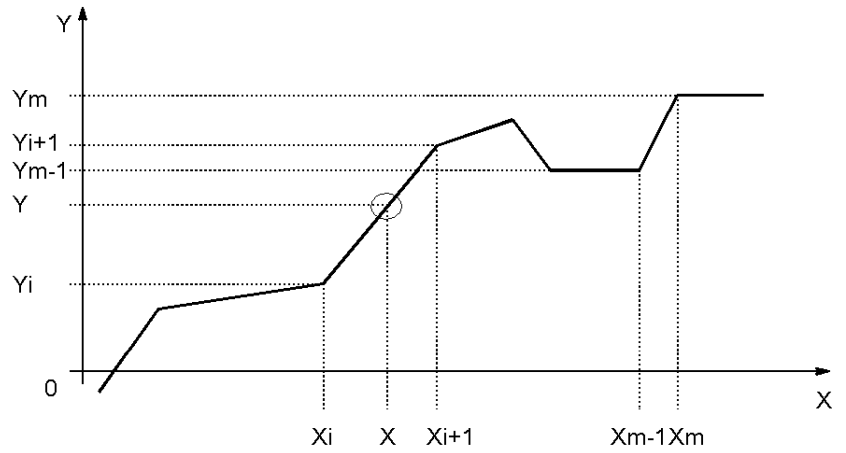
For $X > X_m$ is $Y = Y_m$

If the value at the X input exceeds the last support pointXm , the QXHI output becomes "1".

If the value at the X input is lower than the value of the first support point X1 , the QXLO output becomes "1".

Principle of Interpolation

Polygon with interpolation of the 1st order



Interpolation

The following algorithm applies to the Y coordinate:

$$Y = Y_i + \frac{Y_i + 1 - Y_i}{X_i + 1 - X_i} \times (X - X_i)$$

for $X_i \leq X \leq X_{i+1}$ and i = 1 ... (m-1)

Assuming: $X_1 \leq X_2 \leq .. \leq X_i \leq X_{i+1} \leq .. \leq X_{m-1} \leq X_m$

The X values must be in ascending order.

Two Xvalues in a row may be the same. This means a discontinuous curve is possible.

There are exceptions:

$Y = 0.5 \times (Y_i + Y_{i+1})$

for

$X_i = X = X_{i+1}$ and i = 1 ... (m-1)

Chapter 29
SAH: Detecting and holding a rising edge

Description

Function description

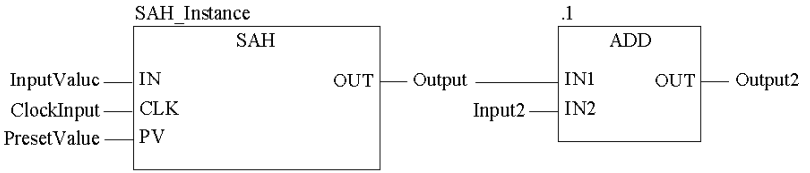
The function block passes the input value PV to the output OUT the first time it is called. With a rising edge (0 to 1) on input CLK, the input value IN is pased on to output Output. This value remains on the output until the next rising edge loads a new value from IN to OUT.

The data types for the input values IN, PV and the output value Output must be the same.

EN and ENO can be configured as additional parameters.

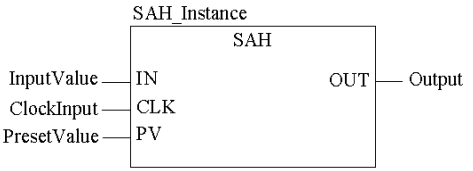
Note

If a variable **and** a link are connected to OUT and the variable Output is changed (e.g. by writing through the program or by forcing), the link follows the Output variable instead of keeping the last output of the SAH.



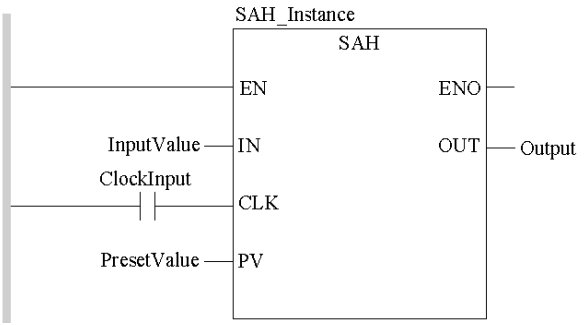
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL SAH_Instance (IN:=InputValue, CLK:=ClockInput,
    PV:=PresetValue, OUT=>Output)
```

Representation in ST

Representation:

```
SAH_Instance (IN:=InputValue, CLK:=ClockInput,
    PV:=PresetValue, OUT=>Output) ;
```

Parameter description

Input parameter description:

Parameter	Data type	Meaning
IN	ANY	Input value
CLK	BOOL	Clock input
PV	ANY	Presetpoint

Output parameter description:

Parameter	Data type	Meaning
OUT	ANY	Output value

Chapter 30
HYST_***: Indicator signal for maximum value delimiters with hysteresis

Introduction

This chapter describes the `HYST_***` block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	262
Detailed description	264

Description

Function description

The function block monitors the input variable X to detect whether it exceeds the upper limit.

NOTE: If you need to monitor the lower limit you can use the INDLIM_*** function block.

The data types for all input values must be the same.

EN and ENO can be configured as additional parameters.

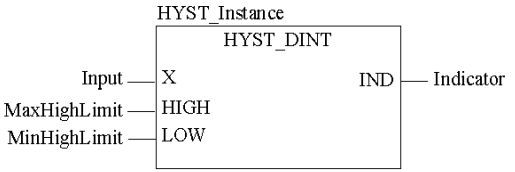
Available functions

The following blocks are available:

- HYST_DINT
- HYST_INT
- HYST_UDINT
- HYST_UINT
- HYST_REAL

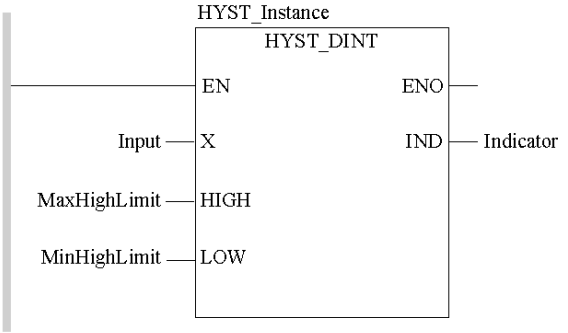
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL HYST_Instance (X:=Input, HIGH:=MaxHighLimit,
    LOW:=MinHighLimit, IND=>Indicator)
```

Representation in ST

Representation:

```
HYST_Instance (X:=Input, HIGH:=MaxHighLimit,
    LOW:=MinHighLimit, IND=>Indicator) ;
```

Parameter description

Input parameter description:

Parameter	Data type	Meaning
X	INT, DINT, UINT, UDINT, REAL	Input variable
HIGH	INT, DINT, UINT, UDINT, REAL	Maximum upper limit
LOW	INT, DINT, UINT, UDINT, REAL	Minimum upper limit

Output parameter description:

Parameter	Data type	Meaning
IND	BOOL	Display: Upper limit reached

Detailed description

Parameter description

If X exceeds the upper limit HIGH, the function block reports this status with IND = 1.

If X exceeds the lower limit LOW, the function block reports this status with IND = 0.

Function

Function description

INDi = 1, if X > HIGH

INDi = 0, if X < LOW

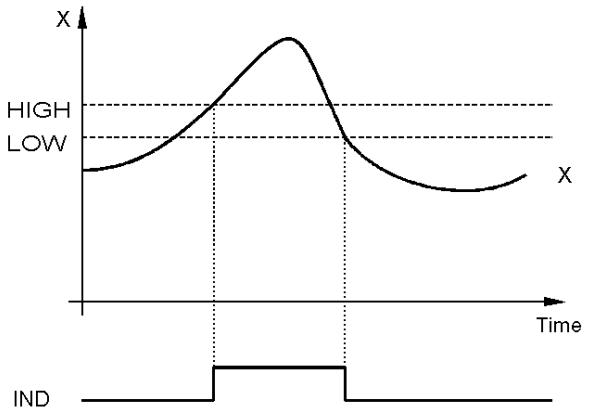
otherwise

INDi = INDi-1

If LOW is greater than HIGH, no hysterisis is created and IND indicates that X is greater than HIGH.

Diagram

Maximum value delimiter with hysteresis



Chapter 31
INDLIM_***: Indicator signal for delimiters with hysteresis

Introduction

This chapter describes the `INDLIM_***` block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	266
Detailed description	269

Description

Function description

The function block monitors the input variable X to detect whether it exceeds the upper limit or falls below the lower limit.

NOTE: If you only want to monitor the upper limit you can use the HYST_*** function block.

The data types for all input values must be the same.

EN and ENO can be configured as additional parameters.

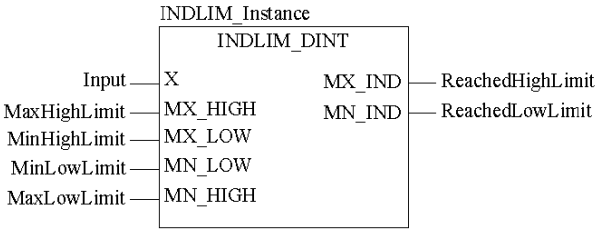
Available functions

The following blocks are available:

- INDLIM_DINT
- INDLIM_INT
- INDLIM_UDINT
- INDLIM_UINT
- INDLIM_REAL

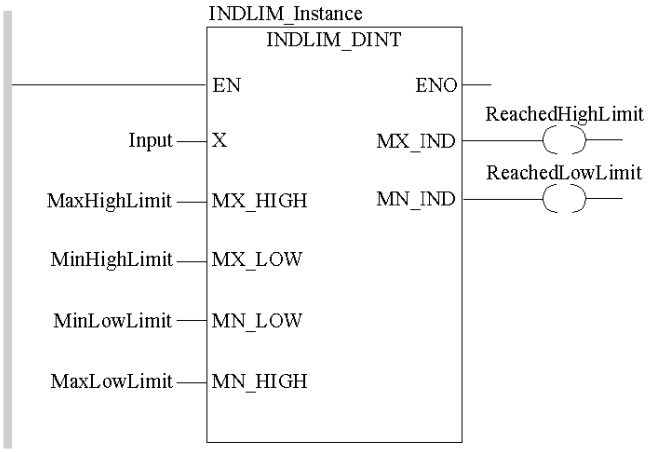
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL INDLIM_Instance (X:=Input, MX_HIGH:=MaxHighLimit,
    MX_LOW:=MinHighLimit, MN_LOW:=MinLowLimit,
    MN_HIGH:=MaxLowLimit, MX_IND=>ReachedHighLimit,
    MN_IND=>ReachedLowLimit)
```

Representation in ST

Representation:

```
INDLIM_Instance (X:=Input, MX_HIGH:=MaxHighLimit,
    MX_LOW:=MinHighLimit, MN_LOW:=MinLowLimit,
    MN_HIGH:=MaxLowLimit, MX_IND=>ReachedHighLimit,
    MN_IND=>ReachedLowLimit) ;
```

Parameter description

Input parameter description:

Parameter	Data type	Meaning
X	INT, DINT, UINT, UDINT, REAL	Input variable
MX_HIGH	INT, DINT, UINT, UDINT, REAL	Maximum upper limit
MX_LOW	INT, DINT, UINT, UDINT, REAL	Maximum lower limit
MN_LOW	INT, DINT, UINT, UDINT, REAL	Minimum lower limit
MN_HIGH	INT, DINT, UINT, UDINT, REAL	Minimum upper limit

Output parameter description:

Parameter	Data type	Meaning
MX_IND	BOOL	Display: Upper limit reached
MN_IND	BOOL	Display: Lower limit reached

Detailed description

Parameter description MX_IND

If X exceeds the value of MX_HIGH, the function block reports this status with MX_IND = 1.

If X falls below the value of MX_LOW, the function block reports this status with MX_IND = 0.

Formulas:

MX_INDi = 1, if X > MX_HIGH

MX_INDi = 0, if X < MX_LOW

otherwise

MX_INDi = MX_IND(i-1)

If MX_LOW is greater than MX_HIGH, no hysterisis is created and the function block reports this state with MX_IND = 1.

Parameter description MN_IND

If X falls below the value of MN_HIGH, the function block reports this status with MN_IND = 1.

If X exceeds the value of MN_LOW, the function block reports this status with MN_IND = 0.

Formulas:

MN_INDi = 1, if X < MN_HIGH

MN_INDi = 0, if X > MN_LOW

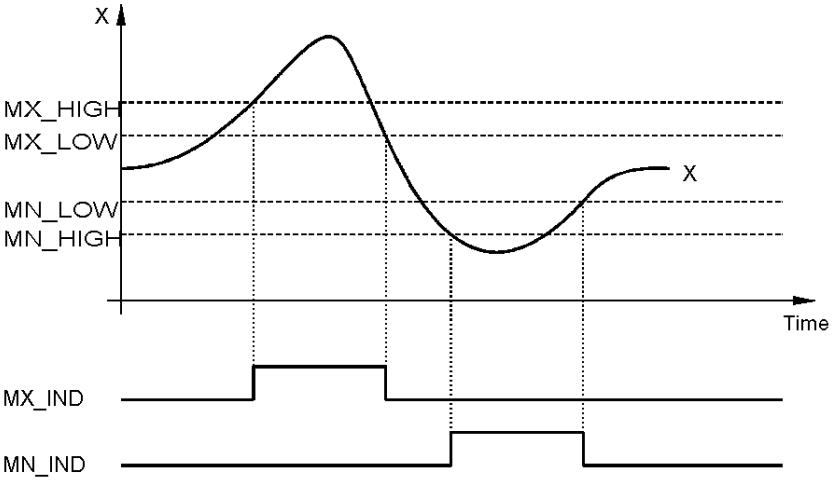
otherwise

MN_INDi = MN_IND(i-1)

If MN_HIGH is greater than MN_LOW, no hysterisis is created and the function block reports this state with MN_IND = 1.

Diagram

Delimiter with hysteresis `INDLIM`



Part VI
Output Processing

Overview

This section describes the elementary functions and elementary function blocks of the `Output Processing` family.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
32	MS: Manual control of an output	273
33	MS_DB: Manually controlling and output with dead zone	285
34	PWM1: Pulse width modulation	297
35	SERVO: Control for servo motors	305
36	SPLRG: Controlling 2 actuators	323

Chapter 32
MS: Manual control of an output

Introduction

This chapter describes the MS block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	274
Detailed description	278
Example	281
Runtime error	282

Description

Function description

This function block serves as the control of a numerical output, which can be switched off via the function block PWM1 *(see page 297)* controlled analog output, server motor or controlling element. The control can appear via server dialog or direct via the PLC software.

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

EN and ENO can be configured as additional parameters.

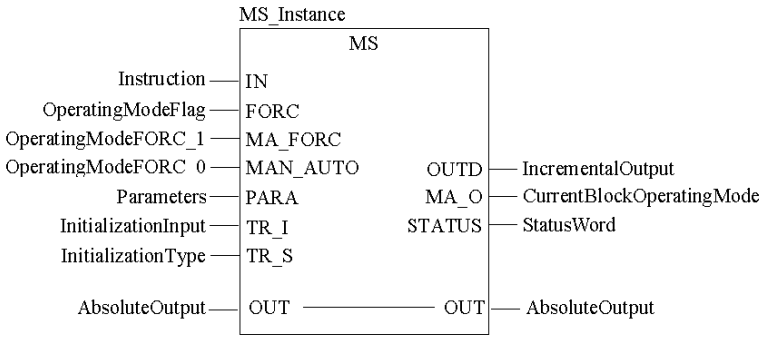
Application possibilities

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

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

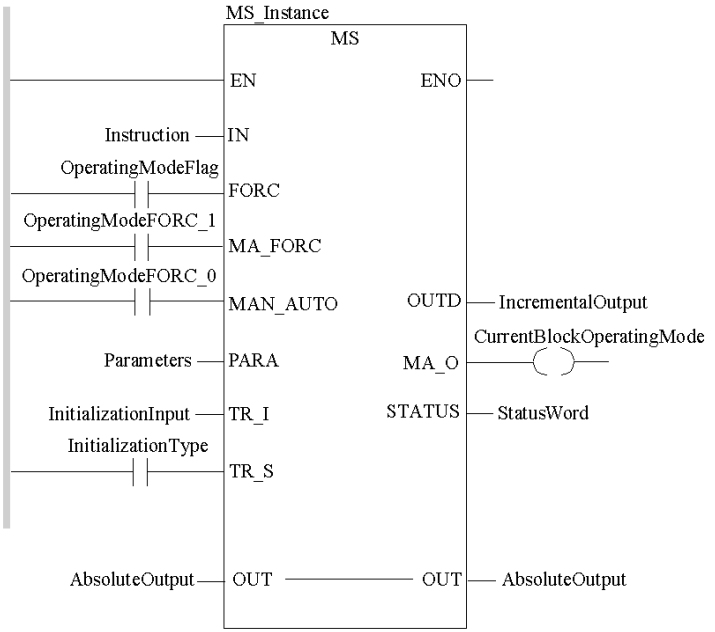
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL MS_Instance (IN:=Instruction, FORC:=OperatingModeFlag,
    MA_FORC:=OperatingModeFORC_1,
    MAN_AUTO:=OperatingModeFORC_0, PARA:=Parameters,
    TR_I:=InitializationInput, TR_S:=InitializationType,
    OUT:=AbsoluteOutput, OUTD=>IncrementalOutput,
    MA_O=>CurrentBlockOperatingMode, STATUS=>StatusWord)
```

Representation in ST

Representation:

```
MS_Instance (IN:=Instruction, FORC:=OperatingModeFlag,
    MA_FORC:=OperatingModeFORC_1,
    MAN_AUTO:=OperatingModeFORC_0, PARA:=Parameters,
    TR_I:=InitializationInput, TR_S:=InitializationType,
    OUT:=AbsoluteOutput, OUTD=>IncrementalOutput,
    MA_O=>CurrentBlockOperatingMode, STATUS=>StatusWord) ;
```

Parameter description MS

Input parameter description:

Parameter	Data type	Meaning
IN	REAL	Manipulated variable used in automatic mode
FORC	BOOL	"1": The mode manual/automatic will be entered via MA_FORC "0": The mode manual/automatic will be entered via MAN_AUTO
MA_FORC	BOOL	Mode manual/automatic (if FORC = 1) "1": Automatic operating mode "0": Manual mode
MAN_AUTO	BOOL	Mode manual/automatic (if FORC = 0) "1": Automatic operating mode "0": Manual mode
PARA	Para_MS	Parameter
TR_I	REAL	Initialization input
TR_S	BOOL	Initialization command

Input / Output parameter description:

Parameter	Data type	Meaning
OUT	REAL	Absolute output

Output parameter description:

Parameter	Data type	Meaning
OUTD	REAL	Incremental output: Difference between the present output and the output of the previous execution
MA_O	BOOL	Current mode of the function block (0: Manual, 1: Automatic)
STATUS	WORD	Status word

Parameter description Para_MS

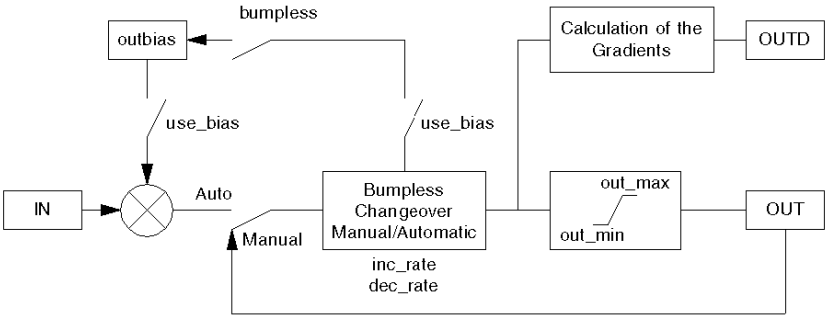
Data structure description

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

Detailed description

Structure diagram

The following diagram displays the structure of the function block:



Setting of the mode selection

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

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

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

Characteristics of the output OUT

The following characteristics apply to the output OUT:

- Automatic mode: The output OUT is a copy of the input IN.
  In this operating mode, the output OUT can be assigned an OUTBIAS value (use _bias to 1).
  OUT calculates as follows: OUT = IN + outbias.
- Manual mode: The function block does not set the output, the server can directly change the value that is the connected variable at the output OUT.
- The output OUT is principally limited to an area between out_min and out_max . When the value calculated by the function block (or entered by the server in manual mode) exceeds one of these limit values, the value of OUT will be cut (to out_min or out_max).). The incremental output OUTD on the other hand, does not take this cut into consideration.

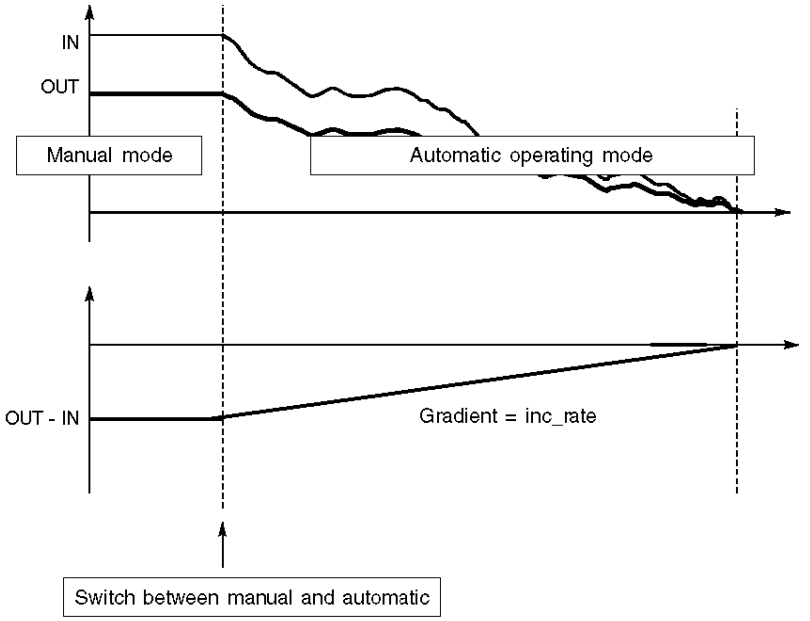
Switch between manual and automatic

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

The output OUT gets closer to input IN ramps with positive (inc_rate) or negative increase (dec_rate):

- inc_rate applies when IN is larger than OUT at the time of the changeover
- dec_rate applies when IN is less than OUT at the time of the changeover

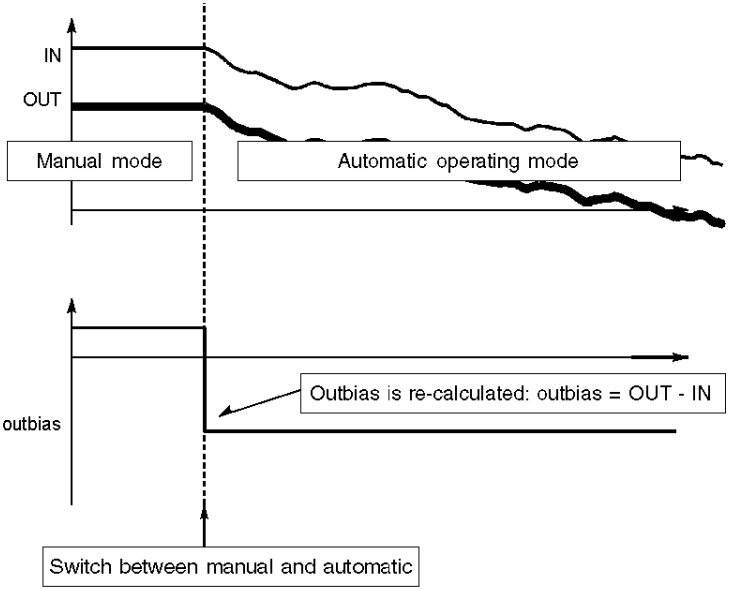
bumpless changeover



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

When the parameter Outbias (`use_bias` = 1) is used, a bumpless manual/automatic changeover can be achieved without change of the output, when the `bumpless` parameter is set to 1. In this case the parameter Outbias will be newly calculated and the deviation between the input `IN` and the output `OUT` will be considered.

Bumpless changeover with the parameter `Outbias`



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

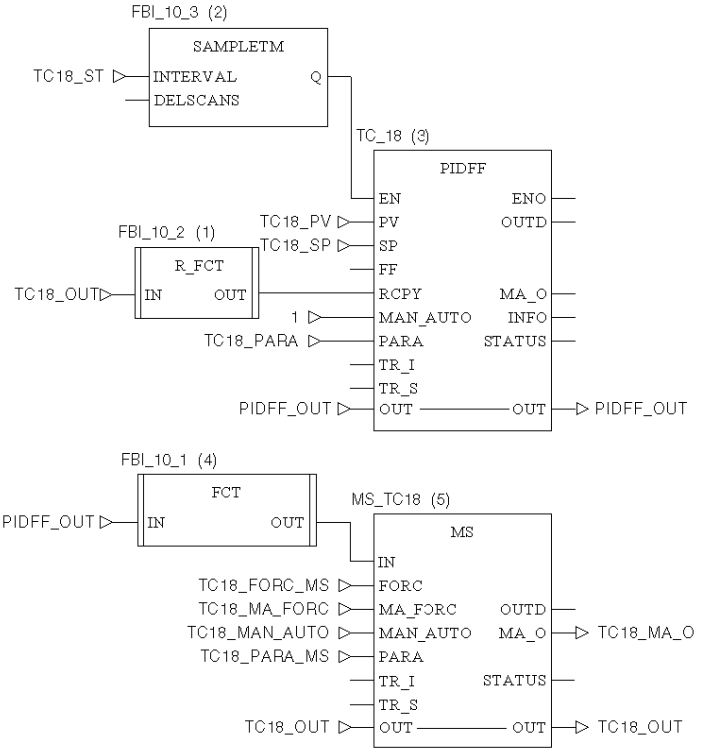
Example

Example

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

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

Display of the function plans



Runtime error

Status word

The bits of the status words have the following meaning:

Bit	Value in Dec.	Value in Hex.	ENO Status	Description
Bit 0 = 1	1	0x0001	False	Error in a calculation in floating point values
Bit 1 = 1	2	0x0002	False	Recording of an invalid value on one of the floating point value inputs
Bit 2 = 1	4	0x0004	False	Division by zero during a floating point value calculation
Bit 3 = 1	8	0x0008	False	Capacity overflow during a floating point value calculation
Bit 4 = 1	16	0x0010	True	The following error will be shown: ● One of the following variables is negative: `inc_rate`, `dec_rate`. For calculation, the function block uses the value 0. ● The parameter `outbias` lies out of the area $[(out\_min \quad out\_max), (out\_\max \quad out\_\min)]$. In this case the function block uses a capped value: $(out\_min \quad out\_max)$ or $(out\_\max \quad out\_\min)$.
Bit 5 = 1	32	0x0020	True	The output `OUT` has reached the lower threshold `out_min` (see Note)
Bit 6 = 1	64	0x0040	True	The output `OUT` has reached the upper threshold `out_max` (see Note)

For the list of other possible floating point error codes, see *Common Floating Point Errors, page 365*.

Note

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

Error message

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

NOTE: For a list of all block error codes and values, see *Output Processing, page 362*.

Warning Message

In the following cases a warning message is given:

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

Chapter 33
MS_DB: Manually controlling and output with dead zone

Introduction

This chapter describes the `MS_DB` block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	286
Detailed description	290
Runtime error	294

Description

Function description

This function block is used to split a digital output. A common usage is duplicating a PID output to an actuator.

A dead band can be applied to the deviation between the IN input and the positional copy.

The manual operating mode enables the control to operate with or without positional feedback (by feeding the output to the input, no deviation will occur on changeover to auto; therefore the start will be smooth).To switch smoothly from manual mode to auto mode in servo control, feed an increasing or decreasing ramp to the setpoint input.

EN and ENO can be configured as additional parameters.

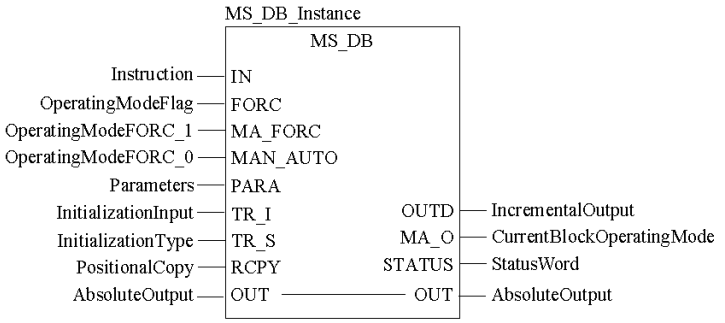
Application possibilities

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

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

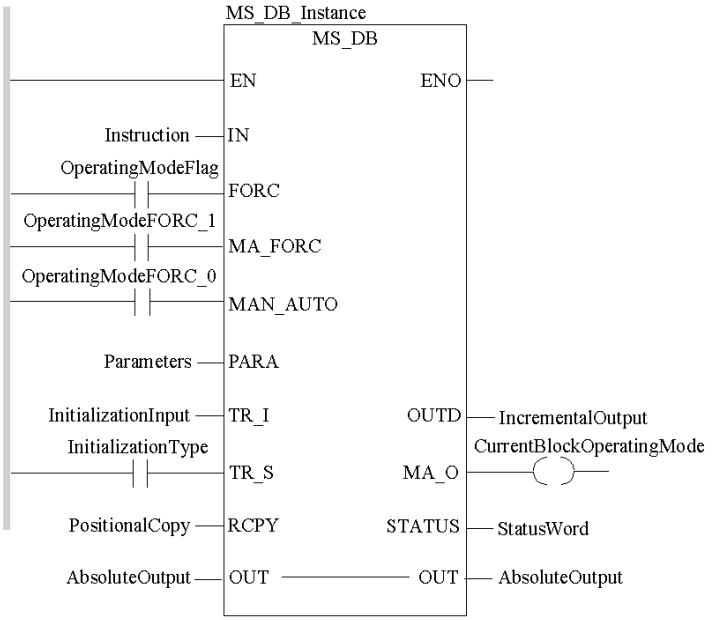
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL MS_Instance (IN:=Instruction, FORC:=OperatingModeFlag,
    MA_FORC:=OperatingModeFORC_1,
    MAN_AUTO:=OperatingModeFORC_0, PARA:=Parameters,
    TR_I:=InitializationInput, TR_S:=InitializationType,
    RCPY:= PositionalCopy,
    OUT:=AbsoluteOutput, OUTD=>IncrementalOutput,
    MA_O=>CurrentBlockOperatingMode, STATUS=>StatusWord)
```

Representation in ST

Representation:

```
MS_Instance (IN:=Instruction, FORC:=OperatingModeFlag,
    MA_FORC:=OperatingModeFORC_1,
    MAN_AUTO:=OperatingModeFORC_0, PARA:=Parameters,
    TR_I:=InitializationInput, TR_S:=InitializationType,
    RCPY:= PositionalCopy,
    OUT:=AbsoluteOutput, OUTD=>IncrementalOutput,
    MA_O=>CurrentBlockOperatingMode, STATUS=>StatusWord) ;
```

Parameter description MS

Input parameter description:

Parameter	Data type	Meaning
IN	REAL	Manipulated variable used in automatic mode
FORC	BOOL	"1": The mode manual/automatic will be entered via MA_FORC "0": The mode manual/automatic will be entered via MAN_AUTO
MA_FORC	BOOL	Mode manual/automatic (if FORC = 1) "1": Automatic operating mode "0": Manual mode
MAN_AUTO	BOOL	Mode manual/automatic (if FORC = 0) "1": Automatic operating mode "0": Manual mode
PARA	Para_MS_DB	Parameter
TR_I	REAL	Initialization input
TR_S	BOOL	Initialization command
RCPY	REAL	Positional feedback (0 to 100%)

Input / output parameter description:

Parameter	Data type	Meaning
OUT	REAL	Absolute output

Output parameter description:

Parameter	Data type	Meaning
OUTD	REAL	Incremental output: Difference between the present output and the output of the previous execution
MA_O	BOOL	Current mode of the function block (0: Manual, 1: Automatic)
STATUS	WORD	Status word

Parameter description Para_MS_DB

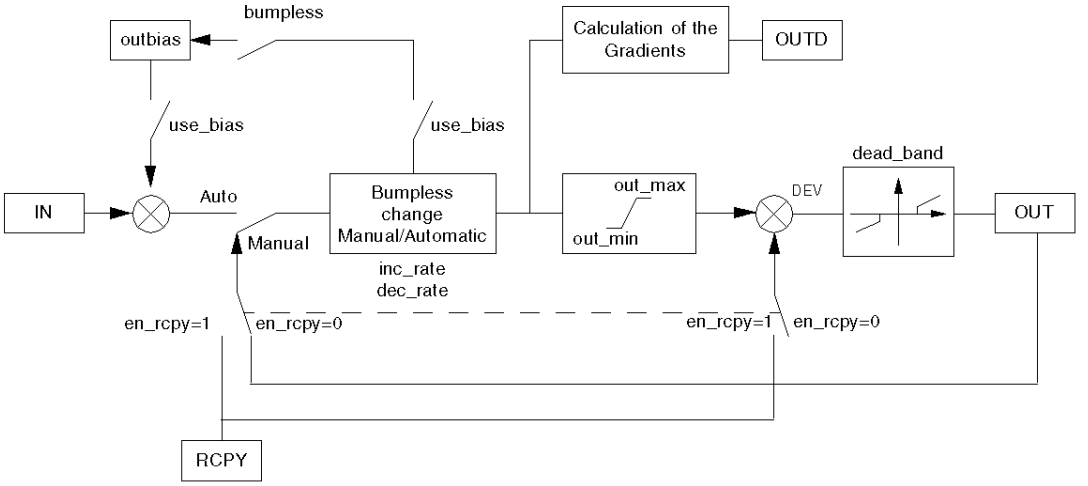
Data structure description

Element	Data type	Meaning
out_min	REAL	lower limit value of the output
out_max	REAL	upper limit value of the output
inc_rate	REAL	Increasing ramp at the changeover manual/automatic (units per second)
dec_rate	REAL	Decreasing ramp at the changeover manual/automatic (units per second)
outbias	REAL	Value of the bias
use_bias	BOOL	"1": Enable the bias
bumpless	BOOL	"1": Settings of the bias with changeover manual/automatic (bumpless)
dead_band	REAL	Dead zone on deviation
en_rcpy	BOOL	"1": Function with positional feedback (including RCPY)

Detailed description

Structure diagram

The following diagram displays the structure of the function block:



Setting of the mode selection

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

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

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

Characteristics of the output OUT

The following characteristics apply to the output OUT:

- Automatic mode: The output OUT is a copy of the input IN.
  In this operating mode, the output OUT can be assigned an OUTBIAS value (use _bias to 1).
  OUT calculates as follows: OUT = IN + outbias.
- Manual mode: The function block does not set the output, the server can directly change the value that is the connected variable at the output OUT.
- The output OUT is principally limited to an area between out_min and out_max . When the value calculated by the function block (or entered by the server in manual mode) exceeds one of these limit values, the value of OUT will be cut (to out_min or out_max).). The incremental output OUTD on the other hand, does not take this cut into consideration.

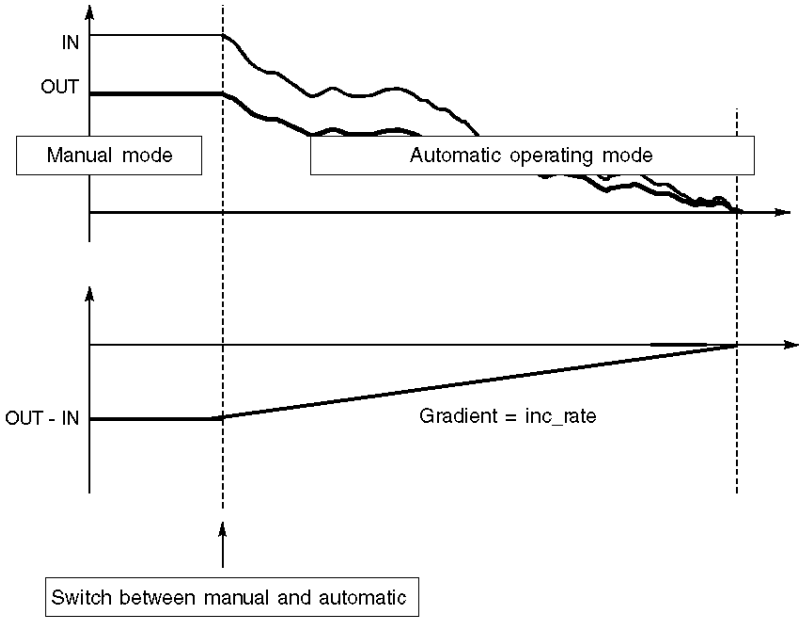
Switch between manual and automatic

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

The output OUT gets closer to input IN ramps with positive (inc_rate) or negative increase (dec_rate):

- inc_rate applies when IN is larger than OUT at the time of the changeover
- dec_rate applies when IN is less than OUT at the time of the changeover

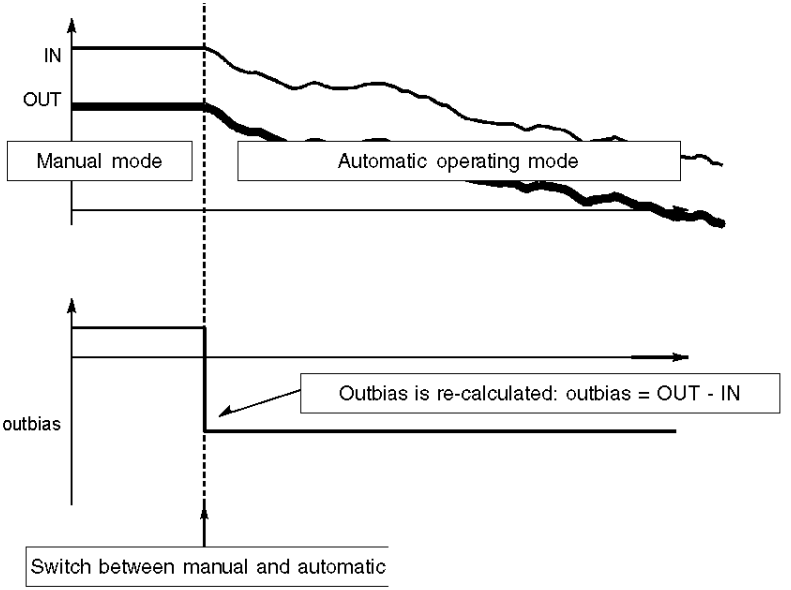
bumpless changeover



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

When the parameter Outbias (`use_bias` = 1) is used, a bumpless manual/automatic changeover can be achieved without change of the output, when the `bumpless` parameter is set to 1. In this case the parameter Outbias will be newly calculated and the deviation between the input `IN` and the output `OUT` will be considered.

Bumpless changeover with the parameter `Outbias`



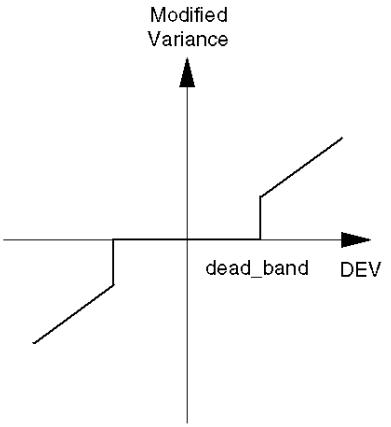
The bumpless changeover manual/automatic is advisable when the input of the function block is not connected to any controller or to a controller output without integral component.

Dead zone and Positional Feedback

The following characteristics apply:

- With manual and positional feedback (en_rcpy=1)
  The dead zone enables a comparison to be made between the OUT output that was calculated by the MS_DB function, and the value of the positional feedback (RCPY). If the result of the compare is less than the value of the dead zone (dead_band, the deviation of this actuator is ignored.
- With automatic and positional feedback (en_rcpy=1)
  The dead zone enables a comparision between the input value and the positional feedback value (RCPY). If the result of the compare is less than the value of the dead zone (dead_band, the deviation of this actuator is ignored.
- Without positional feedback (en_rcpy = 0)
  The dead zone has no function.

Display of dead zone on deviation (dead_band)



Runtime error

Status word (STATUS)

The bits of the status words have the following meaning:

Bit	Description
Bit 0 = 1	Error in a calculation in floating point values
Bit 1 = 1	Recording of an invalid value on one of the floating point value inputs
Bit 2 = 1	Division by zero during a floating point value calculation
Bit 3 = 1	Capacity overflow during a floating point value calculation
Bit 4 = 1	The following error will be shown: ● One of the following variables is negative: inc_rate, dec_rate. For calculation, the function block uses the value 0. ● The parameter Outbias lies out of the area $[(out\_min \quad out\_max), (out\_\max \quad out\_\min)]$. In this case the function block uses a capped value: $(out\_min \quad out\_max)$ or $(out\_\max \quad out\_\min)$.
Bit 5 = 1	The output OUT has reached the lower threshold out_min (see Note)
Bit 6 = 1	The output OUT has reached the upper threshold out_max (see Note)

For the list of other possible floating point error codes, see *Common Floating Point Errors, page 365*.

Note

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

Error message

An error is signaled if a non floating value is detected at an input or if there is a problem with a floating point calculation. In these cases the OUT, OUTD and MA_O outputs remain unchanged.

NOTE: For a list of all block error codes and values, see *Output Processing, page 362*.

Warning Message

In the following cases a warning message is given:

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

Chapter 34
PWM1: Pulse width modulation

Introduction

This chapter describes the PWM1 block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	298
Detailed description	301
Example of the PWM1 block	303

Description

Using the block

Actuators are driven not only by analog quantities, but also through binary actuating signals.

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

NOTE: This function block performs an internal initialization in the first program cycle after a warm start or cold start (e.g. application download or power cycle) of the PLC program.
Due to this, you have to make sure that the function block is invoked in the first program cycle. In case of inkoving the function block in a later program cycle, the internal initialization will not be performed and the ouputs may deliver wrong values.

⚠ DANGER

UNEXPECTED OUTPUT BEHAVIOUR

Make sure that the function block is always invoked in the first program cycle.

Failure to follow these instructions will result in death or serious injury.

Function description

The function block PWM1 converts analog values into digital output signals.

In pulse width modulation (QPWM), a 1-signal is emitted, at a constant clock rate, for a duration that is a function of the analog value. The adjusted average energy corresponds to the quotient of the switch on duration T_on and the cycle time t_period.

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

$$T\_on \sim IN$$

EN and ENO can be configured as additional parameters.

General information about the actuator drive

In general, the binary actuator drive is carried out by two binary signals OUT_POS and OUT_NEG. On a motor the output OUT_POS corresponds to the signal "clockwise rotation" and the output OUT_NEG the signal "counter-clockwise rotation". For an oven the outputs OUT_POS and OUT_NEG could be interpreted as "heating" and "cooling".

Pulse length formulas for OUT_POS and OUT_NEG

The pulse length T_on for output OUT_pos and OUT_neg is determined by the following formulas:

Output	Formula	Condition
OUT_POS	$T\_on = t\_period \times \frac{IN}{in\_\max}$	$0 \le IN \le in\_\max$
OUT_NEG	$T\_on = t\_period \times \frac{\lvert IN \rvert}{in\_\max}$	$0 \le \lvert IN \rvert \le in\_\max$

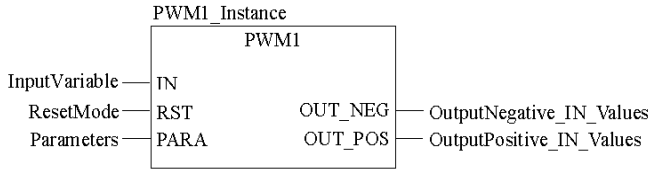
Parametering rules

For correct operation when setting parameters the following rules should be observed:

t_min ≤t_period

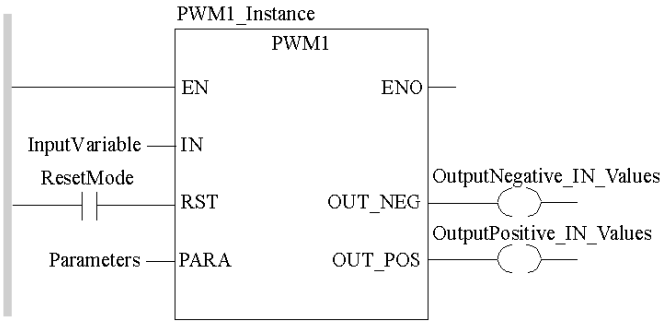
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL PWM1_Instance (IN:=InputVariable, RST:=ResetMode,
    PARA:=Parameters, OUT_NEG=>OutputNegative_IN_Values,
    OUT_POS=>OutputPositive_IN_Values)
```

Representation in ST

Representation:

```
PWM1_Instance (IN:=InputVariable, RST:=ResetMode,
    PARA:=Parameters, OUT_NEG=>OutputNegative_IN_Values,
    OUT_POS=>OutputPositive_IN_Values) ;
```

PWM1 parameter description

Description of input parameters:

Parameter	Data type	Description
IN	REAL	Input variable
RST	BOOL	Reset mode ("1" = Reset)
PARA	Para_PWM1	Parameter

Description of output parameters:

Parameter	Data type	Description
OUT_NEG	BOOL	Output for negative IN values
OUT_POS	BOOL	Output for positive IN values

Parameter description Para_PWM1

Data structure description

Element	Data type	Description
t_period	TIME	Length of period
t_min	TIME	Minimum actuating pulse time (in sec)
in_max	REAL	Upper limiting value for positive/negative IN values

Runtime error

NOTE: For a list of all block error codes and values, see *Output Processing*, *page 362*.

Detailed description

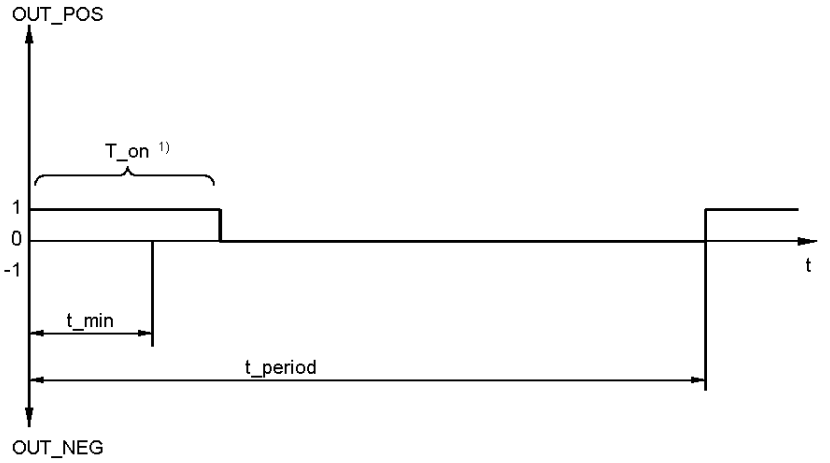
Block mode of operation

The period duration determines the time during which the actuating pulses (1-signals at the output OUT_POS or OUT_NEG) are output at regular intervals, i.e. within a constant time pattern.

The parameter t_min specifies the minimum pulse length, i.e. the time span in which the output OUT_POS or OUT_NEG should carry a "1" signal. If the length of the pulse calculated according to the equation in section *Pulse length formulas for OUT_POS and OUT_NEG, page 299* are smaller than t_min, no pulse will be given for the entire period.

Time ratios display

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

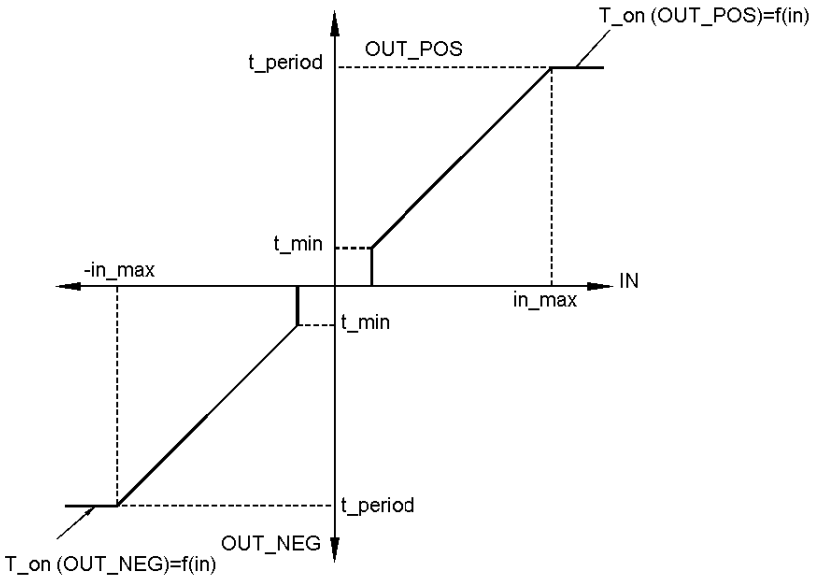


1 Variable turn-on time

The parameter in_max marks the point of the input variable IN, for which the output OUT_POS would continuously carry "1" if the IN input variable is positive.

Time-span dependency

The dependency of the time duration in which the output `OUT_POS` (`OUT_NEG`) carries a 1-Signal, on the input variable `IN` is illustrated in the following diagram:



Operating mode

In reset mode `RST` = 1, the outputs `OUT_POS` and `OUT_NEG` are set to "0". The internal time counters are standardized as well so that the function block begins its transition to `RST=0` with the output of a new 1-signal on the associated output.

Boundary conditions

If the `PWM1` block is operated together with a PID controller, then the period `t_period` should be so selected, that it corresponds to the PID controller's scan time. It is then guaranteed that every new actuating signal from the PID controller within the period time can be fully processed.

The `PWM1` scan time should be selected according to how the pulse time compares to the period length. Though this, the smallest possible actuating pulse is be determined.

The following ratio is recommended:

$$\frac{t\_period}{Abtastzeit\ (PWM1)} \geq 10$$

Example of the PWM1 block

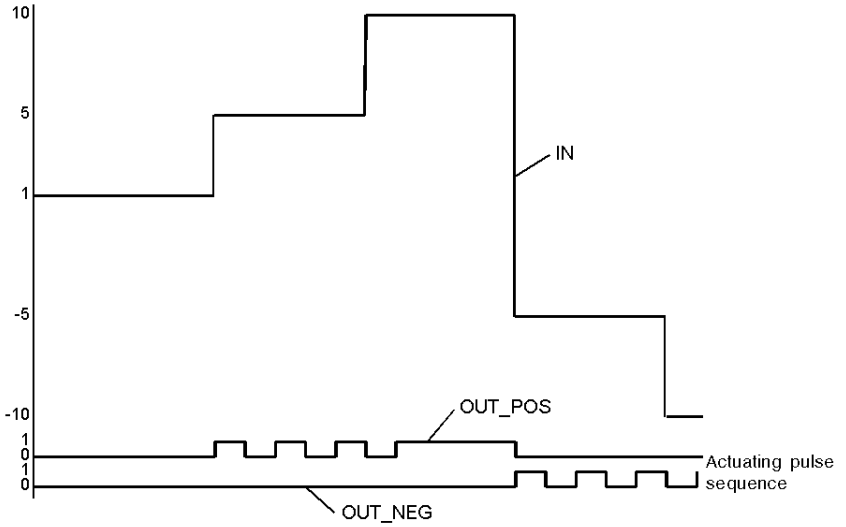
Jump response

In the examples, the signal sequences on the outputs `OUT_POS` and `OUT_NEG` are shown for various `IN` input signal values.

The following parameter specifications apply to the jump response display:

Parameter	Specification
`t_period`	4 s
`t_min`	0.5 s
`in_max`	10

Step response timing diagram



IN Analog signal

It can be seen that pulses are no longer output for very small `IN` input signals. This is directly attributable to the effect of time `t_min`. A continuous pulse is output for large `IN` (`IN` = `in_max`) signals.

Chapter 35
SERVO: Control for servo motors

Introduction

This chapter describes the SERVO block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	306
Parametering	310
SERVO function block algorithms	312
Operating mode	313
Examples of function block SERVO	314
Runtime error	321

Description

Function description

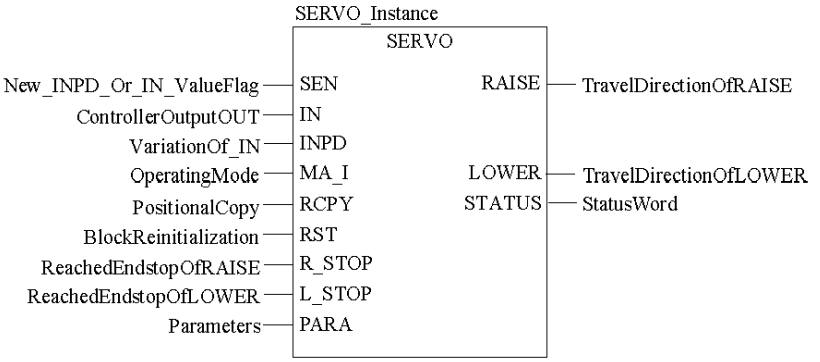
This function block enables PID control of servo motors with or without positional feedback. The function block can be switched to be the controller (`PIDFF`, `PI_B`) so that the digital outputs become the two logical outputs `RAISE` and `LOWER`.

If the function block uses positional feedback, then positioning controlling of the actuator will be performed. If positional feedback is not being used, the controller and the `SERVO` function block operate a continuous static control together.

`EN` and `ENO` can be configured as additional parameters.

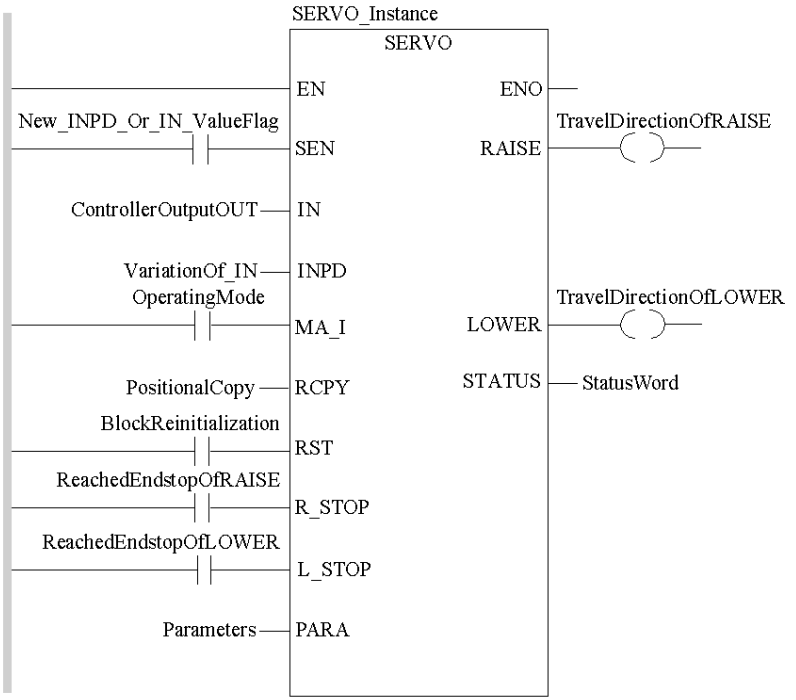
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL SERVO_Instance (SEN:=New_INPD_Or_IN_ValueFlag,
    IN:=ControllerOutputOUT, INPD:=VariationOf_IN,
    MA_I:=OperatingMode, RCPY:=PositionalCopy,
    RST:=BlockReinitialization,
    R_STOP:=ReachedEndstopOfRAISE,
    L_STOP:=ReachedEndstopOfLOWER, PARA:=Parameters,
    RAISE=>TravelDirectionOfRAISE,
    LOWER=>TravelDirectionOfLOWER, STATUS=>StatusWord)
```

Representation in ST

Representation:

```
SERVO_Instance (SEN:=New_INPD_Or_IN_ValueFlag,
    IN:=ControllerOutputOUT, INPD:=VariationOf_IN,
    MA_I:=OperatingMode, RCPY:=PositionalCopy,
    RST:=BlockReinitialization,
    R_STOP:=ReachedEndstopOfRAISE,
    L_STOP:=ReachedEndstopOfLOWER, PARA:=Parameters,
    RAISE=>TravelDirectionOfRAISE,
    LOWER=>TravelDirectionOfLOWER, STATUS=>StatusWord) ;
```

Parameter description SERVO

Input parameter description:

Parameter	Data type	Meaning
SEN	BOOL	"1" : Including a new value at the INPD or IN inputs "0" : no inclusion of the new values of INPD or IN
IN	REAL	Control output OUT (0 to 100%)
INPD	REAL	Output alteration OUTD of the controller (-100% to 100%)
MA_I	BOOL	Control operating mode (Output MA_O) "1" : Automatic mode "0" : Manual or tracking mode
RCPY	REAL	Positional feedback (0 to 100%)
RST	BOOL	"1" : Reinitialization of the function block (resetting outputs and the internal function block status)
R_STOP	BOOL	End position RAISE reached
L_STOP	BOOL	End position LOWER reached
PARA	Para_SERVO	Parameter

Output parameter description:

Parameter	Data type	Meaning
RAISE	BOOL	Logical output in the direction RAISE
LOWER	BOOL	Logical output in the direction LOWER
STATUS	WORD	Status word

Parameter description Para_SERVO

Data structure description

Element	Data type	Meaning
en_rcpy	BOOL	"1" : Function with positional feedback (including RCPY)
rcpy_rev	BOOL	"1" : Inversion of RCPY "0" : no inversion of RCPY
t_motor	TIME	Actuator opening time
t_mini	TIME	Minimum impulse length

Parametering

Parametering overview

The following function block modes are explained in sequence:

- *With positional feedback (en_rcpy = 1), page 310*
- *Without positional feedback (en_rcpy = 0), page 310*
- *Actuator opening time (t_motor), page 310*
- *Minimum impulse length (t_mini), page 310*
- *Sweep / parameter SEN, page 311*
- *Recording the end position, page 311*

With positional feedback (en_rcpy = 1)

If the positional feedback RCPY (en_rcpy = 1) is used, the input IN must be attached to the absolute value output OUT of a controller (control range 0 to 100%). For each new value for output OUT generated by the controller the SERVO function block generates a discrete output RAISE or LOWER whose length is proportional to the variance IN - RCPY. To guarantee that the function block operates correctly, the input MA_I must be attached to the controller's MA_O output.

The RCPY input value can correspond to an opening percentage (with rcpy_rev = 0) or a closing percentage (rcpy_rev set to 1).

Without positional feedback (en_rcpy = 0)

If no positional feedback is assigned (en_rcpy = 0) the INPD input should be attached to a controller's output alteration OUTD (control range -100 to 100%). For each new OUTD value generated by the controller, the function block SERVO generates a discrete output RAISE or LOWER whose length is proportional to the output length of the controller INPD. In this case it is essential that the input MA_I is attached to the same controller's MA_O output because the algorithm varies slightly for each operating mode (see section "*SERVO function block algorithms, page 312*").

Actuator opening time (t_motor)

The parameter t_motor enables the function block to be set to the various servomotors.

The RAISE or LOWER pulse duration to be switched must be proportional to the actuator opening time with full control range.

Minimum impulse length (t_mini)

Use the t_mini parameter to avoid generation of pulses which are too short and can damage the actuator. If the RAISE or LOWER pulse length is calculated to be below t_mini the function block does not generate a pulse. Every pulse which has already commenced lasts at least t_mini.

Sweep / parameter SEN

In automatic mode the resolution of the control performed using the SERVO function block is expressed by the ratio (servoloop sampling period / SERVO function block execution period).

This means the controller must be sampled before the SERVO function block (using a SAMPLETM function block). The SERVO function block must, however, be executed every cycle. In the opposite case (if the control block is executed at the same time as the SERVO block) an inexact two point control, which the actuator makes great use of, is performed.

The SEN input of the SERVO function block indicates whether or not the PID control block was executed while the cycle was running.

The SEN input allows determination of whether or not the controller generated a new output so that the same output is not considered several times.

SEN =	Meaning
1	Including a new value
0	no inclusion of a new value

If the controller samples using the function block SAMPLETM, as is the usual case, it suffices to attach the SERVO block's SEN input to the SAMPLETM output (see section "*Examples of function block SERVO, page 314*").

Recording the end position

If an end position is gathered (R_STOP = 1 or L_STOP = 1), the corresponding output (RAISE or LOWER) is forced to 0.

SERVO function block algorithms

Algorithm without positional feedback

In this case the SERVO function block assigned to the controller allows astatic control. The algorithm uses the output alteration OUTD rather than the controller's absolute value output OUT. The output RAISE (or LOWER, depending on the modification sign) is set to 1 for a certain time. This time is proportional to the valve opening time (t_motor) and the modification value OUTD.

The formula enters an initial theoretical value for the length of the pulse (T_IMP) to be sent to the output:

$$T\_IMP = OUTD(\%).t\_motor$$

The following still applies for T_IMP (the length of the pulse sent to the output):

If	Then
T_IMP < t_mini	the block does not generate a pulse, but stores the value for the next calculation. This allows correct processing of control applications in which the controller's output modifications are weak but continuous. To ensure that pulses which are too short are not generated, the pulses to be sent to the output are limited to a minimum length t_mini.
the PID controller is in manual mode,	T_IMP is calculated continuously at every cycle. The calculation takes into consideration the time periods with a limit of t_motor which have previously been calculated, but not yet assigned. In this way any PID controller output modification can be considered even if the pulse lasts several cycles.
the PID controller is in automatic mode,	the function block SERVO always recalculates the parameter T_IMP if the controller updates its output, i.e. whenever SEN is set to 1. In this operating mode the previously calculated time periods are no longer considered.

Algorithm with positional feedback

The algorithm is very similar to the previous case.

In place of the PID controller output modification the SERVO function block uses the variance between the PID controller absolute value output and the positional feedback (IN - RCPY).

The function block carries out controlling, in which the PID controller output corresponds to the nominal value and the positional feedback RCPY to the actual value.

In contrast to the algorithm without positional feedback, in manual mode the function block stores the time periods, which were calculated previously, but are not yet locked onto the RAISE and LOWER outputs.

Operating mode

Operating mode adjustment

The input MA_I allows the SERVO function block to adjust to the controller's operating mode. To do this it must be attached to the output MA_O of the controller or the corresponding MS function block.

Automatic operating mode

The function block SERVO only rereads the control output if this has been updated (i.e. whenever SEN is set to 1).

Manual mode

The user can modify the control output here at any time. In order that a new value can be included as soon as possible, the function block reads the control output at every cycle.

In this operating mode the user can manually modify variables connected to the OUT output of a controller or a MS block. If no positional feedback is used this variable can adopt the end position (100% or 0%) even if the actuator has not reached either of its end positions. It is still possible to modify the output modification OUTD manually by setting the output OUT of the function block MS to more than 100% (or to less than 0%). The value inputted for OUT is used for the calculation of OUTD before it is limited again.

Examples of function block SERVO

Example overview

In this section the use of the function block SERVO is shown in the following examples:

- *Automatic mode with positional feedback, page 314*
- *Example of operating mode automatic without positional feedback in manual mode, page 317*

Automatic mode with positional feedback

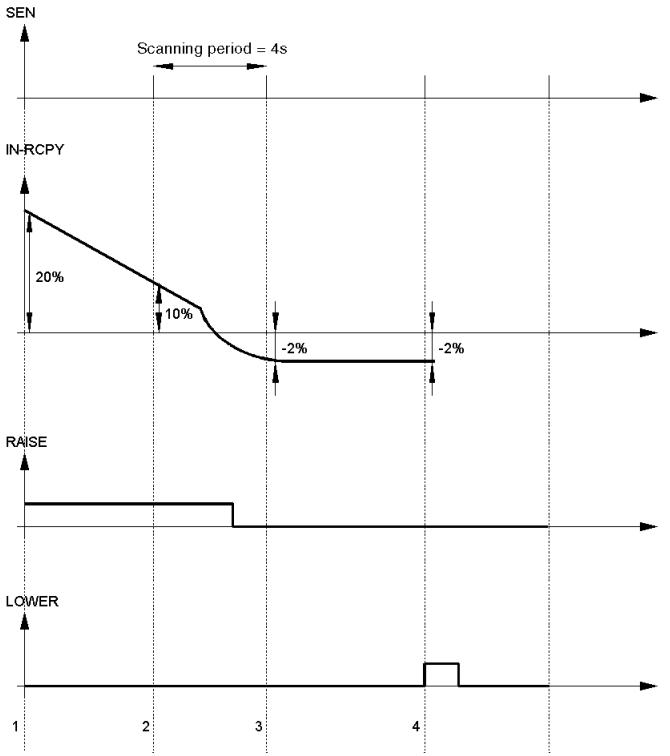
The example shows the behavior of the function block in automatic mode with positional feedback. If the SEN input is set to 1 (every 4 s in the example), the function block SERVO always takes a new variance value IN-RCPY into account.

The following parameter specifications hold:

Parameter	Specification
t_motor	25 s
t_mini	1 s
sampling period	4 s

Timing diagram (automatic with positional feedback)

Timing diagram for automatic mode with positional feedback



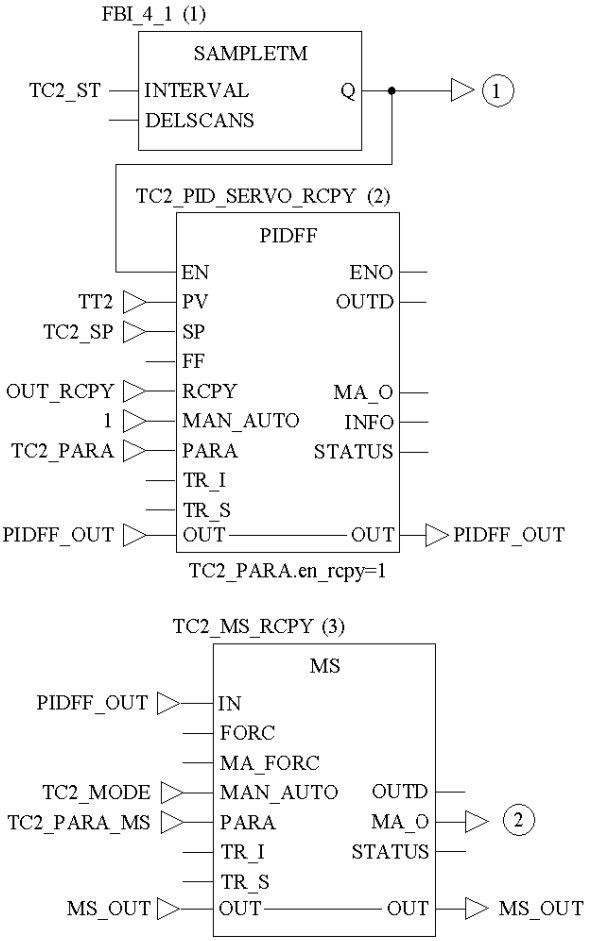
Explanation of the timings

Explanation of the marked positions:

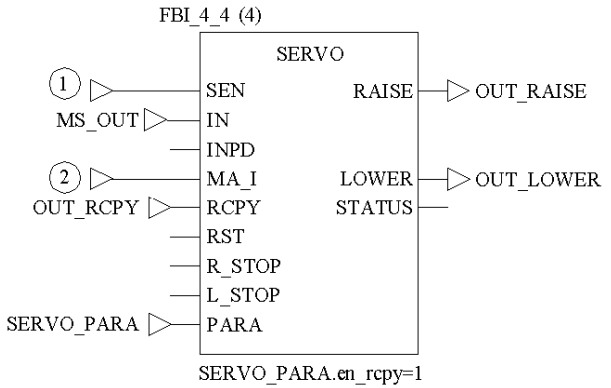
Position No.	Explanation
1	The variance IN-RCPY is 20%: a pulse of length 5 s (=20% of 25 s) was generated at the RAISE output.
2	The variance is still only 10%, a pulse of 2.5 s (= 10% of 25 s) was generated at the RAISE output; the second left over from the previous pulse is not taken into account.
3	The deviation now amounts to -2%, which corresponds to a pulse of 0.5 s on LOWER. Since t_mini is 1 s, no pulse is generated (the duration of 0.5 s is saved however).
4	The variance is still -2%, but the corresponding pulse (0.5 s) is added to the previously stored pulse to make 1 s. The length corresponds to t_mini, so the pulse is locked onto the LOWER output.

Programming example (automatic with positional feedback)

Representation the function plan, part 1



Representation the function plan, part 2



OUT_RCPY Process value of the valve positional feedback

Example of operating mode automatic without positional feedback in manual mode

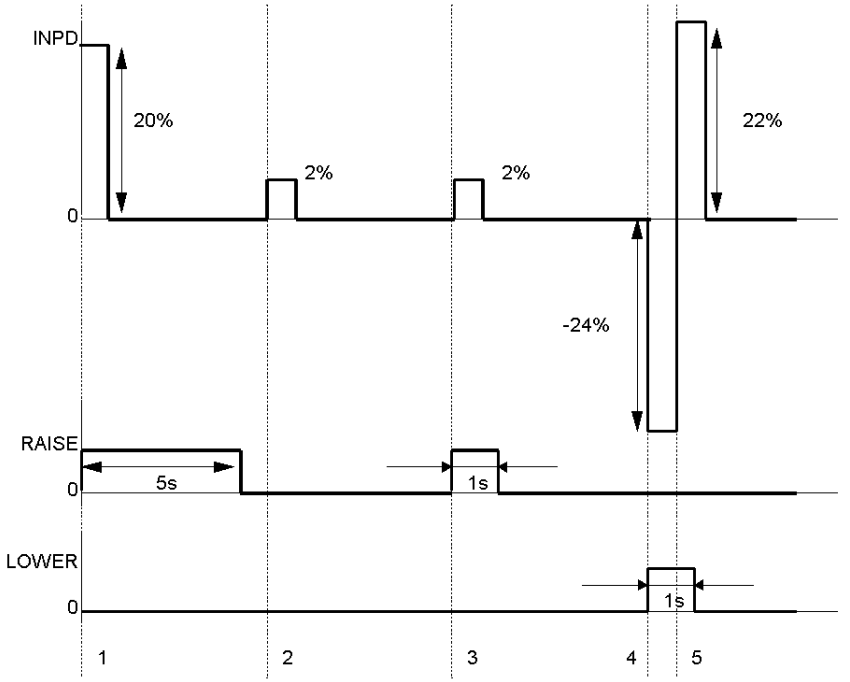
The example shows the behavior of the function block in automatic operating mode without positional feedback in manual mode. In this case the `INPD` value for each execution of the function block `SERVO` is taken into account, irrespective of the value of the `SEN` input.

The following parameter specifications hold:

Parameter	Specification
`t_motor`	25 s
`t_mini`	1 s

Timing diagram (automatic without positional feedback)

Automatic mode without positional feedback in manual mode



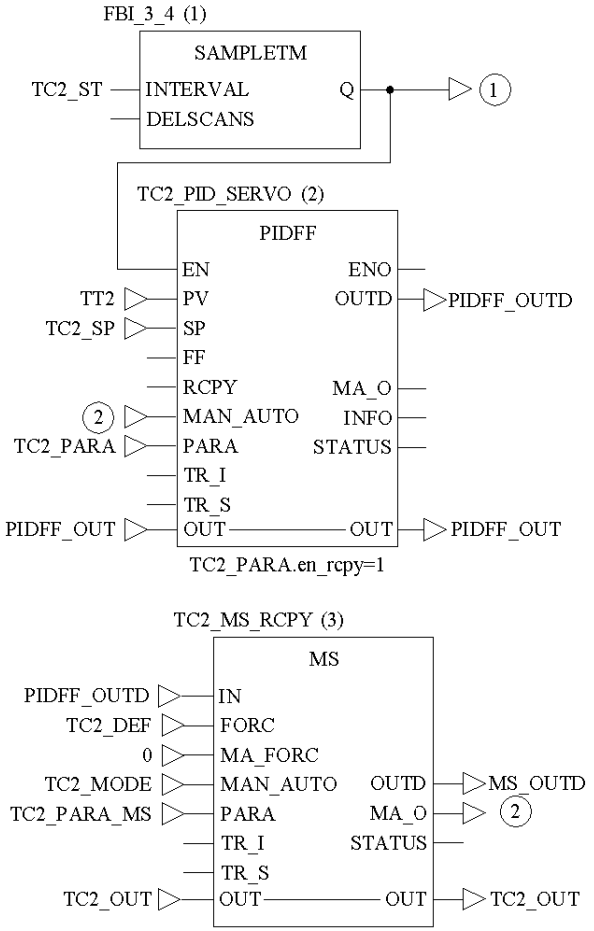
Explanation of the timings

Explanation of the marked positions:

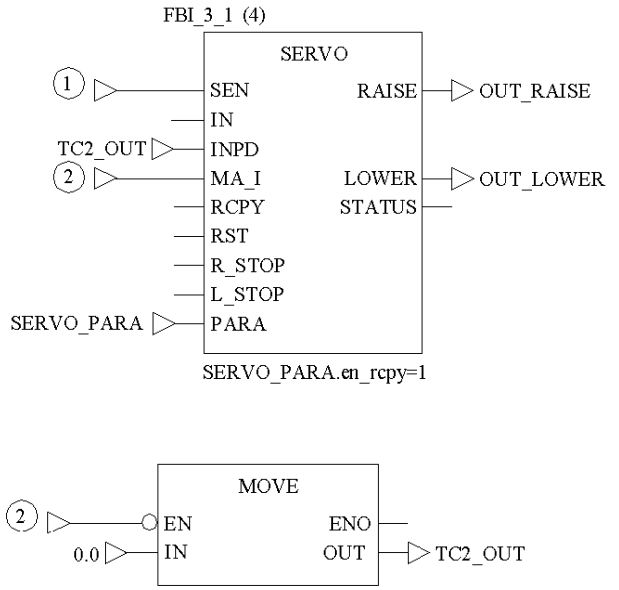
Position No.	Explanation
1	The modification of the PID control output is +20%, in this case the pulse affects the `RAISE` output and lasts 5 s (= 20% of 25 s).
2	The modification of the PID controller is +2% which corresponds to a pulse duration of 0.5 s. The pulse is less than `t_mini` (=1 s.) so it does not influence the outputs.
3	At the second modification of +2 %, the function adds this modification to the previous one (which corresponds to a variance which was below the minimum value), which corresponds to a positive total modification of +4 %, i.e. a pulse of 1 s at the `RAISE` output.
4	For a modification of -24 % the pulse at the `LOWER` output is 6 s
5	Before the end of the following second a further modification of + 22 % leads to a total system modification of 2 %< modification of `t_mini` (4 %). The function ends with the minimum pulse of 1 s.

Programming example (automatic without positional feedback)

Representation of the function plan, part 1



Representation of the function plan, part 2



TC2_DEFF Error output of the process value TC2: If TC2 is faulty, the servoloop is forced into manual mode.

Runtime error

Status word

The following messages are displayed in the Status word:

Bit	Value in Dec.	Value in Hex.	ENO Status	Description
Bit 0 = 1	1	0x0001	False	Error in a floating point value calculation
Bit 1 = 1	2	0x0002	False	Recording of an invalid value on one of the floating point value inputs
Bit 2 = 1	4	0x0004	False	Division by zero during a floating point value calculation
Bit 3 = 1	8	0x0008	False	Capacity overflow during floating point value calculation
Bit 4 = 1	16	0x0010	True	`IN` or `RCPY` do not lie in the range [0, 100] or `INPD` lies outside the range [-100, 100]. To calculate the function block uses a value that is limited by the next closest correct value, i.e. 0, 100 or –100, depending on the value.

For the list of other possible floating point error codes, see *Common Floating Point Errors, page 365*.

Error message

An error appears if a non floating point value is recorded or if there is a problem with a floating point calculation. In this case the outputs `RAISE` and `LOWER` are set to zero.

NOTE: For a list of all block error codes and values, see *Output Processing, page 362*.

Chapter 36
SPLRG: Controlling 2 actuators

Introduction

This chapter describes the SPLRG block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	324
Detailed description	326
Runtime error	328

Description

Function description

This function block should be used when two actuators are in use to enable coverage of the whole area (when two operating points are far apart: one below and one above).

The controller is also suitable for three-point step action controls, i.e. for cases where the two actuators work in opposition (one heats, the other cools).

`EN` and `ENO` can be configured as additional parameters.

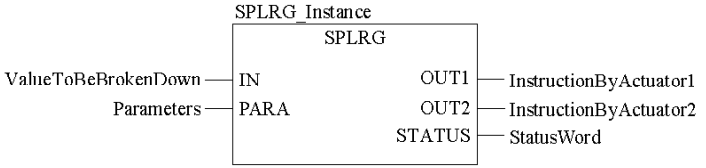
Properties

The function block `SPLRG` has the following properties:

- The possibility of controlling a dead zone or a transition zone where the properties of both actuators are in line
- The `IN` input is expressed as a percentage (0-100%) and the outputs `OUT1` and `OUT2` are expressed in physical units.

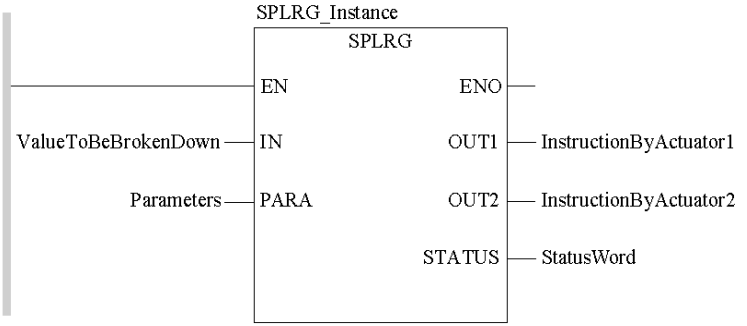
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL SPLRG_Instance (IN:=ValueToBeBrokenDown,
    PARA:=Parameters, OUT1=>InstructionByActuator1,
    OUT2=>InstructionByActuator2, STATUS=>StatusWord)
```

Representation in ST

Representation:

```
SPLRG_Instance (IN:=ValueToBeBrokenDown,
    PARA:=Parameters, OUT1=>InstructionByActuator1,
    OUT2=>InstructionByActuator2, STATUS=>StatusWord) ;
```

Parameter description SPLRG

Input parameter description:

Parameter	Data type	Meaning
IN	REAL	Value to be resolved (0 to 100%)
PARA	Para_SPLRG	Parameter

Output parameter description:

Parameter	Data type	Meaning
OUT1	REAL	Manipulated variable for actuator 1
OUT2	REAL	Manipulated variable for actuator 2
STATUS	WORD	Status word

Parameter description Para_SPLRG

Data structure description

Element	Data type	Meaning
out1_th1	REAL	Input value IN, for which the following applies: OUT1 = out1_inf
out1_th2	REAL	Input value IN, for which the following applies: OUT1 = out1_sup
out1_inf	REAL	Lower threshold of the output OUT1
out1_sup	REAL	Upper threshold of the output OUT1
out2_th1	REAL	Input value IN, for which the following applies: OUT2 = out2 _inf
out2_th2	REAL	Input value IN, for which the following applies: OUT2 = out2_sup
out2_inf	REAL	Lower threshold for output OUT2
out2_sup	REAL	Upper threshold for output OUT2

Detailed description

Parametering

Parametering the function block consists of defining the properties of each actuator, i.e. in the kind of gradient modification of both control outputs in relation to the input IN.

The following points should be defined for the output OUT1:

Element	Meaning
out1_inf	Lower threshold range
out1_sup	Upper threshold range
out1_th1	Threshold value, i.e the input value IN, for which the following applies: Output OUT1 = out1_inf
out1_th2	Threshold value, i.e the input value IN, for which the following applies: Output OUT1 = out1_sup

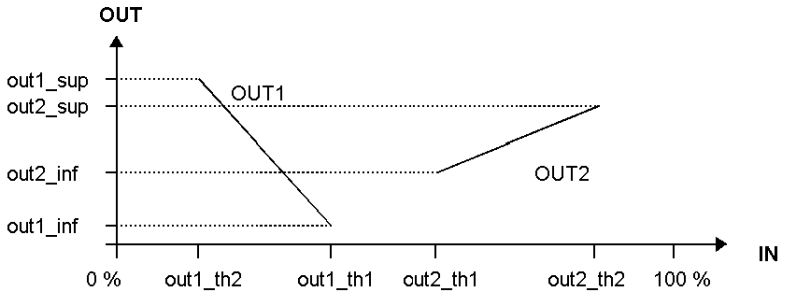
The modification of the value of OUT1 is linear for both threshold values. With the exception of the two threshold values, no further output modification can occur; it is limited to out1_inf or out1_sup.

Depending on the adjustment of the two threshold values, the control properties are designated by a positive increase (for out1_th1 < out1_th2) or a negative increase (for out1_th2 < out1_th1).

The following diagrams show the properties of the two actuators with Split range and Three-point step-action control.

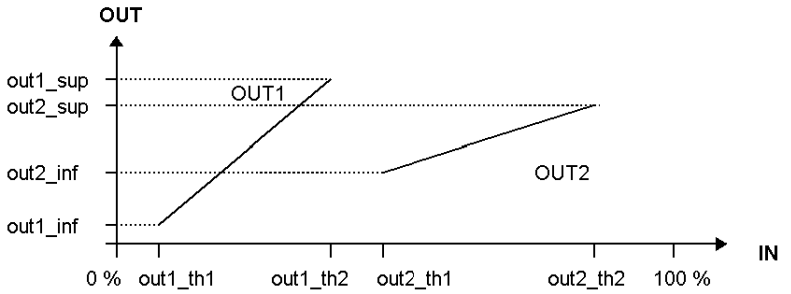
Three step step-control

The following shows the properties of the two actuators in three-point step-control



Split range control

The following shows the properties of the two actuators in split range control



NOTE: The outputs of this controller cannot be used to control a SERVO function block without positional feedback.

Operating mode

The SPLRG function block is not assigned to any specific operating mode. However both function block outputs may be controlled manually because an MS is locked on to each output. During programming the user should ensure a bumpless return to automatic mode.

Runtime error

Status word

The following messages are displayed in the Status word:

Bit	Value in Dec.	Value in Hex.	ENO Status	Description
Bit 0 = 1	1	0x0001	False	Error in a floating point value calculation
Bit 1 = 1	2	0x0002	False	Recording of an invalid value on one of the floating point value inputs
Bit 2 = 1	4	0x0004	False	Division by zero during a floating point value calculation
Bit 3 = 1	8	0x0008	False	Capacity overflow during floating point value calculation
Bit 4 = 1	16	0x0010	True	`IN` or one of the parameters `out1_th1`, `out1_th2`, `out2_th1`, `out2_th2` is not in the range [0, 100]: for calculation, the function block uses the value 0 or 100.
Bit 5 = 1	32	0x0020	True	The output `OUT1` has reached the lower threshold `out1_inf`: `OUT1` is forced to `out1_inf`.
Bit 6 = 1	64	0x0040	True	The output `OUT1` has reached the upper threshold `out1_sup`: `OUT1` is forced to `out1_sup`.
Bit 7 = 1	128	0x0080	False	Both the threshold values of an output are identical: `out1_th1` = `out1_th2`, `out2_th1` = `out2_th2`.
Bit 8 = 1	256	0x0100	True	The output `OUT2` has reached the lower threshold `out2_inf`: `OUT2` is forced to `out2_inf`.
Bit 9 = 1	516	0x0200	True	The output `OUT2` has reached the upper threshold `out2_sup`: `OUT2` is forced to `out2_sup`.

For the list of other possible floating point error codes, see *Common Floating Point Errors, page 365*.

Error message

A runtime error appears in the following cases:

- A non-floating value is on an input
- A problem has occurred during a floating point value calculation.
- Both the thresholds of the same output are identical: `out1_th1` = `out1_th2` or `out2_th1` = `out2_th2`

The outputs `OUT1` and `OUT2` are never modified.

NOTE: For a list of all block error codes and values, see*Output Processing, page 362*.

Warning

A warning is given if one of the parameters `out1_th1`, `out1_th2`, `out2_th1`, `out2_th2` is not in the [0 - 100] range. In this case the function block uses the value 0 or 100 for calculating.

Part VII
Setpoint management

Overview

This section describes the elementary functions and elementary function blocks of the `Setpoint management` family.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
37	RAMP: Ramp generator	333
38	RATIO: Ratio controller	339
39	SP_SEL: Setpoint switch	347

Chapter 37
RAMP: Ramp generator

Introduction

This chapter describes the RAMP block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	334
Detailed description	336
Runtime error	338

Description

Function description

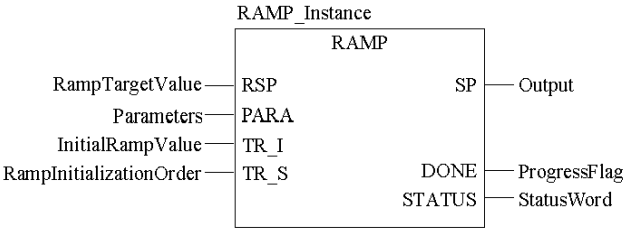
The function block RAMP makes it possible to move in ramp form from an initial setpoint value to a particular target value. The gradients of positive and negative ramps can vary.

A signal (DONE output) indicates the user, whether a target value has already been reached or if the ramp had been implemented.

EN and ENO can be configured as additional parameters.

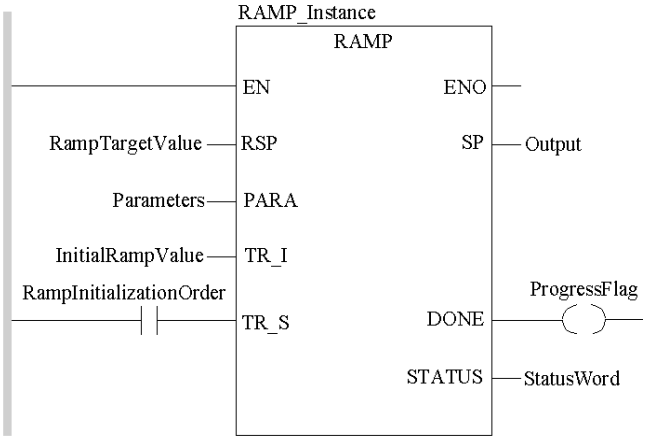
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL RAMP_Instance (RSP:=RampTargetValue, PARA:=Parameters,
    TR_I:=InitialRampValue, TR_S:=RampInitializationOrder,
    SP=>Output, DONE=>ProgressFlag, STATUS=>StatusWord)
```

Representation in ST

Representation:

```
RAMP_Instance (RSP:=RampTargetValue, PARA:=Parameters,
    TR_I:=InitialRampValue, TR_S:=RampInitializationOrder,
    SP=>Output, DONE=>ProgressFlag, STATUS=>StatusWord) ;
```

Parameter description RAMP

Input parameter description:

Parameter	Data type	Meaning
RSP	REAL	Target value of the ramp
PARA	Para_RAMP	Parameter
TR_I	REAL	Initial value of the ramp
TR_S	BOOL	Initialization command of the ramp

Output parameter description:

Parameter	Data type	Meaning
SP	REAL	Output
DONE	BOOL	"1": the target value has been reached "0": the ramp function has been executed
STATUS	WORD	Status word

Parameter description Para_RAMP

Data structure description

Element	Data type	Meaning
inc_rate	REAL	Positive gradient in units per second (≥ 0)
dec_rate	REAL	Negative gradient in units per second (≥ 0)

Detailed description

Parametering

If the value given on input (RSP) exceeds the current value of the SP_output, the function block increases the value of the output with the velocity inc_rate by as much as is necessary for the SP value to reach the RSP value. If the inc_rate is zero, the ramp function will not be executed and the SP is identical to the RSP.

If the value given at an input falls below the current value of SP, the function block lowers the value of SP with the velocity dec_rate. If the dec_rate is zero, the ramp function will not be executed and the SP is identical to the RSP.

If the value of RSP changes whilst the ramp is being generated, the function block immediately attempts to reach this new target value. The ramp function, which is running simultaneously, either continues or changes its direction.

Operating mode

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

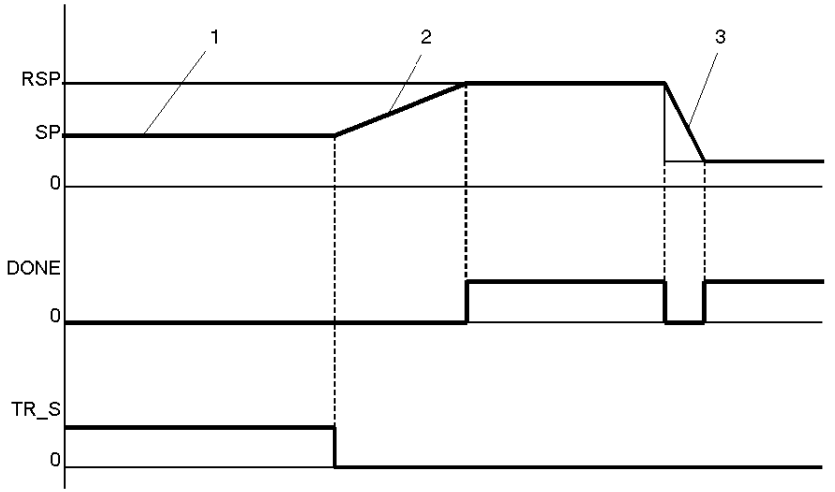
Step	Action
1	TR_I set to the desired initial value.
2	When TR_S is set to 1, the TR_I input will continue to be executed at SP. **Note:** In the tracking mode (TR_S = 1) the DONE output remains permanently at zero.
3	If TR_S is set to 0, the function block resumes normal operation: The SP constantly approaches the RSP, where the value describes a ramp.

DONE display

The DONE output goes above 1, if a ramp function has just been completed. It will be reset to zero, when a new ramp begins or when the function block is switched to tracking mode.

Timing diagram

Timing diagram for RAMP block



1 Initialization: SP = TR_I
2 Increasing ramp = inc_rate
3 Decreasing ramp = dec_rate

Runtime error

Status word

The following messages are displayed in the Status word:

Bit	Value in Dec.	Value in Hex.	ENO Status	Description
Bit 0 = 1	1	0x0001	False	Error in a floating point value calculation
Bit 1 = 1	2	0x0002	False	Recording of an invalid value on one of the floating point value inputs
Bit 2 = 1	4	0x0004	False	Division by zero during a floating point value calculation
Bit 3 = 1	8	0x0008	False	Capacity overflow for a calculation with floating point values
Bit 4 = 1	16	0x0010	True	One of the following variables is negative: `inc_rate`, `dec_rate`. For calculation, the function block uses the value 0.

For the list of other possible floating point error codes, see *Common Floating Point Errors, page 365*.

Error message

An error is signaled if a non floating value is recorded or if there is a problem with a floating point calculation. In this case the outputs `SP` and `DONE` remain unchanged.

NOTE: For a list of all block error codes and values, see *Setpoint Management, page 363*.

Warning

A warning appears in the following cases:

- The parameter `inc_rate` is negative: the function block uses the value 0 instead of the faulty value of `inc_rate`.
- The parameter `dec_rate` is negative: the function block uses the value 0 instead of the faulty value of `dec_rate`.

Chapter 38
RATIO: Ratio controller

Introduction

This chapter describes the RATIO block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	340
Detailed description	343
Runtime error	345

Description

Function description

The function block RATIO carries out ratio control when it is attached to a controller.

The aim of ratio control is to establish a ratio of one process variable PV (controlled variable) to another PV_TRACK (reference variable). The role of the RATIO function block is to calculate the Control setpoint corresponding to the control variable.

EN and ENO can be configured as additional parameters.

Properties

The function block contains the following properties:

- The ratio can be controlled remotely (K) or (RK)
- Upper and lower threshold for K or RK
- Upper and lower threshold for the calculated Setpoint SP
- Calculation of the real ratio: KACT = (PV - bias) / PV_TRACK

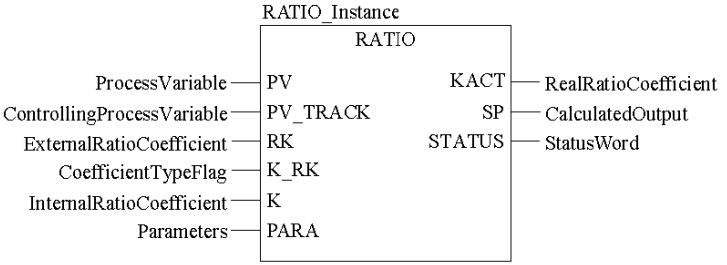
Formula

Calculation of the control setpoint

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

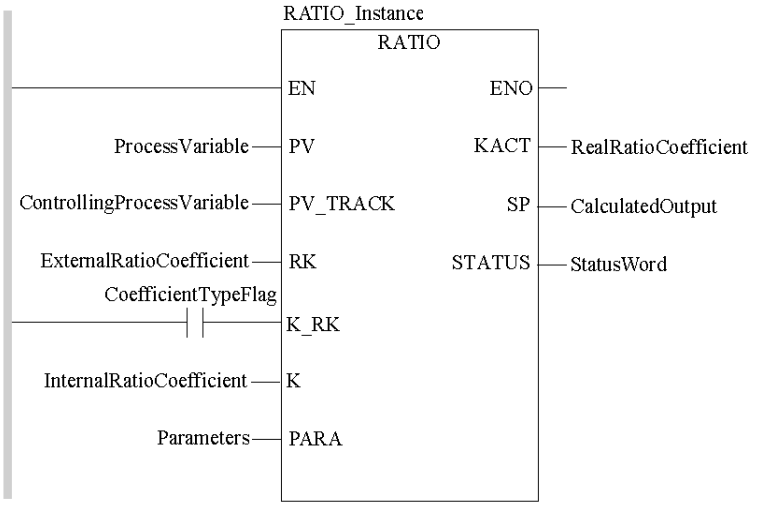
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL RATIO_Instance (PV:=ProcessVariable,
    PV_TRACK:=ControllingProcessVariable,
    RK:=ExternalRatioCoefficient,
    K_RK:=CoefficientTypeFlag, K:=InternalRatioCoefficient,
    PARA:=Parameters, KACT=>RealRatioCoefficient,
    SP=>CalculatedOutput, STATUS=>StatusWord)
```

Representation in ST

Representation:

```
RATIO_Instance (PV:=ProcessVariable,
    PV_TRACK:=ControllingProcessVariable,
    RK:=ExternalRatioCoefficient,
    K_RK:=CoefficientTypeFlag, K:=InternalRatioCoefficient,
    PARA:=Parameters, KACT=>RealRatioCoefficient,
    SP=>CalculatedOutput, STATUS=>StatusWord) ;
```

Parameter description RATIO

Input parameter description:

Parameter	Data type	Meaning
PV	REAL	Process value regulated by the control loop (only used to calculate KACT)
PV_TRACK	REAL	Reference variable of the control loop
RK	REAL	Remote relationship coefficient
K_RK	BOOL	Coefficient type for ratio used "1": remote ratio RK "0": local ratio K
K	REAL	Coefficient for local ratio
PARA	Para_RATIO	Parameter

Output parameter description:

Parameter	Data type	Meaning
KACT	REAL	Coefficient for real ratio
SP	REAL	Calculated output
STATUS	WORD	Status word

Parameter description Para_RATIO

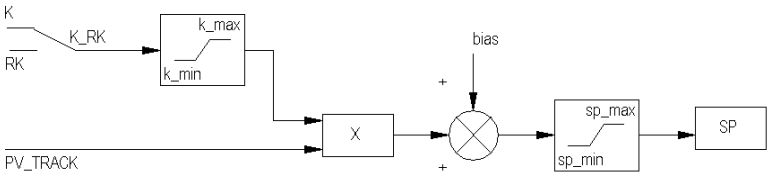
Data structure description

Element	Data type	Meaning
k_min	REAL	Lower threshold with K or RK
k_max	REAL	Upper threshold with K or RK
sp_min	REAL	Lower threshold of the calculated output SP
sp_max	REAL	Upper threshold of the calculated output SP
bias	REAL	Offset coefficient

Detailed description

Structure diagram

Structure diagram of the RATIO

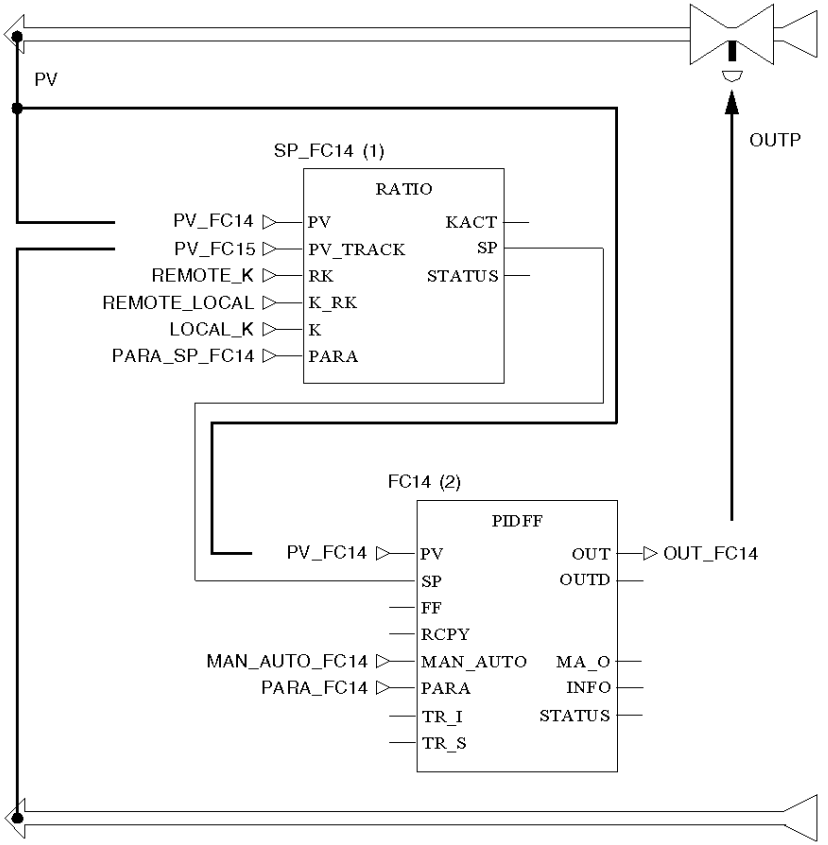


Application

The RATIO function block is upstream of a ratio controller. Its function is to calculate the remote setpoint SP of one of the controllers upgraded subsequently. The ratio controller must consist of the function blocks RATIO, SP_SEL and a controller.

Generally, this type of controller is used to regulate a flow in relation to another measured flow; it observes a specific ratio K between the two flow amounts.

Representation of the ratio controller



Runtime error

Status word

The following messages are displayed in the Status word:

Bit	Value in Dec.	Value in Hex.	ENO Status	Description
Bit 0 = 1	1	0x0001	False	Error in a floating point value calculation
Bit 1 = 1	2	0x0002	False	Recording of an invalid value on one of the floating point value inputs
Bit 2 = 1	4	0x0004	False	Division by zero during a floating point value calculation
Bit 3 = 1	8	0x0008	False	Capacity overflow for a calculation with floating point values
Bit 4 = 1	16	0x0010	True	The input K (or RK) is outside the range [k_min, k_max]: For calculation the function block uses the value k_min or k_max.
Bit 5 = 1	32	0x0020	True	The output SP has reached the lower threshold sp_min: SP is limited to sp_min
Bit 6 = 1	64	0x0040	True	The output SP has reached the upper threshold sp_max: SP is limited to sp_max

For the list of other possible floating point error codes, see *Common Floating Point Errors, page 365*.

Error message

The error appears if a non floating value is recorded or if there is a problem with a floating point calculation. The outputs KACT and SP remain unmodified.

Chapter 39
SP_SEL: Setpoint switch

Introduction

This chapter describes the SP_SEL block.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	348
Detailed description	351
Runtime error	354

Description

Function description

This function block allows the selection of setpoint value types used in the servoloop:

Setpoint value type	Explanation
Remote setpoint (SP_RSP = 1)	The Setpoint comes from a block external calculation using the input RSP (remote setpoint). The input value RSP leads to the SP output.
Local setpoint (SP_RSP = 0)	The setpoint must be modified directly by the user (Local setpoint). In this operating mode the output SP is not entered using the function block, the variable attached to the SP is modified by the user.

EN and ENO can be configured as additional parameters.

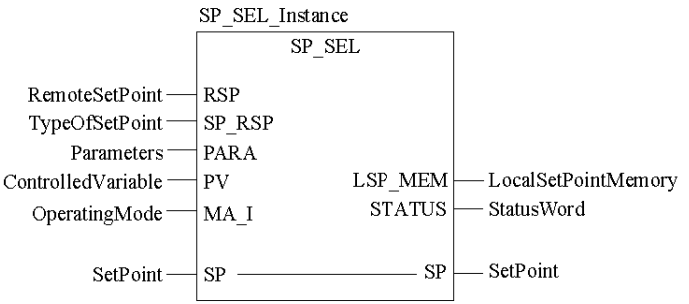
Properties

The function block SP_SEL has the following properties:

- The switchover between the setpoint values can be bumpless
- Operation with adjusting setpoint values if the controller is in manual mode
- Upper and lower limit of the setpoint value used

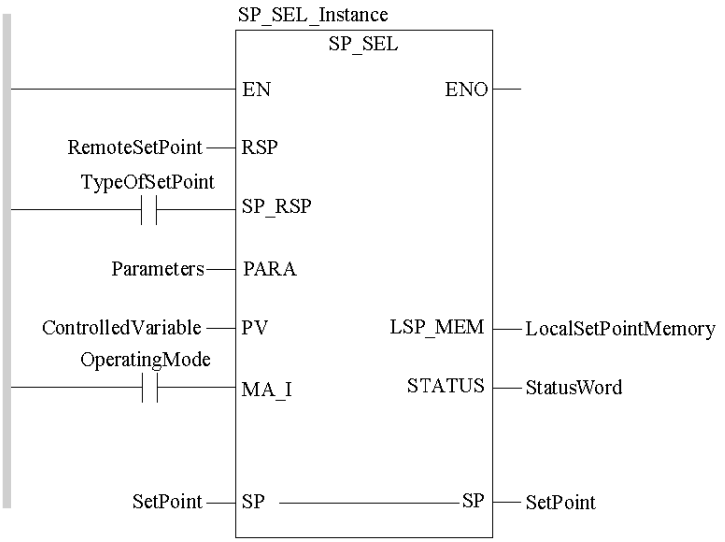
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL SP_SEL_Instance (RSP:=RemoteSetPoint,
    SP_RSP:=TypeOfSetPoint, PARA:=Parameters,
    PV:=ControlledVariable, MA_I:=OperatingMode,
    SP:=SetPoint, LSP_MEM=>LocalSetPointMemory,
    STATUS=>StatusWord)
```

Representation in ST

Representation:

```
SP_SEL_Instance (RSP:=RemoteSetPoint,
    SP_RSP:=TypeOfSetPoint, PARA:=Parameters,
    PV:=ControlledVariable, MA_I:=OperatingMode,
    SP:=SetPoint, LSP_MEM=>LocalSetPointMemory,
    STATUS=>StatusWord) ;
```

Parameter description SP_SEL

Input parameter description:

Parameter	Data type	Meaning
RSP	REAL	Remote setpoint
SP_RSP	BOOL	Setpoint type used by the controller "1" : Remote setpoint "0" : Local setpoint
PARA	Para_SP_SEL	Parameter
PV	REAL	Variables to be controlled
MA_I	BOOL	Operating mode of the linked controller "1" : Automatic operating mode "0" : Manual mode

Input / output parameter description:

Parameter	Data type	Meaning
SP	REAL	Setpoint used by the controller

Output parameter description:

Parameter	Data type	Meaning
LSP_MEM	REAL	Local setpoint MEMory
STATUS	WORD	Status word

Parameter description Para_SP_SEL

Data structure description

Element	Data type	Meaning
sp_min	REAL	Lower threshold for setpoint used
sp_max	REAL	Upper threshold for setpoint used
bump	BOOL	During remote/local changeover: "1" : the SP output is forced with the value of LSP_MEM "0" : bumpless changeover
track	BOOL	"1" : the values of SP and PV are brought into line in manual mode (local setpoint only)
rate	REAL	SP increase during local/remote changeover in units per second (≥0)

Detailed description

Switching the setpoint

The setpoint can be switched in two directions

If	Then
SP_RSP from 0 →1	the local setpoint is switched to a remote setpoint
SP_RSP from 1 →0	the remote setpoint is switched to a local setpoint

SP_RSP from 0 →1

The changeover from local setpoint to remote setpoint is bumpless: the value of the SP output is increasingly adjusted to correspond to the remote setpoint RSP, and it describes the ramp rate.

If rate rate = 0, there is no ramp and the SP is identical to the RSP.

SP_RSP from 0 →0

The changeover from remote setpoint to local setpoint depends on the bump element in two ways:

If	Then
bump = 0	the changeover is bumpless: The function block stops copying the RSP input to the SP output. The local setpoint value SP then corresponds to the last remote setpoint value RSP that was present before the changeover. The user can then modify this. In this case it is not necessary to attach the LSP_MEM output.
bump = 1	the value of the LSP_MEM output is moved to the SP output during changeover (bumps can occur here). The value given for LSP_MEM corresponds to the last setpoint value SP before the function block transfers to remote mode. To restart the local mode with a different setpoint, it is sufficient to modify LSP_MEM as long as the block remains in remote mode (for further details see "*Function of the output LSP_MEM, page 352*").

Tracked setpoint (track = 1)

At local setpoint value (SP_RSP=0), and with the linked controller in manual mode, the PV input can be continuously copied to the setpoint SP value being used. This enables a bumpless changeover from manual to automatic mode (it is also possible for the controller to control the bumpless behavior itself).

In this operating mode, the inputs PV and MA_I of the function block SP_SEL must be attached. You must adopt the same value as the controllers PV input and its outputMA_O. If track = 0, these inputs do not need to be attached.

Limits

In each operating mode (remote or local) the setpoint value SP used is limited to the range between sp_min and sp_max.

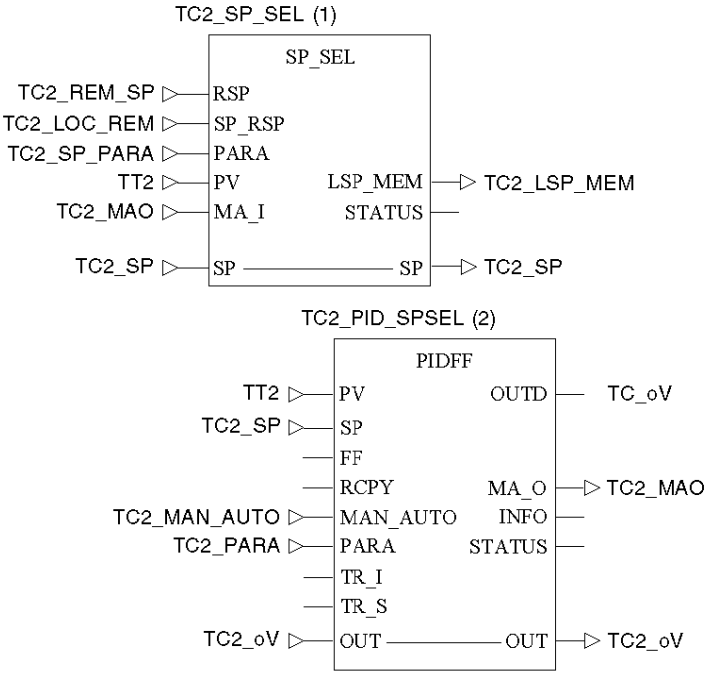
Function of the output LSP_MEM

This output enables the user to control the setpoint value SP during a remote – local changeover:

Type of setpoint	Behavior of the output
Local setpoint	The value of SP is continuously moved to LSP_MEM.
Changeover to remote setpoints	The value of LSP_MEM is no longer modified by the function block and therefore retains the value of the last local setpoint used.
Reverting to the local setpoint	There are three possibilities for this: **1.** bump = 0: The last remote setpoint value is used as a basis; in this case LSP_MEM does not need to be attached). **2.** bump = 1: The last local value saved is used as a basis; during changeover the block copies the value of LSP_MEM onto SP. **3.** The function block can start local mode using any value selected by the user. If the value of the variable attached to LSP_MEM before transfer to the local setpoint (with bump = 1) is modified, it is moved to SP during the changeover.

Example of programming

An example of how to program the `SP_SEL` function block follows.



TC2_SP is entered by the operator in "local setpoint" operating mode.

Runtime error

Status word

The following messages are displayed in the Status word:

Bit	Value in Dec.	Value in Hex.	ENO Status	Description
Bit 0 = 1	1	0x0001	False	Error in a floating point value calculation
Bit 1 = 1	2	0x0002	False	Invalid value recorded at one of the floating point inputs
Bit 2 = 1	4	0x0004	False	Division by zero during a floating point value calculation
Bit 3 = 1	8	0x0008	False	Capacity overflow during a floating point value calculation
Bit 4 = 1	16	0x0010	True	`rate` is negative : For calculation, the function block uses the value 0
Bit 5 = 1	32	0x0020	True	The output `SP` has reached the lower threshold `sp_min`: `SP` is forced to `sp_min`
Bit 6 = 1	64	0x0040	True	The output `SP` has reached the upper threshold `sp_max`: `SP` is forced to `sp_max`

For the list of other possible floating point error codes, see *Common Floating Point Errors, page 365*.

Error message

An runtime error appears if a non floating point value is recorded or if there is a problem with a floating point calculation. The outputs `SP` and `LSP_MEM` remain unmodified.

NOTE: For a list of all block error codes and values, see *Setpoint Management, page 363*.

Warning

A warning is giving if `rate` is negative; the block then uses the value 0 for calculation.

Appendices

Appendix A
EFB Error Codes and Values

Introduction

The following tables show the error codes and error values created for the EFBs of the CONT_CTL Library.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Tables of Error Codes for the CONT_CTL Library	358
Common Floating Point Errors	365

Tables of Error Codes for the CONT_CTL Library

Introduction

The following tables show the error codes and error values created for the EFBs of the CONT_CTL Library.

Conditioning

Table of error codes and errors values created for EFBs of the `Conditioning` family.

EFB name	Error code	ENO state in case of error	Error value in Dec	Error value in Hex	Error description
DTIME	W_WARN_OUT_OF_RANGE	T	30110	16#759E	Parameter out of range
DTIME	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*
DTIME	Status word values	T/F	-	-	For details about the `DTIME` status word refer to the DTIME desription *(see page 47)*
INTEGRATOR	E_ERR_DEN	F	-30152	16#8A38	Not a valid floating point number
INTEGRATOR	E_ERR_IB_MAX_MIN	F	-30102	16#8A6A	YMAX < YMIN
INTEGRATOR	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*
LAG_FILTER	E_ERR_DEN	F	-30152	16#8A38	Not a valid floating point number
LAG_FILTER	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*
LDLG	E_ERR_DEN	F	-30152	16#8A38	Not a valid floating point number
LDLG	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*
LEAD	E_ERR_DEN	F	-30152	16#8A38	Not a valid floating point number
LEAD	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*
MFLOW	W_WARN_OUT_OF_RANGE	T	30110	16#759E	Parameter out of range
MFLOW	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*
MFLOW	Status word values	T/F	-	-	For details about the `MFLOW` status word refer to the MFLOW desription *(see page 83)*

EFB name	Error code	ENO state in case of error	Error value in Dec	Error value in Hex	Error description
QDTIME	E_ERR_DEN	F	-30152	16#8A38	Not a valid floating point number
SCALING	E_ERR_NULL_INPUT_SCALE	F	-30121	16#8A57	Null input scale: max and min limit must be different
SCALING	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*
SCALING	Status word values	T/F	-	-	For details about the `SCALING` status word refer to the SCALING description *(see page 95)*
TOTALIZER	W_WARN_OUT_OF_RANGE	T	30110	16#759E	Parameter out of range
TOTALIZER	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*
TOTALIZER	W_WARN_TOTALIZER_CTER_MAX	T	30113	16#75A1	Maximum value of cter has been reached
TOTALIZER	Status word values	T/F	-	-	For details about the `TOTALIZER` status word refer to the TOTALIZER description *(see page 107)*
VEL_LIM	E_ERR_DEN	F	-30152	16#8A38	Not a valid floating point number
VEL_LIM	E_ERR_AB1_MAX_MIN	F	-30101	16#8A6B	YMAX < YMIN
VEL_LIM	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*

Controller

Table of error codes and errors values created for EFBs of the `Controller` family.

EFB name	Error code	ENO state in case of error	Error value in Dec	Error value in Hex	Error description
AUTOTUNE	W_WARN_OUT_OF_RANGE	T	30110	16#759E	Parameter out of range
AUTOTUNE	E_ERR_NULL_INPUT_SCALE	F	-30121	16#8A57	Null input scale: max and min limit must be different
AUTOTUNE	W_WARN_AUTOTUNE_FAILED	T	30111	16#759F	AUTOTUNE has failed

EFB name	Error code	ENO state in case of error	Error value in Dec	Error value in Hex	Error description
AUTOTUNE	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*
AUTOTUNE	E_ERR_AUTOTUNE_ID_ UNKNOWN	F	-30120	16#8A58	The tuned EFB is not allowed or has not yet been called
AUTOTUNE	Status word values	T/F	-	-	For details about the AUTOTUNE status word refer to the AUTOTUNE description *(see page 139)*
PI_B	W_WARN_OUT_OF_RANGE	T	30110	16#759E	Parameter out of range
PI_B	E_ERR_NULL_INPUT_SCALE	F	-30121	16#8A57	Null input scale: max and min limit must be different
PI_B	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*
PI_B	Status word values	T/F	-	-	For details about the PI_B status word refer to the PI_B description *(see page 165)*
PIDFF	W_WARN_OUT_OF_RANGE	T	30110	16#759E	Parameter out of range
PIDFF	E_ERR_NULL_INPUT_SCALE	F	-30121	16#8A57	Null input scale: max and min limit must be different
PIDFF	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*
PIDFF	Status word values	T/F	-	-	For details about the PIDFF status word refer to the PIDFF description *(see page 193)*
SAMPLETM	E_EFB_SAMPLE_TIME_ OVERFLOW	F	-30184	16#8A18	Internal error
STEP2	W_WARN_OUT_OF_RANGE	T	30110	16#759E	Parameter out of range
STEP2	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*
STEP2	Status word values	T/F	-	-	For details about the STEP2 status word refer to the STEP2 description *(see page 203)*
STEP3	W_WARN_OUT_OF_RANGE	T	30110	16#759E	Parameter out of range
STEP3	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*

EFB name	Error code	ENO state in case of error	Error value in Dec	Error value in Hex	Error description
STEP3	Status word values	T/F	-	-	For details about the STEP3 status word refer to the STEP3 description *(see page 211)*

Mathematics

Table of error codes and errors values created for EFBs of the Mathematics family.

EFB name	Error code	ENO state in case of error	Error value in Dec	Error value in Hex	Error description
COMP_DB	W_WARN_OUT_OF_RANGE	T	30110	16#759E	Parameter out of range
COMP_DB	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*
K_SQRT	W_WARN_OUT_OF_RANGE	T	30110	16#759E	Parameter out of range
K_SQRT	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*
MULDIV_W	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*
SUM_W	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*

Measurement

Table of error codes and errors values created for EFBs of the Measurement family.

EFB name	Error code	ENO state in case of error	Error value in Dec	Error value in Hex	Error description
AVGMV	E_ERR_DEN	F	-30152	16#8A38	Not a valid floating point number
AVGMV	W_WARN_AVGMV	T	30108	16#759C	AVGMV: N must be <= 50

EFB name	Error code	ENO state in case of error	Error value in Dec	Error value in Hex	Error description
AVGMV	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*
AVGMV_K	E_ERR_DEN	F	-30152	16#8A38	Not a valid floating point number
AVGMV_K	W_WARN_AVGMV_K	T	30109	16#759D	AVGMV_K: N must be <= 10000
AVGMV_K	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*
DEAD_ZONE	E_ERR_DEN	F	-30152	16#8A38	Not a valid floating point number
DEAD_ZONE	E_ERR_DZONE	F	-30119	16#8A59	DZONE: DZ must be >= 0
DEAD_ZONE	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*
LOOKUP_ TABLE1	E_ERR_DEN	F	-30152	16#8A38	Not a valid floating point number
LOOKUP_ TABLE1	E_ERR_POLY_ ANZAHL	F	-30107	16#8A65	number of inputs not even
LOOKUP_ TABLE1	E_ERR_POLY_ FOLGE	F	-30108	16#8A64	base point x(i) <= x(i-1)
LOOKUP_ TABLE1	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*

Output Processing

Table of error codes and errors values created for EFBs of the `Output Processing` family.

EFB name	Error code	ENO state in case of error	Error value in Dec	Error value in Hex	Error description
MS	W_WARN_OUT_OF_RANGE	T	30110	16#759E	Parameter out of range
MS	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*
MS	Status word values	T/F	-	-	For details about the `MS` status word refer to the MS description *(see page 282)*
PWM1	WAF_PBM_TMINMAX	F	-30113	16#8A5F	t_min < t_max

EFB name	Error code	ENO state in case of error	Error value in Dec	Error value in Hex	Error description
PWM1	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*
SERVO	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*
SERVO	Status word values	T/F	-	-	For details about the SERVO status word refer to the SERVO description *(see page 321)*
SPLRG	W_WARN_OUT_OF_RANGE	T	30110	16#759E	Parameter out of range
SPLRG	E_ERR_NULL_INPUT_SCALE	F	-30121	16#8A57	Null input scale: max and min limit must be different
SPLRG	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*
SPLRG	Status word values	T/F	-	-	For details about the SPLRG status word refer to the SPLRG description *(see page 328)*

Setpoint Management

Table of error codes and errors values created for EFBs of the Setpoint Management family.

EFB name	Error code	ENO state in case of error	Error value in Dec	Error value in Hex	Error description
RAMP	W_WARN_OUT_OF_RANGE	T	30110	16#759E	Parameter out of range
RAMP	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*
RAMP	Status word values	T/F	-	-	For details about the RAMP status word refer to the RAMP description *(see page 338)*
RATIO	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*
RATIO	Status word values	T/F	-	-	For details about the RATIO status word refer to the RATIO description *(see page 345)*

EFB name	Error code	ENO state in case of error	Error value in Dec	Error value in Hex	Error description
SP_SEL	W_WARN_OUT_OF_RANGE	T	30110	16#759E	Parameter out of range
SP_SEL	FP_ERROR	F	-	-	See table *Common Floating Point Errors, page 365*
SP_SEL	Status word values	T/F	-	-	For details about the `SP_SEL` status word refer to the SP_SEL description *(see page 354)*

Common Floating Point Errors

Introduction

The following table shows the common error codes and error values created for floating point errors. These error information are displayed in the Diagnostic Viewer and the error code values are written in %SW125 *(see Unity Pro, System Bits and Words, Reference Manual)*.

Common Floating Point Errors

Table of common floating point errors

Error codes	Error value in Dec	Error value in Hex	Error description
FP_ERROR	-30150	16#8A3A	Base value (not apearing as an error value)
E_FP_STATUS_FAILED_IE	-30151	16#8A39	Illegal floating point operation
E_FP_STATUS_FAILED_DE	-30152	16#8A38	Operand is denormalized - not a valid REAL number
E_FP_STATUS_FAILED_ZE	-30154	16#8A36	Illegal divide by zero
E_FP_STATUS_FAILED_ZE_IE	-30155	16#8A35	Illegal floating point operation / Divide by zero
E_FP_STATUS_FAILED_OE	-30158	16#8A32	Floating point overflow
E_FP_STATUS_FAILED_OE_IE	-30159	16#8A31	Illegal floating point operation / Overflow
E_FP_STATUS_FAILED_OE_ZE	-30162	16#8A2E	Floating point overflow / Divide by zero
E_FP_STATUS_FAILED_OE_ZE_IE	-30163	16#8A2D	Illegal floating point operation / Overflow / Divide by zero
E_FP_NOT_COMPARABLE	-30166	16#8A2A	Internal error

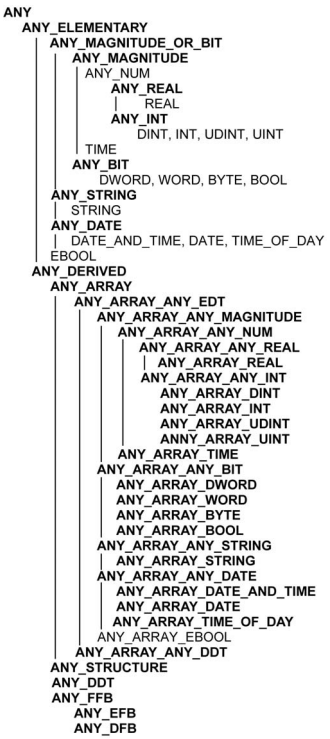
Glossary

A

ANY

There is a hierarchy among the various data types. In the DFBs, it is sometimes possible to declare variables that can contain several types of values. In that case we use `ANY_xxx` types.

The figure below describes this hierarchical structure:



B

BOOL

`BOOL` is the abbreviation for the Boolean type. This is the basic data type in computing. A `BOOL` variable can have either of the following two values: 0 (`FALSE`) or 1 (`TRUE`).

A bit extracted from a word is of type `BOOL`, for example: `%MW10.4`.

D

DINT

DINT is the abbreviation of Double INTeger (encoded in 32 bits).

The upper/lower limits are as follows: -(2 to the power of 31) to (2 to the power of 31) - 1.

Example:

-2147483648, 2147483647, 16#FFFFFFFF.

E

EN

EN stands for **EN**able; it is an optional block input. When the EN input is enabled, an ENO output is set automatically.

If EN = 0, the block is not enabled; its internal program is not executed, and ENO is set to 0.

If EN = 1, the block's internal program is run and ENO is set to 1. If an error occurs, ENO is set to 0.

If the EN input is not connected, it is set automatically to 1.

ENO

ENO stands for **E**rror **NO**tification; this is the output associated with the optional input EN.

If ENO is set to 0 (because EN = 0 or in case of an execution error):

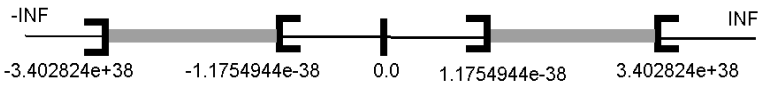
- the status of the function block outputs remains the same as it was during the previous scanning cycle that executed correctly;
- the output(s) of the function, as well as the procedures, are set to "0".

I

INF

Used to indicate that a number exceeds the authorized limits.

For an integer, the value ranges (shown in gray) are as follows:



When a result is:

- less than -3.402824e+38, the symbol -INF (for -infinity) is displayed;
- greater than +3.402824e+38, the symbol INF (for +infinity) is displayed;

INT

INT is the abbreviation of single INTeger (encoded in 16 bits).

The upper/lower limits are as follows: -(2 to the power of 15) to (2 to the power of 15) - 1.

Example:

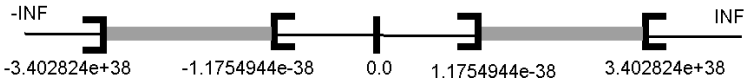
-32768, 32767, 2#1111110001001001, 16#9FA4.

R

REAL

The REAL type is encoded in a 32 bit format.

The possible value ranges are shown in the figure below:



When a result is:

- between -1,175494e-38 and 1,175494e-38, it is considered to be a DEN;
- less than -3.402824e+38, the symbol -INF (for - infinity) is displayed;
- greater than +3.402824e+38, the symbol INF (for + infinity) is displayed;
- undefined (square root of a negative number), the symbolNAN is displayed.

NOTE: The IEC 559 standard defines two classes of NAN: the silent NAN (QNAN) and the signaling NAN (SNAN). A QNAN is a NAN with a most significant fraction bit while an SNAN is a NAN without a most significant fraction bit (bit number 22). QNANs can be propagated via most arithmetic operations without throwing an exception. As for SNANs, they generally indicate n invalid operation when they are used as operands in arithmetic operations (see %SW17 and %S18).

NOTE: When a DEN (non-standardized number) is used as an operand, the result is not significant.

T

TIME

The TIME type expresses a time in milliseconds. Encoded in 32 bits, this type can be used to obtain times from 0 to 2^{32}-1 milliseconds.

The TIME type has the following units: days (d), hours (h), minutes (m), seconds (s) and milliseconds (ms). A literal value of type TIME is represented by a combination of the preceding types prefixed with T#, t#, TIME# or time#.

Examples: T#25h15m, t#14,7S, TIME#5d10h23m45s3ms

U

UDINT

UDINT is the abbreviation of Unsigned Double INTeger (encoded in 32 bits). The upper/lower limits are as follows: 0 to (2 to the power of 32) - 1.

Example:

0, 4294967295, 2#11111111111111111111111111111111, 8#37777777777, 16#FFFFFFFF.

UINT

UINT is the abbreviation of the Unsigned INTeger format (encoded in 16 bits). The upper/lower limits are as follows: 0 to (2 to the power of 16) - 1.

Example:

0, 65535, 2#1111111111111111, 8#177777, 16#FFFF.

W

WORD

The type WORD is encoded in a 16 bit format and is used to perform processing on series of bits.

This table shows the upper/lower limits of each of the bases that can be used:

Base	Lower limit	Upper limit
Hexadecimal	16#0	16#FFFF
Octal	8#0	8#177777
Binary	2#0	2#1111111111111111

Examples of representation

Data	Representation in one of the bases
0000000011010011	16#D3
1010101010101010	8#125252
0000000011010011	2#11010011

Index

A

AUTOTUNE, *117*
availability of the instructions, *25*
AVGMV, *235*
AVGMV_K, *241*

C

COMP_DB, *215*
COMP_DB_DFB, *215*
COMP_DB/COMP_DB_DFB, *216*
conditioning - instructions
  DTIME, *39*
  INTEGRATOR, *49*
  LAG_FILTER, *57*
  LDLG, *63*
  LEAD, *71*
  MFLOW, *77*
  QDTIME, *85*
  SCALING, *91*
  TOTALIZER, *97*
  VEL_LIM, *109*
controller - instructions
  AUTOTUNE, *117*
  general information, *29*
  IMC, *141*
  MS, *273*
  MS_DB, *285*
  PI_B, *153*
  PIDFF, *167*
  SAMPLETM, *195*
  SERVO, *305*
  SPLRG, *323*
  STEP2, *197*
  STEP3, *205*

D

DEAD_ZONE, *247*
DEAD_ZONE_REAL, *247*
DTIME, *39*

E

error codes, *357*

H

HYST_***, *261*

I

IMC, *141*
INDLIM_***, *265*
instructions
  availability, *25*
INTEGRATOR, *49*

K

K_SQRT, *221*

L

LAG_FILTER, *57*
LDLG, *63*
LEAD, *71*
LOOKUP_TABLE1, *253*

M

mathematical - instructions
  COMP_DB, *215*
  COMP_DB_DFB, *215*
  K_SQRT, *221*
  MULDIV_W, *225*
  SUM_W, *229*

measurement - instructions
AVGMV, *235*
AVGMV_K, *241*
DEAD_ZONE, *247*
DEAD_ZONE_REAL, *247*
HYST_***, *261*
INDLIM_***, *265*
LOOKUP_TABLE1, *253*
SAH, *259*
MFLOW, *77*
MS, *273*
MS_DB, *285*
MULDIV_W, *225*

O

output processing - instructions
MS, *273*
MS_DB, *285*
PWM1, *297*
SERVO, *305*
SPLRG, *323*

P

PI_B, *153*
PIDFF, *167*
PWM1, *297*

Q

QDTIME, *85*

R

RAMP, *333*
RATIO, *339*

S

SAH, *259*
SAMPLETM, *195*
SCALING, *91*
SERVO, *305*
setpoint management - instructions
RAMP, *333*
RATIO, *339*
SP_SEL, *347*
SP_SEL, *347*
SPLRG, *323*
STEP2, *197*
STEP3, *205*
SUM_W, *229*

T

TOTALIZER, *97*

V

VEL_LIM, *109*