SIEMENS

SIMATIC

Process Control System PCS 7 Help on CFC Elementary Blocks

Programming and Operating Manual

10/2017
A5E41367261-AB

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER

indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING

indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION

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

NOTICE

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

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING

Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Division Process Industries and Drives
Postfach 48 48
90026 NÜRNBERG
GERMANY

A5E41367261-AB
℗ 10/2017 Subject to change

Table of contents

Security information

1

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines, and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions only form one element of such a concept.

Customer is responsible to prevent unauthorized access to its plants, systems, machines and networks. Systems, machines and components should only be connected to the enterprise network or the internet if and to the extent necessary and with appropriate security measures (e.g. use of firewalls and network segmentation) in place.

Additionally, Siemens' guidance on appropriate security measures should be taken into account. For more information about industrial security, please visit:

http://www.siemens.com/industrialsecurity.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends to apply product updates as soon as available and to always use the latest product versions. Use of product versions that are no longer supported, and failure to apply latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under

http://www.siemens.com/industrialsecurity.

Block parameters EN, ENO, SAMPLE_T

2

EN

EN (enable): Enable input.

The input exists only in the graphical interface of CFC however it is switched to hidden. You can use the enable input to enable/disable block processing. It ensures that the block is only called in the task code at AS level if it has been enabled via EN = 1.

ENO

ENO: ENO is the same as BR (Binary Result - refer to the STEP 7 documentation).

ENO = 1 shows a valid result as appropriate to the function. When the operating system and/or the error-handling routine detects an error in the block program, ENO will be set to 0 to indicate an invalid result. You can use this information in order to switch to other values (e.g., safety values) and to output messages to the OS as required.

If EN = FALSE, then ENO = FALSE.

SAMPLE_T

All blocks with the input parameter SAMPLE_T must be processed in temporally equidistant tasks, e.g. OB 35: Watchdog interrupt 100 ms. If they are implemented in acyclic tasks, e.g., process alarms, these blocks will return incorrect results

Startup on S7-300 CPUs

3

Startup

As it is not possible for S7-300 CPU restarts to be detected automatically, memory word 0 (MW0) is used as a startup bit memory word in startup blocks (blocks included in ELEM_300). This memory word must not, therefore, be modified in the user program.

In order to ensure a proper restart, you must insert the RESTART (FC 70) function into a CFC chart once for each S7-300 CPU.

Procedure:

1. Open the process editor with the menu command **Edit > Process Sequence** or via the icon in the toolbar.
2. Position the RESTART block at the beginning of OB 100.
3. Delete the RESTART block in the cyclic task (default: OB 35)
   the block will only still be called in OB 100.

CFC blocks

4

The following CFC block families are available:

Family	Purpose
BIT_LGC (Page 17)	Logic blocks of the data type BOOL
WRD_LGC (Page 25)	Logic blocks of the data type WORD and DWORD
COMPARE (Page 39)	Blocks for comparing two input values of the same type
CONVERT (Page 45)	Blocks for converting data types
MATH_FP (Page 71)	Arithmetic blocks of the data type REAL
MATH_INT (Page 97)	Arithmetic blocks of the data type INT and DINT
FLIPFLOP (Page 123)	Flip-flop blocks
SHIFT (Page 127)	Shift blocks
MULTIPLX (Page 137)	Multiplexer blocks
COUNTER (Page 145)	Count blocks
PULSE (Page 151)	Blocks for generating or processing pulses
TIME (Page 159)	Blocks for acquiring or processing time intervals and timebases
CONTROL (Page 163)	Closed-loop-control blocks
EVENT: Start of the priority class (Page 185)	Blocks for system functions
Blocks for AS-wide connections (Page 195)	Blocks for AS-wide connections

You will find information on the ALARM_8P, BSEND and BRCV blocks in the online help for PCS 7 Standard Library.

Note

The initial value at the output of the CFC blocks is "1", regardless of the values at the inputs. This means the following logic is supplied with "1" as long as the block is not processed.

Logic blocks of the data type BOOL

5

5.1 BIT_LGC

CFC blocks of the "BIT_LGC" family

In this family we have implemented the following blocks with which logic combinations can be implemented:

AND: AND operation (Page 18)	AND operation
OR: OR operation (Page 19)	OR operation
XOR: Exclusive-OR operation (Page 20)	Exclusive-OR operation
NAND: NAND operation (Page 21)	NAND operation
NOR: NOR operation (Page 22)	NOR operation
NOT: NOT operation (Page 23)	NOT operation

5.2 AND: AND operation

Function

This block generates logic AND operations at inputs. The output is 1 if all inputs are 1. Otherwise, the output is 0. The number of "IN" inputs at the block can be modified.

Truth table (example of n = 2)

IN1	IN2	OUT
0	0	0
0	1	0
1	0	0
1	1	1

I/Os

	Name	Data type	Default setting
Inputs	IN1	BOOL	1
	IN2	BOOL	1
	...	...	
	INn	BOOL	1
Output	OUT	BOOL	1

5.3 OR: OR operation

Function

This block generates logic OR operations at inputs. The output is 1 if at least one input is 1. If all inputs are 0, the output is 0. The number of "IN" inputs at the block can be modified.

Truth table (example of n = 2)

IN1	IN2	OUT
0	0	0
0	1	1
1	0	1
1	1	1

I/Os

	Name	Data type	Default setting
Inputs	IN1	BOOL	0
	IN2	BOOL	0
	...	...	
	INn	BOOL	0
Output	OUT	BOOL	0

5.4 XOR: Exclusive-OR operation

Function

This block generates logic exclusive-OR operations at inputs. The output is 0 if all inputs have the same value. Otherwise, the output is 1. The number of "IN" inputs at the block can be modified.

Truth table (example of n = 2)

IN1	IN2	OUT
0	0	0
0	1	1
1	0	1
1	1	0

I/Os

	Name	Data type	Default setting
Inputs	IN1	BOOL	0
	IN2	BOOL	0
	...	...	
	INn	BOOL	0
Output	OUT	BOOL	0

5.5 NAND: NAND operation

Function

This block generates and inverts logic AND operations at inputs. The output is 0 if all inputs are 1. The number of "IN" inputs at the block can be modified.

Truth table (example of n = 2)

IN1	IN2	OUT
0	0	1
0	1	1
1	0	1
1	1	0

I/Os

	Name	Data type	Default setting
Inputs	IN1	BOOL	1
	IN2	BOOL	1
	...	...	
	INn	BOOL	1
Output	OUT	BOOL	0

5.6 NOR: NOR operation

Function

This block generates and inverts OR logic operations at inputs. The output is 1 if all inputs are 0. The number of "IN" inputs at the block can be modified.

Truth table (example of n = 2)

IN1	IN2	OUT
0	0	1
0	1	0
1	0	0
1	1	0

I/Os

	Name	Data type	Default setting
Inputs	IN1	BOOL	0
	IN2	BOOL	0
	...	...	
	INn	BOOL	0
Output	OUT	BOOL	1

5.7 NOT: NOT operation

Function

This block inverts the input.

Truth table

IN	OUT
0	1
1	0

I/Os

	Name	Data type	Default setting
Input	IN	BOOL	0
Output	OUT	BOOL	1

Logic blocks of the data type WORD and DWORD 6

6.1 WRD_LGC

CFC blocks of the "WRD_LGC" family

This family contains the following word logic blocks for the data types WORD and DWORD:

WAND_W: Word AND operation (Page 26)	Word AND operation
AUTOHOTSPOT	Double word AND operation
WOR_W: Word OR operation (Page 27)	Word OR operation
AUTOHOTSPOT	Double word OR operation
WXOR_W: Word exclusive-OR operation (Page 28)	Word exclusive-OR operation
AUTOHOTSPOT	Double word exclusive-OR operation
WNAND_W: Word AND operation (Page 29)	Word NAND operation
AUTOHOTSPOT	Double word NAND operation
WNOR_W: Word NOR operation (Page 30)	Word NOR operation
AUTOHOTSPOT	Double word NOR operation
AUTOHOTSPOT	Word NOT operation
WNOT_DW: Double word NOT operation (Page 37)	Double word NOT operation

6.2 WAND_W: Word AND operation

Function

This block generates word logic AND operations at inputs. All input bits of the same significance are logically linked by AND; the result is written to the corresponding output bit. The number of "IN" inputs at the block can be modified.

Example (two inputs)

IN1	2# 1 1 1 1_0 0 0 0_1 1 1 1_0 0 0 0_1 1 0 1
IN2	2# 1 1 1 1_0 0 0 0_0 0 1 1_0 0 0 0_0 0 1 1
OUT	2# 1 1 1 1_0 0 0 0_0 0 1 1_0 0 0 0_0 0 0 1

I/Os

	Name	Data type	Default setting
Inputs	IN1	WORD	0
	IN2	WORD	0
	...		
Output	OUT	WORD	0

6.3 WOR_W: Word OR operation

Function

This block generates word logic OR operations at inputs. All input bits of the same significance are logically linked by OR and the result is written to the corresponding output bit. The number of "IN" inputs at the block can be modified.

Example (two inputs)

IN1	2# 1 1 1 1_0 0 0 0_1 1 1 1_0 0 0 0_1 1 0 1
IN2	2# 1 1 1 1_0 0 0 0_0 0 1 1_0 0 0 0_0 0 1 1
OUT	2# 1 1 1 1_0 0 0 0_1 1 1 1_0 0 0 0_1 1 1 1

I/Os

	Name	Data type	Default setting
Inputs	IN1	WORD	0
	IN2	WORD	0
	...		
Output	OUT	WORD	0

6.4 WXOR_W: Word exclusive-OR operation

Function

This block generates word logic exclusive-OR operations at inputs. All input bits of the same significance are logically linked by exclusive-OR and the result is written to the corresponding output bit. The bit is 0 if all input bits of the same significance have the same value. Otherwise the bit is 1. The number of "IN" inputs at the block can be modified.

Example (two inputs)

IN1	2# 1 1 1 1_0 0 0 0_1 1 1 1_0 0 0 0_1 1 0 1
IN2	2# 1 1 1 1_0 0 0 0_0 0 1 1_0 0 0 0_0 0 1 1
OUT	2# 0 0 0 0_0 0 0 0_1 1 0 0_0 0 0 0_1 1 1 0

I/Os

	Name	Data type	Default setting
Inputs	IN1	WORD	0
	IN2	WORD	0
	...		
Output	OUT	WORD	0

6.5 WNAND_W: Word AND operation

Function

This block generates word logic NOT AND operations at inputs. All input bits of the same significance are logically linked by AND, inverted and the result is written to the corresponding output bit. The number of "IN" inputs at the block can be modified.

Example (two inputs)

IN1	2# 1 1 1 1_0 0 0 0_1 1 1 1_0 0 0 0_1 1 0 1
IN2	2# 1 1 1 1_0 0 0 0_0 0 1 1_0 0 0 0_0 0 1 1
OUT	2# 0 0 0 0_1 1 1 1_1 1 0 0_1 1 1 1_1 1 1 0

I/Os

	Name	Data type	Default setting
Inputs	IN1	WORD	0
	IN2	WORD	0
	...		
Output	OUT	WORD	0

6.6 WNOR_W: Word NOR operation

Function

This block generates word logic NOT OR operations at inputs. All input bits of the same significance are logically linked by OR. The result is then inverted and written to the corresponding output bit. The number of "IN" inputs at the block can be modified.

Example (two inputs)

IN1	2# 1 1 1 1_0 0 0 0_1 1 1 1_0 0 0 0_1 1 0 1
IN2	2# 1 1 1 1_0 0 0 0_0 0 1 1_0 0 0 0_0 0 1 1
OUT	2# 0 0 0 0_1 1 1 1_0 0 0 0_1 1 1 1_0 0 0 0

I/Os

	Name	Data type	Default setting
Inputs	IN1	WORD	0
	IN2	WORD	0
Output	OUT	WORD	0

6.7 WNOT_W: Word NOT operation

Function

This block inverts the input word. Each bit of the input is inverted and written to the output bit of the corresponding significance.

Example

IN	2# 1 1 1 1_0 0 0 0_1 1 1 1_0 0 0 0_1 1 0 1
OUT	2# 0 0 0 0_1 1 1 1_0 0 0 0_1 1 1 1_0 0 1 0

I/Os

	Name	Data type	Default setting
Input	IN	WORD	0
Output	OUT	WORD	1

6.8 WAND_DW: Double word AND operation

Function

This block generates DWORD logic AND operations at inputs. All input bits of the same significance are logically linked by AND; the result is written to the corresponding output bit. The number of "IN" inputs at the block can be modified.

Example (same as WAND_W, extended to 32 bits)

I/Os

	Name	Data type	Default setting
Inputs	IN1	DWORD	0
	IN2	DWORD	0
Output	OUT	DWORD	0

6.9 WOR_DW: Double word OR operation

Function

This block generates DWORD logic OR operations at inputs. All input bits of the same significance are logically linked by OR and the result is written to the corresponding output bit. The number of "IN" inputs at the block can be modified.

Example (same as WOR_W, extended to 32 bits)

I/Os

	Name	Data type	Default setting
Inputs	IN1	DWORD	0
	IN2	DWORD	0
Output	OUT	DWORD	0

6.10 WXOR_DW: Double word exclusive-OR operation

Function

This block generates DWORD logic exclusive-OR operations at inputs. All input bits of the same significance are logically linked by exclusive-OR and the result is written to the corresponding output bit. The bit is 0 if all input bits of the same significance have the same value. Otherwise the bit is 1. The number of "IN" inputs at the block can be modified.

Example (same as WXOR_W, extended to 32 bits)

I/Os

	Name	Data type	Default setting
Inputs	IN1	DWORD	16#0
	IN2	DWORD	16#0
Output	OUT	DWORD	16#00000000

6.11 WNAND_DW: Double word NAND operation

Function

This block generates DWORD logic NOT AND operations at inputs. All input bits of the same significance are logically linked by AND, inverted and the result is written to the corresponding output bit. The number of "IN" inputs at the block can be modified.

Example (same as WNAND_W, extended to 32 bits)

I/Os

	Name	Data type	Default setting
Inputs	IN1	DWORD	0
	IN2	DWORD	0
Output	OUT	DWORD	0

6.12 WNOR_DW: Double word NOR operation

Function

This block generates DWORD logic NOT OR operations at inputs. All input bits of the same significance are logically linked by OR. The result is then inverted and written to the corresponding output bit. The number of "IN" inputs at the block can be modified.

Example (same as WNOR_W, extended to 32 bits)

I/Os

	Name	Data type	Default setting
Inputs	IN1	DWORD	0
	IN2	DWORD	0
Output	OUT	DWORD	0

6.13 WNOT_DW: Double word NOT operation

Function

This block inverts the input word. Each bit of the input is inverted and written to the output bit of the corresponding significance.

Example (same as WNOT_W, extended to 32 bits)

I/Os

	Name	Data type	Default setting
Input	IN	DWORD	0
Output	OUT	DWORD	1

Blocks for comparing two input values of the same type 7

7.1 COMPARE

CFC blocks of the "COMPARE" family

This family contains blocks for comparing two input variables:

CMP_I: Comparator for INT values (Page 40)	Comparator for INT values
CMP_DI: Comparator for DINT values (Page 41)	Comparator for DINT values
CMP_R: Comparator for REAL values (Page 42)	Comparator for REAL values
CMP_T: Comparator for TIME values (Page 43)	Comparator for TIME values

7.2 CMP_I: Comparator for INT values

Function

This block compares two input variables and sets the outputs as follows:

GT = 1 if IN1 > IN2

GE = 1 if IN1 ≥ IN2,

EQ = 1 if IN1 = IN2

LE = 1 if IN1 ≤ IN2

LT = 1 if IN1 < IN2

I/Os

	Name	Data type	Description	Default setting
Inputs	IN1	INT	Input variable 1	0
	IN2	INT	Input variable 2	0
Outputs	GT	BOOL	1, IN1 > IN2	0
	GE	BOOL	1, IN1 ≥ IN2	0
	EQ	BOOL	1, IN1 = IN2	0
	LE	BOOL	1, IN1 ≤ IN2	0
	LT	BOOL	1, IN1 < IN2	0

7.3 CMP_DI: Comparator for DINT values

Function

This block compares two input variables and sets the outputs as follows:

GT = 1 if IN1 > IN2

GE = 1 if IN1 ≥ IN2

EQ = 1 if IN1 = IN2

LE = 1 if IN1 ≤ IN2

LT = 1 if IN1 < IN2

I/Os

	Name	Data type	Description	Default setting
Inputs	IN1	DINT	Input variable 1	0
	IN2	DINT	Input variable 2	0
Outputs	GT	BOOL	1, IN1 > IN2	0
	GE	BOOL	1, IN1 ≥ IN2	0
	EQ	BOOL	1, IN1 = IN2	0
	LE	BOOL	1, IN1 ≤ IN2	0
	LT	BOOL	1, IN1 < IN2	0

7.4 CMP_R: Comparator for REAL values

Function

This block compares two input variables and sets the outputs as follows:

GT = 1 if IN1 > IN2

GE = 1 if IN1 ≥ IN2

EQ = 1 if IN1 = IN2

LT = 1 if IN1 < IN2

LE = 1 if IN1 ≤ IN2

The other four outputs are set to 0 in each case.

I/Os

	Name	Data type	Description	Default setting
Inputs	IN1	REAL	Input variable 1	0
	IN2	REAL	Input variable 2	0
Outputs	GT	BOOL	1, IN1 > IN2	0
	GE	BOOL	1, IN1 ≥ IN2	0
	EQ	BOOL	1, IN1 = IN2	0
	LT	BOOL	1, IN1 < IN2	0
	LE	BOOL	1, IN1 ≤ IN2	0

7.5 CMP_T: Comparator for TIME values

Function

This block compares two input variables and sets the outputs as follows:

GT = 1 if IN1 > IN2

GE = 1 if IN1 ≥ IN2

EQ = 1 if IN1 = IN2

LE = 1 if IN1 ≤ IN2

LT = 1 if IN1 < IN2

I/Os

	Name	Data type	Description	Default setting
Inputs	IN1	TIME	Input variable 1	0
	IN2	TIME	Input variable 2	0
Outputs	GT	BOOL	1, IN1 > IN2	0
	GE	BOOL	1, IN1 ≥ IN2	0
	EQ	BOOL	1, IN1 = IN2	0
	LE	BOOL	1, IN1 ≤ IN2	0
	LT	BOOL	1, IN1 < IN2	0

Blocks for converting data types

8

8.1 CONVERT

Introduction

CFC only allows the interconnection of block outputs (source type) to block inputs (target type) if both data types are identical (e.g., REAL output <-> REAL input). Conversion blocks must be used to allow the interconnection of different data types. The input and output data of the block are of a different type, and it thus converts the input data type according to the data type set at the output.

Conversion rules

The abbreviated name of the source and target data type, connected by means of an underscore "_", form the type name.

The table below shows a brief description of conversion rules for specific blocks. If the IN input value is not within the allowed range, the OUT output value becomes invalid and the test output ENO = 0 is displayed.

You can evaluate ENO, for example, to provide a substitute/safety value for further processing.

The CONVERT library contains blocks you can use to convert n values of a data type to m values of another data type (the values of m and n may be equal).

The following blocks convert **one** value of a data type into **one** value of another data type:

BY_DW (Page 47)	Converts BYTE to DWORD
BY_W (Page 48)	Converts BYTE to WORD
DI_DW (Page 49)	Converts DINT to DWORD
DI_I (Page 50)	Converts DINT to INT
DI_R (Page 51)	Converts DINT to REAL
DW_DI (Page 52)	Converts DWORD to DINT
DW_R (Page 53)	Converts DWORD to REAL
DW_W (Page 54)	Converts DWORD to WORD
I_DI (Page 55)	Converts INT to DINT
I_DW (Page 56)	Converts INT to DWORD
I_R (Page 57)	Converts INT to REAL
I_W (Page 58)	Converts INT to WORD
R_DI (Page 59)	Converts REAL to DINT
R_DW (Page 60)	Converts REAL to DWORD
R_I (Page 61)	Converts REAL to INT
W_BY (Page 62)	Converts WORD to BYTE
W_DW (Page 63)	Converts WORD to DWORD
W_I (Page 64)	Converts WORD to INT

The following blocks convert **several** BOOL type values into **one** BYTE, WORD or DWORD type value:

BO_BY (Page 65)	Converts BOOL to BYTE, 8 inputs
BO_W (Page 66)	Converts BOOL to WORD, 16 inputs
BO_DW (Page 67)	Converts BOOL to DWORD, 32 inputs

The following blocks convert **one** BYTE, WORD or DWORD type values into **several** BOOL type values:

BY_BO (Page 68)	Converts BYTE to BOOL, 8 outputs
W_BO (Page 69)	Converts WORD to BOOL, 16 outputs
DW_BO (Page 70)	Converts DWORD to BOOL, 32 outputs

8.2 BY_DW

Function

Copies the byte of IN to the low-byte of OUT and sets the high-bytes to 0.

Troubleshooting

n.a.

I/Os

	Name	Data type	Default setting
Input	IN	BYTE	0
Output	OUT	DWORD	0

8.3 BY_W

Function

Copies the byte of IN to the low-byte of OUT and sets the high-byte to 0.

Troubleshooting

n.a.

I/Os

	Name	Data type	Default setting
Input	IN	BYTE	0
Output	OUT	WORD	0

8.4 DI_DW

Function

Copies the bit string of IN to OUT.

Troubleshooting

n.a.

I/Os

	Name	Data type	Description	Default setting
Input	IN	DINT	Input value	0
Output	OUT	DWORD	Output value	0

8.5 DI_I

Function

Converts the IN bit string to INT and copies the result to OUT.

Troubleshooting

If the values of IN lie outside the range of -32 768 to 32 767, ENO = 0 and OUT is an invalid value.

I/Os

	Name	Data type	Description	Default setting
Input	IN	DINT	Input value	0
Output	OUT	INT	Output value	0

8.6 DI_R

Function

Converts the value of IN to a REAL number and copies the result to OUT.

Troubleshooting

n.a.

I/Os

	Name	Data type	Default setting
Input	IN	DINT	0
Output	OUT	REAL	0

8.7 DW_DI

Function

Copies the bit string of IN to OUT.

Troubleshooting

n.a.

I/Os

	Name	Data type	Description	Default setting
Input	IN	DWORD	Input value	0
Output	OUT	DINT	Output value	0

8.8 DW_R

Method of operation

The block only forwards the bit string and does not change any values. To cause a value change to REAL, you must use the DW_DI block and then the DI_R block.

Function

Copies the bit string of IN to OUT.

Troubleshooting

Not applicable

I/Os

	Name	Data type	Description	Default setting
Input	IN	DWORD	Input value	0
Output	OUT	REAL	Output value	0

8.9 DW_W

Function

Copies the low-word of IN to the word of OUT.

Troubleshooting

ENO = 0 if the IN high-word > 0

I/Os

	Name	Data type	Description	Default setting
Input	IN	DWORD	Input value	0
Output	OUT	WORD	Output value	0

8.10 I_DI

Function

Copies the value of IN to OUT.

Troubleshooting

n.a.

I/Os

	Name	Data type	Default setting
Input	IN	INT	0
Output	OUT	DINT	0

8.11 I_DW

Function

Copies the bit string of IN to the low-word of OUT and sets the high-word to 0.

Troubleshooting

n.a.

I/Os

	Name	Data type	Description	Default setting
Input	IN	INT	Input value	0
Output	OUT	DWORD	Output value	0

8.12 I_R

Function

Copies the integer of IN to OUT.

Troubleshooting

n.a.

I/Os

	Name	Data type	Default setting
Input	IN	INT	0
Output	OUT	REAL	0

8.13 I_W

Function

Copies the bit string of IN to OUT.

Troubleshooting

n.a.

I/Os

	Name	Data type	Description	Default setting
Input	IN	INT	Input value	0
Output	OUT	WORD	Output value	0

8.14 R_DI

Function

Converts the REAL value of IN to OUT.

Troubleshooting

If the value of IN is outsidce the range -2.147483648e+09 und 2.147483647e+09, then ENO = 0 and OUT is invalid.

I/Os

	Name	Data type	Default setting
Input	IN	REAL	0
Output	OUT	DINT	0

8.15 R_DW

Method of operation

The block only forwards the bit string and does not change any values. In order to convert REAL to DWORD, you must use the R_TO_DW block (PCS 7 Library).

Function

Copies the bit string of IN to OUT.

Troubleshooting

Not applicable

I/Os

	Name	Data type	Default setting
Input	IN	REAL	0
Output	OUT	DWORD	0

8.16 R_I

Function

The REAL number at input IN is converted to an INT number and output at output OUT. The number is rounded as follows: 0.5 → 0, 1.5 → 2, 2.5 → 2, 3.5 → 4, etc.

Troubleshooting

If the value at IN is not in the range –32768 to 32767, then ENO = 0 and OUT is invalid.

I/Os

	Name	Data type	Default setting
Input	IN	REAL	0
Output:	OUT	INT	0

8.17 W_BY

Function

Copies the low-byte of IN to OUT.

Troubleshooting

If the high-byte > 0, then ENO = 0.

I/Os

	Name	Data type	Description	Default setting
Input	IN	WORD	Input value	0
Output	OUT	BYTE	Output value	0

8.18 W_DW

Function

Copies the word of IN to the low-word of OUT.

Troubleshooting

n.a.

I/Os

	Name	Data type	Default setting
Input	IN	WORD	0
Output	OUT	DWORD	0

8.19 W_I

Function

Copies the bit string of IN to OUT.

Troubleshooting

n.a.

I/Os

	Name	Data type	Description	Default setting
Input	IN	WORD	Input value	0
Output	OUT	INT	Output value	0

8.20 BO_BY

Function

This block converts the eight Boolean type input values to a BYTE type value and outputs the result. 8 BOOL -> 1 BYTE conversions are performed as follows:
The i-th bit of the BYTE value is set to 0 (or 1) when the i-th input value is 0 (or 1). (i = 0 to 7).

Troubleshooting

Not applicable

I/Os

	Name	Data type	Default setting
Inputs	IN0	BOOL	0
	...	...	
	IN7	BOOL	0
Output	OUT	BYTE	0

8.21 BO_W

Function

This block converts the 16 Boolean type input values to a WORD type value and outputs the result. 16 BOOL -> 1 WORD conversions are performed as follows:
The i-th bit of the WORD value is set to 0 (or 1) when the i-th input value is 0 (or 1) (i = 0 to 15).

Troubleshooting

Not applicable

I/Os

	Name	Data type	Default setting
Inputs	IN0	BOOL	0
	...	...	
	IN15	BOOL	0
Output	OUT	WORD	0

8.22 BO_DW

Function

This block converts the 32 Boolean type input values to a DWORD type value and outputs the result. 32 BOOL -> 1 DWORD conversions are performed as follows:
The i-th bit of the DWORD value is set to 0 (or 1) when the i-th input value is 0 (or 1) (i = 0 to 31).

Troubleshooting

Not applicable

I/Os

	Name	Data type	Default setting
Inputs	IN0	BOOL	0
	...	...	
	IN31	BOOL	0
Output	OUT	DWORD	0

8.23 BY_BO

Function

This block converts the input value of data type BYTE to 8 values of data type BOOL and outputs the result in 8 variables. IN bit0 is mapped to OUT0, IN bit1 to OUT1, etc.

Troubleshooting

n.a.

I/Os

	Name	Data type	Default setting
Input	IN	BYTE	0
Outputs	OUT0	BOOL	0
	...	...	
	OUT7	BOOL	0

8.24 W_BO

Function

This block converts the input value of the WORD data type in 16 values of the BOOL data type, which are created at the 16 outputs. IN-Bit0 is then converted to OUT0, IN-Bit1 to OUT1 etc.

Troubleshooting

Not applicable

I/Os

	Name	Data type	Default setting
Input	IN	WORD	0
Outputs	OUT0	BOOL	0
	...	...	
	OUT15	BOOL	0

8.25 DW_BO

Function

This block converts the input value of the DWORD data type in 32 values of the BOOL data type, which are created at the 32 outputs. IN-Bit0 is then converted to OUT0, IN-Bit1 to OUT1 etc.

Troubleshooting

Not applicable

I/Os

	Name	Data type	Default setting
Input	IN	DWORD	0
Outputs	OUT0	BOOL	0
	...	...	
	OUT31	BOOL	0

Arithmetic blocks of the data type REAL

9

9.1 MATH_FP

CFC blocks of the "MATH_FP" family

This family contains the following floating-point arithmetic blocks for data of the type REAL.:

ADD_R: Addition of REAL values (Page 72)	Addition of REAL values
SUB_R: Subtraction of REAL values (Page 73)	Subtraction of REAL values
MUL_R: Multiplication of REAL values (Page 74)	Multiplication of REAL values
DIV_R: Division of REAL values (Page 75)	Division of REAL values
ABS_R: Absolute value of REAL values (Page 78)	Absolute value of REAL values
EPS_R: Accuracy approximation (Page 92)	Precision; approximation
NEG_R: Inverter for REAL values (Page 90)	Negation of REAL values
MAXn_R: Maximum of REAL values (Page 76)	Maximum of REAL values
MINn_R: Minimum of REAL values (Page 77)	Minimum of REAL values
LIM_R: Limiter of REAL values (Page 91)	Limiter of REAL values
CADD_R: Controllable adder of REAL values (Page 93)	Controllable adder of REAL values
SQRT: Square root (Page 79)	Square root
EXP: Exponential function (Page 80)	Exponential function
POW10: Power-of-10 function (Page 81)	Power of ten
LN: Natural logarithm (Page 82)	Natural logarithm
LOG10: Base-10 logarithm (Page 83)	Logarithm to base 10
SIN: Sin function (Page 84)	Sine function
COS: Cos function (Page 85)	Cosine function
TAN: Tan function (Page 86)	Tangent function
ASIN: Arc sin function (Page 87)	Arc sine function
ACOS: Arc cos function (Page 88)	Arc cosine function
ATAN: Arc tan function (Page 89)	Arc tangent function
POWXY: General power function (Page 94)	General power function
SAMP_AVE: Floating average value (Page 95)	Variable average

Note

The value range of real numbers is:
-3.40282e^+38 ... -1.755e^-38 ... 0 ... 1.755e^-38 ... 3.40282e^+38

9.2 ADD_R: Addition of REAL values

Function

This block adds the values of inputs and outputs the sum.

OUT = IN1 + IN2

Troubleshooting

ENO = 0 on overflow or underflow.

I/Os

	Name	Data type	Description	Default setting
Inputs	IN1	REAL	Addend 1	0.0
	IN2	REAL	Addend 2	0.0
Output	OUT	REAL	Sum	0.0

9.3 SUB_R: Subtraction of REAL values

Function

This block subtracts the value at input IN2 from input IN1 and outputs the difference.

OUT = IN1 - IN2

Troubleshooting

ENO = 0 on overflow and underflow.

I/Os

	Name	Data type	Description	Default setting
Inputs	IN1	REAL	Minuend	0.0
	IN2	REAL	Subtrahend	0.0
Output	OUT	REAL	Difference	0.0

9.4 MUL_R: Multiplication of REAL values

Function

This block multiplies the inputs and outputs the product.

OUT = IN1 * IN2

Troubleshooting

ENO = 0 on overflow and underflow.

I/Os

	Name	Data type	Description	Default setting
Inputs	IN1	REAL	Multiplicand	0.0
	IN2	REAL	Multiplier	0.0
Output	OUT	REAL	Product	0.0

9.5 DIV_R: Division of REAL values

Function

This block divides the value at input IN1 by the value at input IN2 and outputs the quotient.

OUT = IN1 / IN2

Troubleshooting

ENO is set to 0 on divisions by 0, overflow and underflow.

I/Os

	Name	Data type	Description	Default setting
Inputs	IN1	REAL	Dividend	0.0
	IN2	REAL	Divisor	0.0
Output	OUT	REAL	Quotient	0.0

9.6 MAXn_R: Maximum of REAL values

Function

This block compares the inputs and outputs their maximum value.

OUT = MAX {IN1, ... , INn}

Blocks

Name	Description
MAX2_R	2 inputs of type REAL
MAX4_R	4 inputs of type REAL
MAX8_R	8 inputs of type REAL

I/Os

	Name	Data type	Description	Default setting
Inputs	IN1	REAL	Input variable 1	0.0
	...			
	INn	REAL	Input variable n	0.0
Output	OUT	REAL	Maximum value	0.0

9.7 MINn_R: Minimum of REAL values

Function

This block compares the inputs and outputs their minimum value.

OUT = MIN {IN1, ... , INn}

Blocks

Name	Description
MIN2_R	2 inputs of type REAL
MIN4_R	4 inputs of type REAL
MIN8_R	8 inputs of type REAL

I/Os

	Name	Data type	Description	Default setting
Inputs	IN1	REAL	Input variable 1	0.0
	...			
	INn	REAL	Input variable n	0.0
Output	OUT	REAL	Minimum value	0.0

9.8 ABS_R: Absolute value of REAL values

Function

This block outputs the absolute value of the input.

OUT = | IN |

I/Os

	Name	Data type	Description	Default setting
Inputs	IN	REAL	Input value	0.0
Output	OUT	REAL	Absolute value	0.0

9.9 SQRT: Square root

Function

This block calculates and outputs the square root of the input.

OUT = SQRT(IN)

Troubleshooting

ENO = 0 and OUT = 0 if IN < 0.

I/Os

	Name	Data type	Description	Default setting
Input	IN	REAL	Radicand	0.0
Output	OUT	REAL	Square root	0.0

9.10 EXP: Exponential function

Function

This block calculates and outputs the exponential function of the input. In this equation, "e" is Euler's constant 2.71…, the base of the natural logarithm.

OUT = e^IN

Troubleshooting

ENO = 0 on overflow and underflow.

ENO = 0 and OUT = 0 if IN < 0.

I/Os

	Name	Data type	Description	Default setting
Input	IN	REAL	Exponent	0.0
Output	OUT	REAL	Exponential function	0.0

9.11 POW10: Power-of-10 function

Function

This block calculates the power of 10^{IN} of the input and send the result to the output.

OUT = 10^IN

Troubleshooting

ENO = 0 with IN1 < -37.9 and IN1 > 38.5

I/Os

	Name	Data type	Description	Default setting
Input	IN	REAL	Exponent	0.0
Output	OUT	REAL	Power of ten	0.0

9.12 LN: Natural logarithm

Function

This block calculates and outputs the natural logarithm of the input.

OUT = LN (IN)

The variable at input IN must be positive.

Troubleshooting

ENO = 0 on overflow and underflow.

ENO = 0 and OUT = 0 if IN < 0.

I/Os

	Name	Data type	Description	Default setting
Input	IN	REAL	Argument	0.0
Output	OUT	REAL	nat. Logarithm	0.0

9.13 LOG10: Base-10 logarithm

Function

This block calculates and outputs the base-10 logarithm value of the input.

OUT = LOG10(IN)

The variable at input IN must be positive.

Troubleshooting

ENO = 0 on overflow and underflow.

ENO = 0 and OUT = 0 if IN < 0.

I/Os

	Name	Data type	Description	Default setting
Input	IN	REAL	Argument	0.0
Output	OUT	REAL	Logarithm	0.0

9.14 SIN: Sin function

Function

This block calculates and outputs the sine of the input. The variable at IN must be a radian value.

OUT = SIN(IN)

I/Os

	Name	Data type	Description	Default setting
Input	IN	REAL	Argument	0.0
Output	OUT	REAL	Sine	0.0

9.15 COS: Cos function

Function

This block calculates and outputs the cosine of the input. The variable at IN must be a radian value.

OUT = COS(IN)

I/Os

	Name	Data type	Description	Default setting
Input	IN	REAL	Argument	0.0
Output	OUT	REAL	Cosine	0.0

9.16 TAN: Tan function

Function

This block calculates and outputs the tangent of the input. The variable at IN must be a radian value.

OUT = TAN(IN)

I/Os

	Name	Data type	Description	Default setting
Input	IN	REAL	Argument	0.0
Output	OUT	REAL	Tangent	0.0

9.17 ASIN: Arc sin function

Function

This block calculates and outputs the arc sine of the input. The result is output as a radian value between -π/2 and +π/2. The range of the function argument must be between 1 and +1.

OUT = ASIN(IN)

I/Os

	Name	Data type	Description	Default setting
Input	IN	REAL	Argument	0.0
Output	OUT	REAL	Arc sine	0.0

9.18 ACOS: Arc cos function

Function

This block calculates and outputs the arc cosine of the input. The result is output as a radian value between 0 and π. The range of the function argument must be between -1 and +1.

OUT = ACOS(IN)

Troubleshooting

ENO = 0 at IN < -1 --> OUT = 3.14..

ENO = 0 at IN > 1 --> OUT = 0

I/Os

	Name	Data type	Description	Default setting
Input	IN	REAL	Argument	0.0
Output	OUT	REAL	Arc cosine	0.0

9.19 ATAN: Arc tan function

Function

This block calculates and outputs the arc tangent of the input. The result is output as a radian value between -π/2 and +π/2. The range of the function argument is the entire REAL number range.

OUT = ATAN(IN)

I/Os

	Name	Data type	Description	Default setting
Input	IN	REAL	Argument	0.0
Output	OUT	REAL	Arc tangent	0.0

9.20 NEG_R: Inverter for REAL values

Function

This block outputs the input variable with inverted sign.

I/Os

	Name	Data type	Description	Default setting
Input	IN	REAL	Input variable	0.0
Output	OUT	REAL	Output variable	0.0

9.21 LIM_R: Limiter of REAL values

Function

This block compares the input variables IN, MAX and MIN. It checks if IN is within or outside the intervals limited by MIN and MAX. If the low limit MIN of the interval is greater than or equal to the high limit MAX, the output OUT = MAX and the outputs OUTU and OUTL are set to 1. IN > MAX represents a violation of the high limit, OUT = MAX, OUTU = 1 and OUTL = 0. IN < MIN represents a violation of the low limit, OUT = MIN, OUTU = 0, OUTL = 1. If IN is between MIN and MAX, OUT = IN, OUTU = 0, OUTL = 0 are set.

If the low limit MIN is equal to the high limit, the block behaves as follows:

IN < MAX: OUT = MAX; OUTU = 0; OUTL = 1

IN > MAX: OUT = MAX; OUTU = 1; OUTL = 0

I/Os

	Name	Data type	Description	Default setting
Inputs	IN	REAL	Input variable	0.0
	MIN	REAL	Low limit	-100.0
	MAX	REAL	High limit	100.0
Outputs	OUT	REAL	Output variable	0
	OUTU	BOOL	Out of high limits	0
	OUTL	BOOL	Out of low limits	0

9.22 EPS_R: Accuracy approximation

Function

This block compares the absolute values of the inputs. If the absolute value of input IN < than the INTERVAL limit, output QA will be set to 1 and output QN to 0. In this case the input variable IN is contained in the interval. Otherwise output QA is set to 0 and output QN is set to 1. The input variable IN is then outside the interval.

I/Os

	Name	Data type	Description	Default setting
Inputs	IN	REAL	Input variable	0.0
	INTERVAL	REAL	Interval limit	0.0
Outputs	QA	BOOL	Validation bit memory	0
	QN	BOOL	Inverted validation bit memory	0

9.23 CADD_R: Controllable adder of REAL values

Function

This block adds input variable IN to output variable OUT, if input CI = 1 and inputs RI and SI = 0. If RI = 1, OUT = 0. If SI = 1 and RI = 0, then OUT = IN.

Troubleshooting

ENO = 0 on overflow and underflow.

Truth table

RI	SI	CI	OUT	ENO
1	X	X	0	1
0	1	X	IN	1
0	0	1	OUT* + IN	1
0	0	0	OUT*	1

X is a random value

OUT* is the old value from the previous cycle

I/Os

	Name	Data type	Description	Default setting
Inputs	IN	REAL	Addend	0.0
	RI	BOOL	Reset	0
	SI	BOOL	Set	0
	CI	BOOL	Count	0
Output	OUT	REAL	Sum	0.0

9.24 POWXY: General power function

Function

This block sends the input variable IN1 to the output, raised to the power of the input variable IN2.

OUT = IN1^IN2

Requirement: IN1 > 0

Troubleshooting

M7 goes to STOP with an overshoot or undershoot.

I/Os

	Name	Data type	Description	Default setting
Inputs	IN1	REAL	Base	0.0
	IN2	REAL	Exponent	0.0
Output	OUT	REAL	Output variable	0.0

9.25 SAMP_AVE: Floating average value

Function

This block outputs the average value of the last N input values.

OUT = (Ink + Ink -1 + ... + Ink -n + 1) / N

where INk is the current input value. For the number of input values N the condition

0 < N < 33

must be satisfied.

Startup characteristics

During startup and initial run each element of the buffer for IN and OUT values is set to 0.

I/Os

	Name	Data type	Description	Default setting
Inputs	IN	REAL	Input variable	0.0
	N	INT	Number of input values considered	1
Output	OUT	REAL	Average value	0.0

Arithmetic blocks of the data type INT and DINT 10

10.1 MATH_INT

CFC blocks of the "MATH_INT" family

This family contains the following blocks used for arithmetic operations with data of the INT and DINT types:

ADD_I: Addition of INT values (Page 99)	Addition of INT values
ADD_DI: Addition of DINT values (Page 111)	Addition of DINT values
SUB_I: Subtraction of INT values (Page 100)	Subtraction of INT values
SUB_DI: Subtraction of DINT values (Page 112)	Subtraction of DINT values
MUL_I: Multiplication of INT values (Page 101)	Multiplication of INT values
MUL_DI: Multiplication of DINT (Page 113)	Multiplication of DINT values
DIV_I: Division of INT values (Page 102)	Division of INT values
DIV_DI: Division of DINT values (Page 114)	Division of DINT values
ABS_I Absolute value of INT values (Page 106)	Absolute value of INT values
ABS_DI: Absolute value of DINT values (Page 118)	Absolute value of DINT values
EPS_I: Accuracy approximation of INT values (Page 109)	Accuracy; approximation of INT values
EPS_DI: Accuracy approximation of DINT values (Page 121)	Accuracy; approximation of DINT values
NEG_I: Inverter for INT values (Page 107)	Inverter for INT values
NEG_DI: Inverter for DINT values (Page 119)	Inverter for DINT values
MOD_I: Modulo function of INT values (Page 103)	Modulo function of INT values
MOD_DI: Modulo function of DINT values (Page 115)	Modulo function of DINT values

MAXn_I: Maximum of INT values (Page 104)	Maximum of INT values
MAXn_DI: Maximum of DINT values (Page 116)	Maximum of DINT values
MINn_I: Minimum of INT values (Page 105)	Minimum of INT values
MINn_DI: Minimum of DINT values (Page 117)	Minimum of DINT values
LIM_I: Limiter for INT values (Page 108)	Limiter for INT values
LIM_DI: Limiter for DINT values (Page 120)	Limiter for DINT values
CADD_I: Controllable adder of INT values (Page 110)	Controllable adder of INT values
CADD_DI: Controllable adder of DINT values (Page 122)	Controllable adder of DINT values

Note

The value range of data types INT and DINT is

INT: -32 768 ... 32 767

DINT: -2 147 483 648 ... 2 147 483 647

10.2 ADD_I: Addition of INT values

Function

This block adds the values of inputs and outputs the sum.

OUT = IN1 + IN2

Troubleshooting

ENO = 0 on overflow and underflow.

I/Os

	Name	Data type	Description	Default setting
Inputs	IN1	INT	Addend 1	0
	IN2	INT	Addend 2	0
Output	OUT	INT	Sum	0

10.3 SUB_I: Subtraction of INT values

Function

This block subtracts input IN2 from input IN1and outputs the difference.

OUT = IN1 - IN2

Troubleshooting

ENO = 0 on overflow and underflow.

I/Os

	Name	Data type	Description	Default setting
Inputs	IN1	INT	Minuend	0
	IN2	INT	Subtrahend	0
Output	OUT	INT	Difference	0

10.4 MUL_I: Multiplication of INT values

Function

This block multiplies the inputs and outputs the product.

OUT = IN1 * IN2

Troubleshooting

ENO = 0 on overflow and underflow.

I/Os

	Name	Data type	Description	Default setting
Inputs	IN1	INT	Multiplicand	0
	IN2	INT	Multiplier	0
Output	OUT	INT	Product	0

10.5 DIV_I: Division of INT values

Function

This block divides the value at input IN1 by the value at input IN2 and outputs the quotient.

OUT = IN1 / IN2

Troubleshooting

ENO = 0 in the case of division by 0 and –32768 in the case of division by –1.

I/Os

	Name	Data type	Description	Default setting
Inputs	IN1	INT	Dividend	0
	IN2	INT	Divisor	0
Output	OUT	INT	Quotient	0

10.6 MOD_I: Modulo function of INT values

Function

This block outputs the modulo of an integer division DIV_I (Page 102) of input IN1 by input IN2.

Troubleshooting

ENO = 0 in the case of division by 0.

I/Os

	Name	Data type	Description	Default setting
Inputs	IN1	INT	Dividend	0
	IN2	INT	Divisor	1
Output	OUT	INT	Remainder	0

10.7 MAXn_I: Maximum of INT values

Function

This block compares the inputs and outputs their maximum value.

OUT = MAX {IN1, ... , INn}

Blocks

Name	Description
MAX2_I	2 inputs of the type INT
MAX4_I	4 inputs of the type INT
MAX8_I	8 inputs of the type INT

I/Os

	Name	Data type	Description	Default setting
Inputs	IN1	INT	Input variable 1	0
	...			
	INn	INT	Input variable n	0
Output	OUT	INT	Maximum value	0

10.8 MINn_I: Minimum of INT values

Function

This block compares the inputs and outputs their minimum value.

OUT = MIN {IN1, ... , INn}

Blocks

Name	Description
MIN2_I	2 inputs of the type INT
MIN4_I	4 inputs of the type INT
MIN8_I	8 inputs of the type INT

I/Os

	Name	Data type	Description	Default setting
Inputs	IN1	INT	Input variable 1	0
	...			
	INn	INT	Input variable n	0
Output	OUT	INT	Minimum value	0

10.9 ABS_I Absolute value of INT values

Function

This block outputs the absolute value of the input.

OUT = | IN |

Troubleshooting

ENO = 0 if IN = -32 768

I/Os

	Name	Data type	Description	Default setting
Input	IN	INT	Input value	0
Output	OUT	INT	Absolute value	0

10.10 NEG_I: Inverter for INT values

Function

This block outputs the input variable with inverted sign.

Troubleshooting

ENO = 0 if IN = -32 768

I/Os

	Name	Data type	Description	Default setting
Input	IN	INT	Input variable	0
Output	OUT	INT	Output variable	0

10.11 LIM_I: Limiter for INT values

Function

This block compares the input variables IN, MAX and MIN. It checks if IN is within or outside the intervals limited by MIN and MAX.

If the low limit MIN of the interval is greater than the high limit MAX, the output OUT = MAX and the outputs OUTU and OUTL are set to 1.

IN ≥ MAX represents a violation of the high limit, OUT = MAX, OUTU = 1 and OUTL = 0. IN ≤ MIN represents a violation of the low limit, OUT = MIN, OUTU = 0, OUTL = 1. If IN is between MIN and MAX, OUT = IN, OUTU = 0, OUTL = 0 are set.

If MAX = MIN, the outputs OUTU and OUTL will depend on IN:

- OUTU = 1 with IN = MIN = MAX
- OUTL = 1 with IN < MIN = MAX
- OUTU = 1 with IN > Max = MAX
- In addition: OUT = MAX = MIN

Troubleshooting

ENO = 0 with MIN > MAX --> OUT = MAX; OUTU = OUTL = 1

I/Os

	Name	Data type	Description	Default setting
Inputs	MAX	INT	High limit	0
	IN	INT	Input variable	0
	MIN	INT	Low limit	0
Outputs	OUTU	BOOL	Out of high limits	0
	OUTL	BOOL	Out of low limits	0
	OUT	INT	Output variable	0

10.12 EPS_I: Accuracy approximation of INT values

Function

This block compares the absolute value of the input IN with the value of the INTERVAL input. If the absolute amount of input IN < than the INTERVAL limit, output QA will be set to 1 and output QN to 0. In this case the input variable IN is contained in the interval. Otherwise output QA is set to 0 and output QN is set to 1. The input variable IN is then outside the interval.

INTERVAL must have a positive value.
If INTERVAL ≤ 0, then QA = 0.

Troubleshooting

ENO = 0 with IN = -32 768

I/Os

	Name	Data type	Description	Default setting
Inputs	IN	INT	Input variable	0
	INTERVAL	INT	Interval limit	0
Outputs	QA	BOOL	Validation bit memory	0
	QN	BOOL	Inverted validation bit memory	0

10.13 CADD_I: Controllable adder of INT values

Function

This block adds the input variable IN to the output variable OUT when the CI input is set to 1 and the RI and SI inputs are set to 0. If RI = 1, the OUT output is set to 0. If SI = 1 and RI = 0, then OUT = IN.

Troubleshooting

ENO = 0 on overflow and underflow.

Truth table

RI	SI	CI	OUT	ENO
1	X	X	0	1
0	1	X	IN	1
0	0	1	OUT* + IN	1
0	0	0	OUT*	1

X is any value

OUT* is the old value from the last cycle

I/Os

	Name	Data type	Description	Default setting
Inputs	IN	INT	Addend	0
	RI	BOOL	Reset	0
	SI	BOOL	Set	0
	CI	BOOL	Count	0
Output	OUT	INT	Sum	0

10.14 ADD_DI: Addition of DINT values

Function

This block adds the values of inputs and outputs the sum.

OUT = IN1 + IN2

Troubleshooting

ENO = 0 on overflow and underflow.

I/Os

	Name	Data type	Description	Default setting
Inputs	IN1	DINT	Addend 1	0
	IN2	DINT	Addend 2	0
Output	OUT	DINT	Sum	0

10.15 SUB_DI: Subtraction of DINT values

Function

This block subtracts the value at input IN2 from input IN1 and outputs the difference.

OUT = IN1 - IN2

Troubleshooting

ENO = 0 on overflow and underflow.

I/Os

	Name	Data type	Description	Default setting
Inputs	IN1	DINT	Minuend	0
	IN2	DINT	Subtrahend	0
Output	OUT	DINT	Difference	0

10.16 MUL_DI: Multiplication of DINT

Function

This block multiplies the inputs and outputs the product.

OUT = IN1 * IN2

Troubleshooting

ENO = 0 on overflow and underflow.

I/Os

	Name	Data type	Description	Default setting
Inputs	IN1	DINT	Multiplicand	0
	IN2	DINT	Multiplier	0
Output	OUT	DINT	Product	0

10.17 DIV_DI: Division of DINT values

Function

This block divides the value at input IN1 by the value at input IN2 and outputs the quotient.

OUT = IN1 / IN2

Troubleshooting

ENO = 0 in the case of division by 0 and –2147483648 in the case of division by –1.

I/Os

	Name	Data type	Description	Default setting
Inputs	IN1	DINT	Dividend	0
	IN2	DINT	Divisor	0
Output	OUT	DINT	Quotient	0

10.18 MOD_DI: Modulo function of DINT values

Function

This block outputs the modulo of an integer division DIV_DI (Page 114) of input IN1 by input IN2.

Troubleshooting

ENO = 0 in the case of division by 0.

I/Os

	Name	Data type	Description	Default setting
Inputs	IN1	DINT	Dividend	0
	IN2	DINT	Divisor	0
Output	OUT	DINT	Remainder	0

10.19 MAXn_DI: Maximum of DINT values

Function

This block compares the inputs and outputs their maximum value.

OUT = MAX {IN1, ... , INn}

Blocks

Name	Description
MAX2_DI	2 inputs of the type DINT
MAX4_DI	4 inputs of the type DINT
MAX8_DI	8 inputs of the type DINT

I/Os

	Name	Data type	Description	Default setting
Inputs	IN1	DINT	Input variable 1	0
	...			
	INn	DINT	Input variable n	0
Output	OUT	DINT	Maximum value	0

10.20 MINn_DI: Minimum of DINT values

Function

This block compares the inputs and outputs their minimum value.

OUT = MIN {IN1, ... , INn}

Blocks

Name	Description
MIN2_DI	2 inputs of the type DINT
MIN4_DI	4 inputs of the type DINT
MIN8_DI	8 inputs of the type DINT

I/Os

	Name	Data type	Description	Default setting
Inputs	IN1	DINT	Input variable 1	0
	...			
	INn	DINT	Input variable n	0
Output	OUT	DINT	Minimum value	0

10.21 ABS_DI: Absolute value of DINT values

Function

This block outputs the absolute value of the input.

OUT = | IN |

Troubleshooting

ENO = 0 if IN = -2 147 483 648 (smallest negative number)

I/Os

	Name	Data type	Description	Default setting
Input	IN	DINT	Input value	0
Output	OUT	DINT	Absolute value	0

10.22 NEG_DI: Inverter for DINT values

Function

This block outputs the input variable with inverted sign.

Troubleshooting

ENO = 0 if IN = -2 147 483 648

I/Os

	Name	Data type	Description	Default setting
Input	IN	DINT	Input variable	0
Output	OUT	DINT	Output variable	0

10.23 LIM_DI: Limiter for DINT values

Function

This block compares the input variables IN, MAX and MIN. It checks if IN is within or outside the intervals limited by MIN and MAX.

If the low limit MIN of the interval is greater than the high limit MAX, the output OUT = MAX and the outputs OUTU and OUTL are set to 1.

IN ≥ MAX represents a violation of the high limit, OUT = MAX, OUTU = 1 and OUTL = 0. IN ≤ MIN represents a violation of the low limit, OUT = MIN, OUTU = 0, OUTL = 1. If IN is between MIN and MAX, then OUT = IN, OUTU = 0, OUTL = 0 are set.

If MAX = MIN, the outputs OUTU and OUTL will depend on IN:

- OUTU = 1 with IN = MIN = MAX
- OUTL = 1 with IN < MIN = MAX
- OUTU = 1 with IN > Max = MAX
- In addition: OUT = MAX = MIN

Troubleshooting

ENO = 0 with MIN > MAX --> OUT = MAX; OUTU = OUTL = 1

I/Os

	Name	Data type	Description	Default setting
Inputs	MAX	DINT	High limit	0
	IN	DINT	Input variable	0
	MIN	DINT	Low limit	0
Outputs	OUTU	BOOL	Out of high limits	0
	OUTL	BOOL	Out of low limits	0
	OUT	DINT	Output variable	0

10.24 EPS_DI: Accuracy approximation of DINT values

Function

This block compares the absolute value of the input IN with the value of the INTERVAL input. If the absolute value of the IN input is less than the INTERVAL limit, the QA output is set to 1 and the QN output is set to 0. In this case the input variable IN is contained in the interval. Otherwise output QA is set to 0 and output QN is set to 1. In this case the input variable IN is outside of the range of the interval.

INTERVAL must have a positive value.
If INTERVAL is ≤ 0 , then QA = 0.

Troubleshooting

ENO = 0 with IN = -2 147 483 648

I/Os

	Name	Data type	Description	Default setting
Inputs	IN	DINT	Input variable	0
	INTERVAL	DINT	Interval limit	0
Outputs	QA	BOOL	Validation bit memory	0
	QN	BOOL	Inverted validation bit memory	0

10.25 CADD_DI: Controllable adder of DINT values

Function

This block adds the input variable IN to the output variable OUT when the CI input is set to 1 and the RI and SI inputs are set to 0. If RI = 1, the OUT output is set to 0. If SI = 1 and RI = 0, then OUT = IN.

Troubleshooting

ENO = 0 on overflow and underflow.

Truth table

RI	SI	CI	OUT	ENO
1	X	X	0	1
0	1	X	IN	1
0	0	1	OUT* + IN	1
0	0	0	OUT*	1

X is any value

OUT* is the old value from the last cycle

I/Os

	Name	Data type	Description	Default setting
Inputs	IN	DINT	Addend	0
	RI	BOOL	Reset	0
	SI	BOOL	Set	0
	CI	BOOL	Count	0
Output	OUT	DINT	Sum	0

Flip-flop blocks

11

11.1 FLIPFLOP

CFC blocks of the "FLIPFLOP" family

This family contains the following flip-flop blocks:

JK_FF: JK flip-flop (Page 124)	JK flip-flop
RS_FF: RS flip-flop, reset dominant (Page 125)	RS flip-flop, reset dominant
SR_FF: SR flip-flop, set dominant (Page 126)	SR flip-flop, set dominant

11.2 JK_FF: JK flip-flop

Function

J	K	Q_n	$\overline{Q}_n$
0	0	Q_{n-1}	$\overline{Q}_{n-1}$
0	1	0	1
1	0	1	0
1	1	$\overline{Q}_{n-1}$	Q_{n-1} *)

*) The outputs toggle the value in the cycle of the inserted tasks.

I/Os

	Name	Data type	Description	Default setting
Inputs	J	BOOL	Set	0
	K	BOOL	Reset	0
Outputs	Q	BOOL	Output	0
	$\overline{Q}$	BOOL	Negated output	1

11.3 RS_FF: RS flip-flop, reset dominant

Function

R	S	Q_n	$\overline{Q}_n$
0	0	Q_{n-1}	$\overline{Q}_{n-1}$
0	1	1	0
1	0	0	1
1	1	0	1

I/Os

	Name	Data type	Description	Default setting
Inputs	R	BOOL	Reset	0
	S	BOOL	Set	0
Outputs	Q	BOOL	Output	0
	$\overline{Q}$	BOOL	Negated output	1

11.4 SR_FF: SR flip-flop, set dominant

Function

R	S	Q_n	$\overline{Q}_n$
0	0	Q_{n-1}	$\overline{Q}_{n-1}$
0	1	1	0
1	0	0	1
1	1	1	0

I/Os

	Name	Data type	Description	Default setting
Inputs	R	BOOL	Reset	0
	S	BOOL	Set	0
Outputs	Q	BOOL	Output	0
	$\overline{Q}$	BOOL	Negated output	1

Shift blocks

12

12.1 SHIFT

CFC blocks of the "SHIFT" family

This family contains the following blocks to shift or rotate input-value bits and output the result:

SHL_W: WORD shift left (Page 128)	WORD shift left
SHL_DW: DWORD shift left (Page 129)	DWORD shift left
SHR_W: WORD shift right (Page 130)	WORD shift right
SHR_DW: DWORD shift right (Page 131)	DWORD shift right
ROL_W: WORD rotate left (Page 132)	WORD rotate left
ROL_DW: DWORD rotate left (Page 133)	DWORD rotate left
ROR_W: WORD rotate right (Page 134)	WORD rotate right
ROR_DW: DWORD rotate right (Page 135)	DWORD rotate right

12.2 SHL_W: WORD shift left

Function

The bits of input value IN are shifted left by the number of steps specified in input IN. The result is displayed at the output.

I/Os

	Name	Data type	Description	Default setting
Inputs	IN	WORD	Input value	0
	N	WORD	Number of steps	0
Output	OUT	WORD	Output	0

12.3 SHL_DW: DWORD shift left

Function

The bits of input value IN are shifted left by the number of steps specified in input IN. The result is displayed at the output.

I/Os

	Name	Data type	Description	Default setting
Inputs	IN	DWORD	Input value	0
	N	WORD	Number of steps	0
Output	OUT	DWORD	Output	0

12.4 SHR_W: WORD shift right

Function

The bits of input value IN are shifted right by the number of steps specified in input IN. The result is displayed at the output.

I/Os

	Name	Data type	Description	Default setting
Inputs	IN	WORD	Input value	0
	N	WORD	Number of steps	0
Output	OUT	WORD	Output	0

12.5 SHR_DW: DWORD shift right

Function

The bits of input value IN are shifted right by the number of steps specified in input IN. The result is displayed at the output.

I/Os

	Name	Data type	Description	Default setting
Inputs	IN	DWORD	Input value	0
	N	WORD	Number of steps	0
Output	OUT	DWORD	Output	0

12.6 ROL_W: WORD rotate left

Function

The bits of input value IN are rotated left by the number of steps specified in input IN. The result is displayed at the output.

I/Os

	Name	Data type	Description	Default setting
Inputs	IN	WORD	Input value	0
	N	WORD	Number of rotations	0
Output	OUT	WORD	Output	0

12.7 ROL_DW: DWORD rotate left

Function

The bits of input value IN are rotated left by the number of steps specified in input IN. The result is displayed at the output.

I/Os

	Name	Data type	Description	Default setting
Inputs	IN	DWORD	Input value	0
	N	WORD	Number of rotations	0
Output	OUT	DWORD	Output	0

12.8 ROR_W: WORD rotate right

Function

The bits of input value IN are rotated right by the number of steps specified in input IN. The result is displayed at the output.

I/Os

	Name	Data type	Description	Default setting
Inputs	IN	WORD	Input value	0
	N	WORD	Number of rotations	0
Output	OUT	WORD	Output	0

12.9 ROR_DW: DWORD rotate right

Function

The bits of input value IN are rotated right by the number of steps specified in input IN. The result is displayed at the output.

I/Os

	Name	Data type	Description	Default setting
Inputs	IN	DWORD	Input value	0
	N	WORD	Number of rotations	0
Output	OUT	DWORD	Output	0

Multiplexer blocks

13

13.1 MULTIPLX

CFC blocks of the "MULTIPLX" family

This family contains the following blocks used to set one of n inputs to the output, depending on the value at a specific input:

MUXn_I: Multiplexer 1 of n for INT values (Page 138)	Multiplexer 1 of n for INT values (n = 2, 4, 8)
MUXn_DI: Multiplexer 1 of n for DINT values (Page 139)	Multiplexer 1 of n for DINT values (n = 2, 4, 8)
MUXn_R: Multiplexer 1 of n for REAL values (Page 140)	Multiplexer 1 of n for REAL values (n = 2, 4, 8)
MUXn_BO: Multiplexer 1 of n for BOOL values (Page 141)	Multiplexer 1 of n for BOOL values (n = 2, 4, 8)
SEL_BO: Multiplexer 1 of 2 for BOOL values (Page 142)	Multiplexer 1 of 2 for BOOL values
SEL_R: Multiplexer 1 of 2 for REAL values (Page 143)	Multiplexer 1 of 2 for REAL values

13.2 MUXn_I: Multiplexer 1 of n for INT values

Function

The block is a multiplexer 1 of n for INT values (n = 2, 4, 8). One of the inputs IN0...IN7 is set to the output, depending on the value at selection input K.

Troubleshooting

ENO = 0 and OUT = 0, if k > (n-1) or k < 0.

Function table

Number of inputs									
2	K:	0	1						
	OUT:	IN0	IN1						
4	K:	0	1	2	3				
	OUT:	IN0	IN1	IN2	IN3				
8	K:	0	1	2	3	4	5	6	7
	OUT:	IN0	IN1	IN2	IN3	IN4	IN5	IN6	IN7

I/Os

	Name	Data type	Description	Default setting
Inputs	K	INT	Selection input	0
	IN0	INT	Value 1	0
	...		...	
	INm (n-1)	INT	Value n	0
Output	OUT	INT	Output	0

13.3 MUXn_DI: Multiplexer 1 of n for DINT values

Function

The block is a multiplexer 1 of n for DINT values (n = 2, 4, 8). One of the inputs IN0...IN7 is set to the output, depending on the value at selection input K.

Troubleshooting

ENO = 0 and OUT = 0, if k > (n-1) or k < 0.

Function table

Number of inputs									
2	K:	0	1						
	OUT:	IN0	IN1						
4	K:	0	1	2	3				
	OUT:	IN0	IN1	IN2	IN3				
8	K:	0	1	2	3	4	5	6	7
	OUT:	IN0	IN1	IN2	IN3	IN4	IN5	IN6	IN7

I/Os

	Name	Data type	Description	Default setting
Inputs	K	INT	Selection input	0
	IN0	DINT	Value 1	0
	...		...	
	INm (n-1)	DINT	Value n	0
Output	OUT	DINT	Output	0

13.4 MUXn_R: Multiplexer 1 of n for REAL values

Function

The block is a multiplexer 1 of n for REAL values (n = 2, 4, 8). One of the inputs IN0...IN7 is set to the output, depending on the value at selection input K.

Troubleshooting

ENO = 0 and OUT = 0, if k > (n-1) or k < 0.

Function table

Number of inputs									
2	K:	0	1						
	OUT:	IN0	IN1						
4	K:	0	1	2	3				
	OUT:	IN0	IN1	IN2	IN3				
8	K:	0	1	2	3	4	5	6	7
	OUT:	IN0	IN1	IN2	IN3	IN4	IN5	IN6	IN7

I/Os

	Name	Data type	Description	Default setting
Inputs	K	INT	Selection input	0
	IN1	REAL	Value 1	0
	...		...	
	INm	REAL	Value m (m=n-1)	0
Output	OUT	REAL	Output	0

13.5 MUXn_BO: Multiplexer 1 of n for BOOL values

Function

The block is a multiplexer 1 of n for BOOL values (n = 2, 4, 8). One of the inputs IN0...IN7 is set to the output, depending on the value at selection input K.

Troubleshooting

ENO = 0 and OUT = 0, if k > (n-1) or k < 0.

Function table

Number of inputs									
2	K:	0	1						
	OUT:	IN0	IN1						
4	K:	0	1	2	3				
	OUT:	IN0	IN1	IN2	IN3				
8	K:	0	1	2	3	4	5	6	7
	OUT:	IN0	IN1	IN2	IN3	IN4	IN5	IN6	IN7

I/Os

	Name	Data type	Description	Default setting
Inputs	K	INT	Selection input	0
	IN0	BOOL	Value 1	0
	...		...	
	INm (n-1)	BOOL	Value n	0
Output	OUT	BOOL	Output	0

13.6 SEL_BO: Multiplexer 1 of 2 for BOOL values

Function

This block sets the value of input IN0 (K = 1) or IN1 (K = 0) at the output, depending on the value at input K.

I/Os

	Name	Data type	Default setting
Inputs	K	BOOL	0
	IN0	BOOL	0
	IN1	BOOL	0
Output	OUT	BOOL	0

13.7 SEL_R: Multiplexer 1 of 2 for REAL values

Function

This block sets the value of input IN0 (K = 1) or IN1 (K = 0) at the output, depending on the value at input K.

I/Os

	Name	Data type	Default setting
Inputs	K	BOOL	0
	IN0	REAL	0.0
	IN1	REAL	0.0
Output	OUT	REAL	0.0

Counter blocks

14

14.1 COUNTER

CFC blocks of the "COUNTER" family

This family contains the following counter blocks:

CTU: Up-counter (Page 146)	Up counter
CTD: Down-counter (Page 148)	Down counter
CTUD: Up/down-counter (Page 149)	Up/down counter

14.2 CTU: Up-counter

Function

This block is an edge-triggered up counter. In the case of a rising edge, the counter has the default value PV at input S. A rising edge at input CU increments the counter. The counter value is output at CV. The counter is stopped when the maximum INT value (32767) is reached. A reset signal sets the counter to 0 and restarts the up count.

Q = 0, if CV = 0,

Q = 1 if CV > 0

Startup characteristics

The block's startup behavior is identical to its reset behavior (CV = 0).

Truth table

R	CU	CV	ENO
1	X	0	1
0	1	CV*+1	1
0	0	CV*	1

X is a random value

CV* is the old value from the previous cycle

I/Os

	Name	Data type	Description	Default setting
Inputs	CU	BOOL	Up-count pulse	0
	R	BOOL	Reset	0
	W	BOOL	Set (load)	0
	PV	INT	Load value	1000
Outputs	Q	BOOL	Overflow	0
	CV	INT	Counter value	0

Note

This block exists twice: In the standard library as SFB0 and in the CFC elementary library as FB24. Both blocks have different behavior:

- The SFB0 type possesses no preassigned value. The PV input is compared with the CV output for reaching / exceeding. The Q output indicates whether the current counter value is greater than or equal to PV.
- The FB24 has input PV as preassigned value. Counting starts as of this value. The output Q is not evaluated.

14.3 CTD: Down-counter

Function

This block is an edge-triggered down-counter. On a rising edge at input S, the value PV is written to the counter. A rising edge at input CD decrements the counter. The counter value is output at CV. The counter is stopped when the minimum INT value is reached. A reset signal sets the counter to 0.

Q = 0, if CV = 0

Q = 1 if CV > 0

The counter is stopped when CV = 0 is reached. There is no counting in the negative range.

Startup characteristics

The block's startup behavior is identical to its reset behavior (CV = 0).

Truth table

R	CD	CV	ENO
1	X	0	1
0	1	CV*-1	1
0	0	CV*	1

X is a random value

CV* is the old value from the previous cycle

I/Os

	Name	Data type	Description	Default setting
Inputs	CD	BOOL	Down-count pulse	0
	R	BOOL	Reset	0
	W	BOOL	Set (load)	0
	PV	INT	Load value	1000
Outputs	Q	BOOL	Underflow	0
	CV	INT	Counter value	0

14.4 CTUD: Up/down-counter

Function

This block is an edge-triggered up/down counter. On a rising edge at input S, the value PV is written to the counter. A rising edge at input CU increments the counter. A rising edge at input CD decrements the counter. The counter value is output at CV. Outputs QU or QD can be used to monitor the counter value. The counter is reset with R=1.

QU = 1, if CV < max. INT (32767)

QU = 0 if CV = max. INT (32767)

QD = 0 if CV = – max. INT (-32768)

QD = 1, if CV > - max. INT (-32768)

Startup characteristics

The block's startup behavior is identical to its reset behavior (CV = 0).

Truth table

R	CU	CD	CV	ENO
1	X	X	0	1
0	1	0	CV*+1	1
0	0	1	CV*-1	1
0	0	0	CV*	1
0	1	1	CV*	1

X is a random value

CV* is the old value from the previous cycle

I/Os

	Name	Data type	Description	Default setting
Inputs	CU	BOOL	Up-count pulse	0
	CD	BOOL	Down-count pulse	0
	R	BOOL	Reset	0
	W	BOOL	Set (load)	0
	PV	INT	Load value	1000
Outputs	QU	BOOL	Counter at high limit	0
	QD	BOOL	Counter at low limit	0
	CV	INT	Counter value	0

Blocks for generating or processing pulses 15

15.1 PULSE

CFC blocks of the "PULSE" family

This family contains the following pulse-processing blocks:

TIMER_P: Pulse generator (Page 152)	Pulse generator
R_TRIG: Detection of the rising edge (Page 155)	Detection of the rising edge
F_TRIG: Detection of the falling edge (Page 156)	Detection of the falling edge
AFP: Clock (Page 157)	Clock

15.2 TIMER_P: Pulse generator

Function

The block starts the timer in the operating mode set by the value at the MODE input:

- Pulse generator
- Extended pulse
- ON delay
- ON delay with memory
- OFF delay

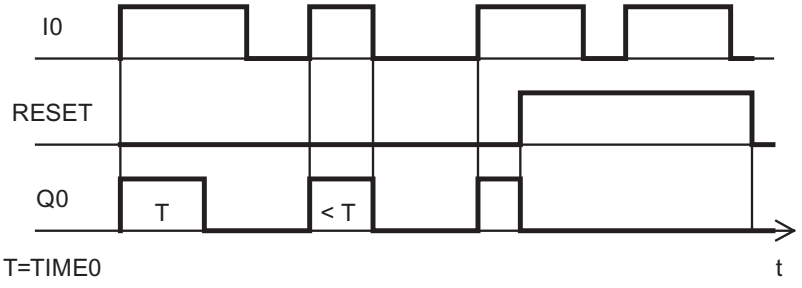
Operating modes

MODE	Operating mode
0	Start pulse forming timer
1	Start extended pulse timer
2	Start ON delay timer
3	Start ON delay timer with memory
4	Start OFF delay timer

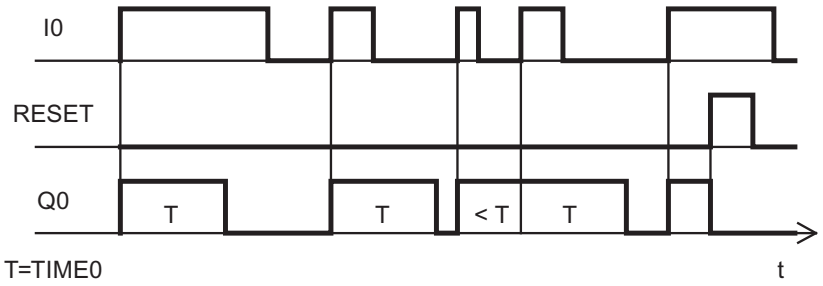
The block only applies the operating mode (MODE) on a rising edge at input I0. This edge change is also necessary following a CPU restart. The timer counter PTIME is loaded with the value TIME0 and decremented periodically by the sampling time SAMPLE_T. The status at the output Q0 changes according to the set MODE once the time has expired. The values at outputs Q0 = 0 and PTIME = 0 are output via RESET = 1.

Pulse diagrams

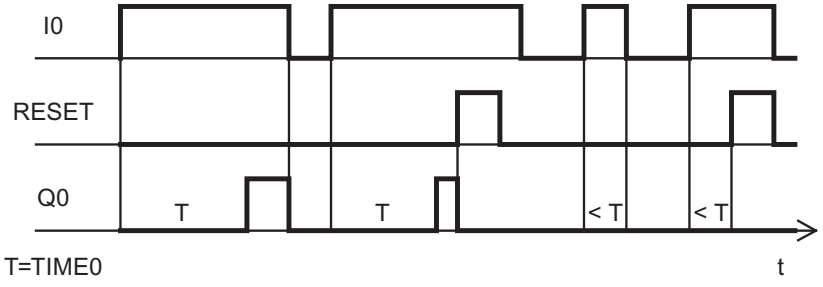
MODE=0 pulse



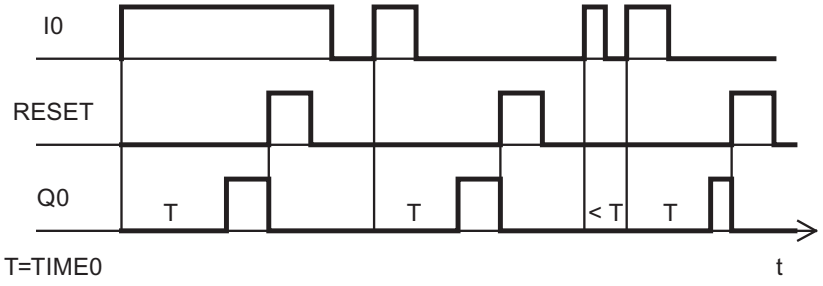
MODE=1 extended pulse



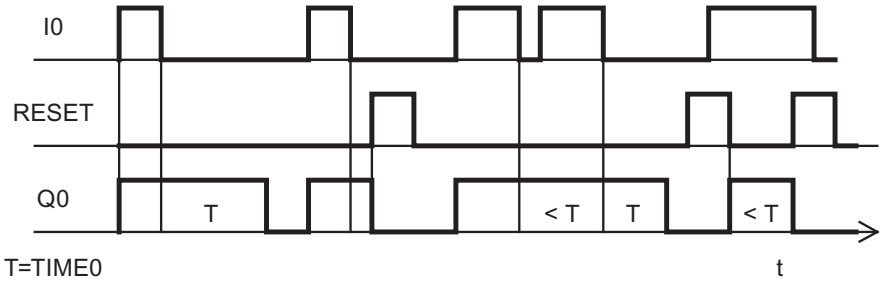
MODE=2 ON delay



MODE=3 ON delay with memory



MODE=4 OFF delay



Note the following when entering values:

- The sampling time (SAMPLE_T) must be less than the switching time (TIME0).
- The difference between TIME0 and SAMPLE_T may not be more than 10^7.

I/Os

Note

Cumulation of sampling time SAMPL_T

The TIMER_P block works internally with REAL values. Whenever the sampling time SAMPL_T cannot be displayed exactly as a REAL number, inaccuracies occur in cumulation.

The sampling time can only be displayed exactly as a REAL number when the decimal places of the sampling time occur through division by 2, 4, 8, 16 etc.

Examples of exact sampling times: x.5, x.25, x.125, x.0625 etc.

Examples of inexact sampling times: x.1, x.2

"x" represents a positive integer here.

	Name	Data type	Description	Default setting
Inputs	SAMPLE_T	REAL	Task sampling time in s	1.0
	TIME0	REAL	Time in s	0.0
	MODE	INT	Operating mode (see above)	0
	RESET	BOOL	Reset	0
	I0	BOOL	Input pulse	0
Outputs	QERR	BOOL	Error	1
	Q0	BOOL	Output pulse	0
	PTIME	REAL	Time-to-go	0.0

15.3 R_TRIG: Detection of the rising edge

Note

In order to ensure proper operation of the R_TRIG block, it must be installed in a watchdog interrupt (cyclic task).

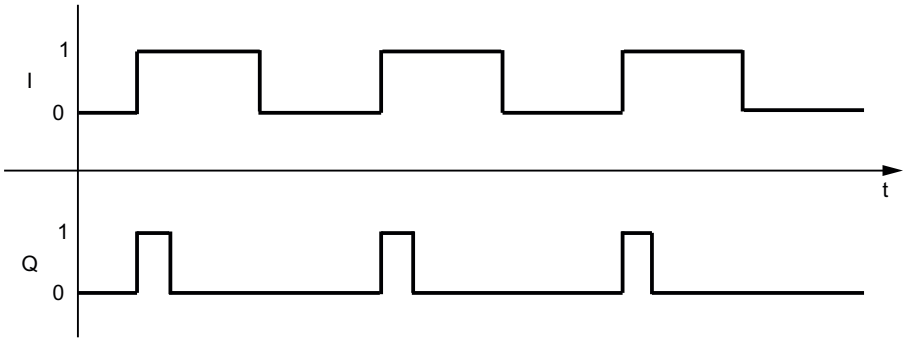
Function

This block checks for a rising edge at the input variable and outputs the result. Output Q = 1 if a rising edge is detected at pulse input CLK.

Startup characteristics

The edge bit memory is set to 0 on startup.

Pulse diagram



I/Os

	Name	Data type	Description	Default setting
Input	CLK	BOOL	Input pulse	0
Output	Q	BOOL	Output pulse	0

15.4 F_TRIG: Detection of the falling edge

Note

To operate correctly, the F_TRIG block must be installed in a cyclic interrupt (cyclic task).

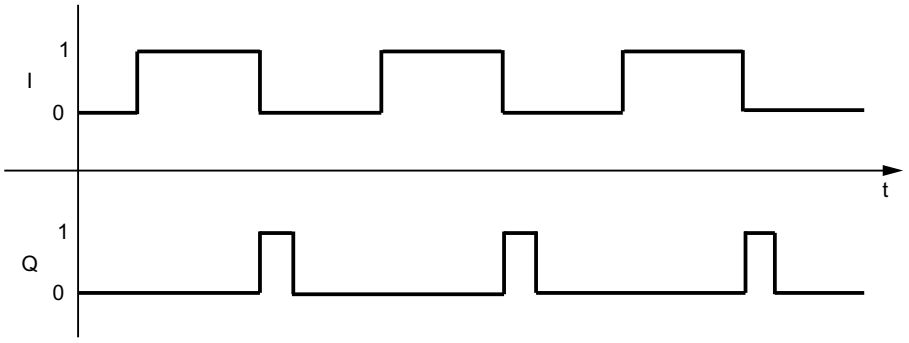
Function

This block checks for a falling edge at the input variable and outputs the result. Output Q = 1 if a falling edge is detected at pulse input CLK.

Startup characteristics

The edge bit memory is set to 1 on startup.

Pulse diagram



I/Os

	Name	Data type	Description	Default setting
Input	CLK	BOOL	Input pulse	0
Output	Q	BOOL	Output pulse	0

15.5 AFP: Clock

Note

To operate correctly, the AFP block must be installed in a cyclic interrupt (cyclic task).

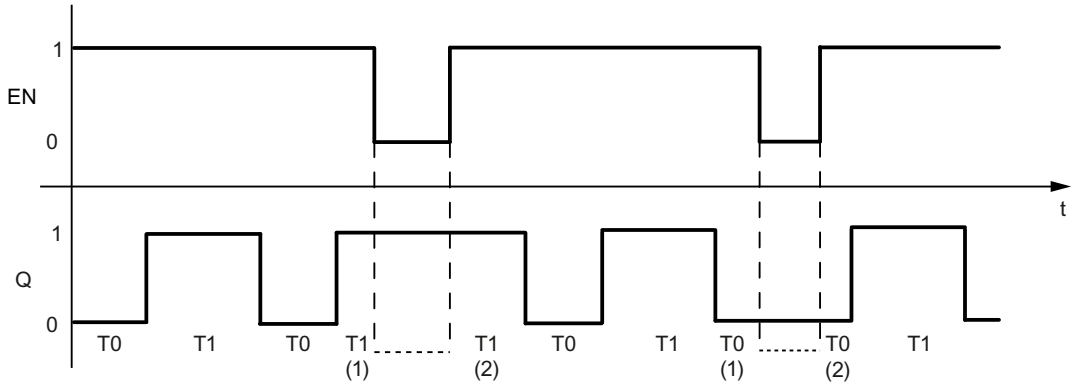
Function

Clock. This block generates pulses according to pulse/break ratio that is to be configured. This ratio can be configured and is specified in ms units.

Startup characteristics

During startup, the counters and enable bits are set for the periods Q = 0 and Q = 1. Whereby the enable bit for Q = 0 will be set to 1, the other values to 0.

Pulse diagram



If EN = 0, output Q remains in the current state. The expired time of T0 or T1 is stopped (T0(1) or T1(1) in the diagram) and not reset.

If EN = 1, the remaining time runs from T0 or T1 (T0(2) or T1(2) in the diagram).

I/Os

	Name	Data type	Description	Default setting
Inputs	SAMPLE_T	REAL	Task sampling time in s	1.0
	T0	TIME	Interpulse period	T#0ms
	T1	TIME	Pulse duration	T#0ms
Output	Q	BOOL	Output pulse	0

Blocks for acquiring or processing time intervals and timebases

16

16.1 TIME

CFC blocks of the family

This family contains the following timer blocks:

TIME: Measures the execution time (Page 160)	Measures the execution time
TIME_BEG: Outputs the current time of day (Page 161)	Outputs the current time of day
TIME_END: Compares the input time with the actual time (Page 162)	Compares the input time with the actual time

16.2 TIME: Measures the execution time

Function

This block measures the time between two calls (maximum: 2 147 483 647 ms).

Note

There must be no date change between the calls otherwise there would be a negative time difference (00:00:00 – measured time).

I/Os

	Name	Data type	Description	Default setting
Inputs	DIFF	BOOL	Difference measurement ON	1
Output	OUT	TIME	Time	

16.3 TIME_BEG: Outputs the current time of day

Function

This block outputs the system time at which the block is called at output TM.

I/Os

	Name	Data type	Description	Default setting
Output	TM	TIME	Current time	T#0ms

16.4 TIME_END: Compares the input time with the actual time

Function

This block indicates the time difference between input TM and the actual system time at output TM_DIFF. Input TM of this block can be interconnected to output TM of a TIME_BEG block in order to determine the time between calls to those two blocks.

I/Os

	Name	Data type	Description	Default setting
Input	TM	TIME	Input time in ms	T#0ms
Output	TM_DIFF	TIME	Time difference	T#0ms

Control blocks

17

17.1 CONTROL

CFC blocks of the "CONTROL" family

This family contains the following blocks:

CONT_C: Continuous-action controller (Page 164)	Continuous control (closed-loop)
CONT_S: Step controller (Page 170)	Step control
PULSEGEN: Pulse duration modulation for PID controller (Page 175)	Pulse generator

17.2 CONT_C

17.2.1 CONT_C: Continuous-action controller

Object name (type + number)

FB 1

Introduction

The CONT_C function block is used on SIMATIC S7 programmable logic controllers to control technical processes with continuous input and output variables. By means of parameter assignment you can activate or deactivate subfunctions of the PID controller to adapt the controller to the controlled system.

Application

You can use the controller as a PID fixed-setpoint controller either alone or in multi-loop feedback controls as a cascade, blending or ratio controller. The functions of the controller are based on the PID control algorithm of the sampling controller with an analog output signal, if necessary extended by including a pulse generator stage to generate pulse-duration modulated output signals for two-step or three-step controllers with proportional final controlling elements.

Description

Apart from the functions in the setpoint and process variable branches, the function block implements a complete PID controller with continuous manipulated value output and the option of influencing the manipulated value manually. The following subfunctions exist:

- Setpoint branch (Page 213)
- Process variable branch (Page 207)
- Error signal (Page 212)
- PID algorithm (Page 211)
- Manual-value processing (Page 205)
- Manipulated-value processing (Page 214)
- Disturbance variable input (Page 215)

Complete restart/Restart operating states

The CONT_C function block has a complete-restart routine.

During startup, the integrator is set internally to the initialization value I_ITVAL. When it is called in a watchdog interrupt priority class, it continues to work starting at this value.

All other outputs are set to their default values.

Error information

The error-message word RET_VAL is not used.

Input parameters

Parameters	Data type	Value range	Default setting	Description
MAN_ON	BOOL		TRUE	MANUAL VALUE ON If the "manual value on" input is set, the control loop is interrupted. A manual value is set as the manipulated value.
PVPER_ON	BOOL		FALSE	PROCESS VARIABLE PERIPHERY ON If the process variable is to be read in from the I/Os, the PV_PER input must be interconnected to the I/Os and the "process variable peripheral on" input must be set.
P_SEL	BOOL		TRUE	PROPORTIONAL ACTION ON The PID components can be activated or deactivated individually in the PID algorithm. The P component is on when the "proportional action on" input is set.
I_SEL	BOOL		TRUE	INTEGRAL ACTION ON The PID components can be activated or deactivated individually in the PID algorithm. The I component is on when the "integral action on" input is set.
INT_HOLD	BOOL		FALSE	INTEGRAL ACTION HOLD The output of the integrator can be frozen. by setting the "integral action hold" input.
I_ITL_ON	BOOL		FALSE	INITIALIZATION OF THE INTEGRAL ACTION The output of the integrator can be set to the I_ITL_VAL input. The "initialization of the integral action" input must be set for this purpose.
D_SEL	BOOL		FALSE	DERIVATIVE ACTION ON The PID components can be activated or deactivated individually in the PID algorithm. The D component is on when the "derivative action on" input is set.
SAMPLE_T	REAL	≥ 0.001 s	T#1s	SAMPLE TIME The time between block calls must be constant. The "sample time" input specifies the time between block calls.
SP_INT	REAL	-100.0 ... +100.0% or phys. value	0.0	INTERNAL SETPOINT The "internal setpoint" input is used to specify a setpoint.

Parameters	Data type	Value range	Default setting	Description
PV_IN	REAL	-100.0 ... +100.0% or phys. value	0.0	PROCESS VARIABLE IN An initialization value can be set at the "process variable in" input or an external process variable in floating point format can be connected.
PV_PER	WORD		W#16#0000	PROCESS VARIABLE PERIPHERAL The process variable peripheral is interconnected to the controller at the "process variable peripheral" input.
MAN	REAL	-100.0 ... +100.0% or phys. value	0.0	MANUAL VALUE The "manual value" input is used to set a manual value using the operator control and monitoring function.
GAIN	REAL		2.0	PROPORTIONAL GAIN The "proportional gain" input sets the controller gain.
TN	TIME	≥ SAMPLE_T	T#20s	RESET TIME The "reset time" input determines the time response of the integrator.
TV	TIME	≥ SAMPLE_T	T#10s	DERIVATIVE TIME The "derivative time" input determines the time response of the derivative unit.
TM_LAG	TIME	≥ SAMPLE_T/2	T#2s	TIME LAG OF THE DERIVATIVE ACTION The algorithm of the D component includes a time lag that can be assigned at the "time lag of the derivative action" input.
DEADB_W	REAL	≥ 0.0% or phys. value	0.0	DEAD BAND WIDTH A dead band is applied to the error signal. The "dead band width" input determines the size of the dead band.
LMN_HLM	REAL	LMN_LLM ... +100.0% or phys. value	100.0	MANIPULATED VALUE HIGH LIMIT The manipulated value is always limited to an upper and lower limit. The "manipulated value high limit" input specifies the upper limit.
LMN_LLM	REAL	-100.0 ... LMN_HLM % or phys. value	0.0	MANIPULATED VALUE LOW LIMIT The manipulated value is always limited to an upper and lower limit. The "manipulated value low limit" input specifies the lower limit.
PV_FAC	REAL		1.0	PROCESS VARIABLE FACTOR The "process variable factor" input is multiplied by the process variable. The input is used to adapt the process variable range.
PV_OFF	REAL		0.0	PROCESS VARIABLE OFFSET The "process variable offset" input is added to the process variable. The input is used to adapt the process variable range.

Parameters	Data type	Value range	Default setting	Description
LMN_FAC	REAL		1.0	MANIPULATED VALUE FACTOR The "manipulated value factor" input is multiplied by the manipulated value. The input is used to adapt the manipulated value range.
LMN_OFF	REAL		0.0	MANIPULATED VALUE OFFSET The "manipulated value offset" is added to the manipulated value. The input is used to adapt the manipulated value range.
I_ITLVAL	REAL	-100.0 ... +100.0% or phys. value	0.0	INITIALIZATION VALUE OF THE INTEGRAL ACTION The output of the integrator can be set at the I_ITL_ON input. The initialization value is applied to the "initialization value of the integral action" input.
DISV	REAL	-100.0 ... +100.0% or phys. value	0.0	DISTURBANCE VARIABLE For feedforward control, the disturbance variable is connected to the "disturbance variable" input.

Output parameters

Parameters	Data type	Value range	Default setting	Description
LMN	REAL		0.0	MANIPULATED VALUE The effective manipulated value is output in floating point format at the "manipulated value" output.
LMN_PER	WORD		W#16#0000	MANIPULATED VALUE PERIPHERY The manipulated value periphery is connected to the controller at the "manipulated value periphery" output.
QLMN_HLM	BOOL		FALSE	HIGH LIMIT OF MANIPULATED VALUE REACHED The manipulated value is always limited to an upper and lower limit. The "high limit of manipulated value reached" output indicates that the upper limit has been exceeded.
QLMN_LLM	BOOL		FALSE	LOW LIMIT OF MANIPULATED VALUE REACHED The manipulated value is always limited to an upper and lower limit. The "low limit of manipulated value reached" output indicates that the lower limit has been undershot.

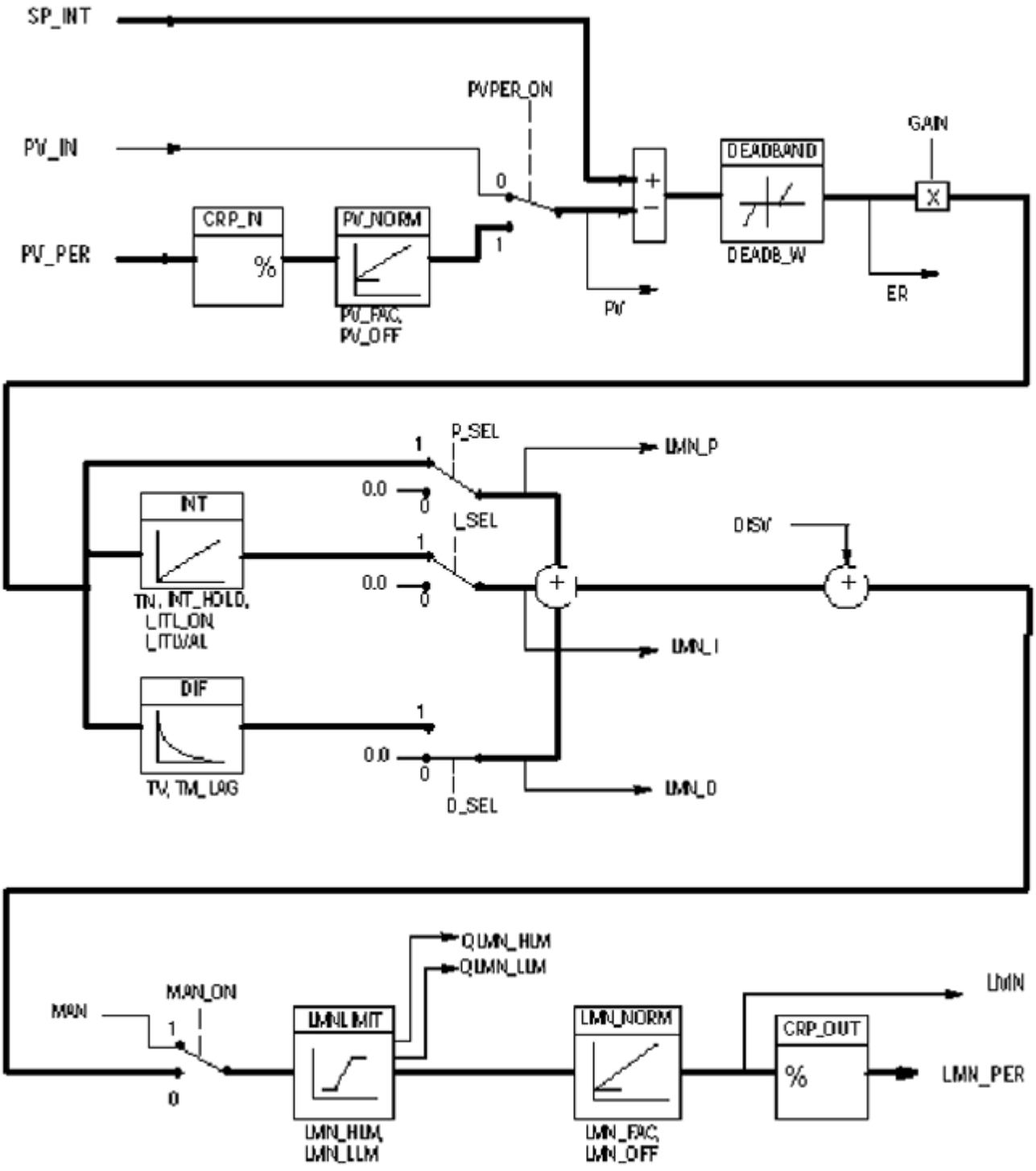
Parameters	Data type	Value range	Default setting	Description
LMN_P	REAL		0.0	PROPORTIONALITY COMPONENT The "proportional component" output contains the proportional component of the manipulated value.
LMN_I	REAL		0.0	INTEGRAL COMPONENT The "integral component" output contains the integral component of the manipulated value.
LMN_D	REAL		0.0	DERIVATIVE COMPONENT The "derivative component" output contains the derivative component of the manipulated value.
PV	REAL		0.0	PROCESS VARIABLE The effective process variable is output at the "process variable" output.
ER	REAL		0.0	ERROR SIGNAL The effective error signal is output at the "error signal" output.

Additional information

For more information, refer to the section:

CONT_C: Block diagram (Page 169)

17.2.2 CONT_C: Block diagram



17.3 CONT_S

17.3.1 CONT_S: Step controller

Object name (type + number)

FB 2

Introduction

The CONT_S function block is used on SIMATIC S7 programmable logic controllers to control technical processes with binary manipulated-value output signals for integrating final controlling elements. By means of parameter assignment you can activate or deactivate subfunctions of the PI step controller to adapt the controller to the process.

Application

You can use the controller as a PI fixed-setpoint controller or in secondary control loops in cascade, blending or ratio controllers, but not as the master controller. The functions of the controller are based on the PI control algorithm of the sampling controller supplemented by the functions for generating the binary output signal from the analog actuating signal.

The I component of the controller can be closed with TN = T#0ms. In this way the block can be used as a P controller.

So that the controller can work without repeated manipulated value, the manipulated value calculated internally does not match the final controlling element position. A comparison is made if the manipulated value (ER * GAIN) is negative. The controller then sets the output manipulated-value signal down (QLMNDN) until the low limit signal of the repeated manipulated value (LMNR_LS) is set.

The controller can also be used in a controller cascade as a subordinate position controller. The final controlling element position is set with the setpoint input SP_INT. In this case the process-variable input and the integration time parameter (TN) have to be set to zero. A possible application would be a temperature controller with heating-capacity control via interpulse period control and cooling-capacity control via a butterfly valve.

In order to close the valve completely, the manipulated value (ER * GAIN) should be negative.

Description

In addition to the functions in the process-variable branch, the function block implements a complete PI controller with a binary manipulated-value output and the option of influencing the manipulated value manually. The step controller operates without repeated manipulated value. The following subfunctions exist:

- Setpoint branch (Page 213)
- Process variable branch (Page 207)
- Error signal (Page 212)

- PI step algorithm (Page 210)
- Disturbance variable input (Page 215)

Complete restart/Restart operating states

The CONT_S function block has a complete-restart routine.

All outputs are set to their default values.

Error information

The error-message word RET_VAL is not used.

Input parameters

Parameters	Data type	Value range	Default setting	Description
LMNR_HS	BOOL		FALSE	HIGH LIMIT SIGNAL OF REPEATED MANIPULATED VALUE The "control valve at upper limit stop" signal is connected to the "high limit signal of repeated manipulated value" input. LMNR_HS = TRUE means: That the control valve is at the upper limit stop.
LMNR_LS	BOOL		FALSE	LOW LIMIT SIGNAL OF REPEATED MANIPULATED VALUE The "control valve at lower limit stop" signal is connected to the "low limit signal of repeated manipulated value" input. LMNR_LS = TRUE means: that the control valve is at the lower limit stop.
LMNS_ON	BOOL		TRUE	MANIPULATED SIGNALS ON Manipulated-value signal processing is switched to manual at the "manipulated signals on" input.
LMNUP	BOOL		FALSE	MANIPULATED SIGNALS UP When manipulated-value signal processing is switched to manual, the output signal QLMNUP is set at the "manipulated signals up" input.
LMNDN	BOOL		FALSE	MANIPULATED SIGNALS DOWN When manipulated-value signal processing is switched to manual, the output signal QLMNDN is set at the "manipulated signals down" input.
PVPER_ON	BOOL		FALSE	PROCESS VARIABLE PERIPHERY ON If the process variable is to be read in from the I/Os, the PV_VER input must be interconnected to the I/Os and the "process variable periphery on" input must be set.

Parameters	Data type	Value range	Default setting	Description
SAMPLE_T	REAL	≥ 0.001s	T#1s	SAMPLE TIME The time between block calls must be constant. The "sample time" input specifies the time between block calls.
SP_INT	REAL	-100.0 ... +100.0% or phys. value	0.0	INTERNAL SETPOINT The "internal setpoint" input is used to specify a setpoint.
PV_IN	REAL	-100.0 ... +100.0% or phys. value	0.0	PROCESS VARIABLE IN An initialization value can be set at the "process variable in" input or an external process variable in floating point format can be connected.
PV_PER	WORD		W#16#0000	PROCESS VARIABLE PERIPHERAL The process variable peripheral is interconnected to the controller at the "process variable peripheral" input.
GAIN	REAL		2.0	PROPORTIONAL GAIN The "proportional gain" input sets the controller gain.
TN	TIME	≥ SAMPLE_T	T#20s	RESET TIME The "reset time" input determines the time response of the integrator.
DEADB_W	REAL	≥ 0.0% or phys. value	0.0	DEAD BAND WIDTH A dead band is applied to the error signal. The "dead band width" input determines the size of the dead band.
PV_FAC	REAL		1.0	PROCESS VARIABLE FACTOR The "process variable factor" input is multiplied by the process variable. The input is used to adapt the process variable range.
PV_OFF	REAL		0.0	PROCESS VARIABLE OFFSET The "process variable offset" input is added to the process variable. The input is used to adapt the process variable range.
PULSE_TM	TIME	≥ SAMPLE_T	T#3s	MINIMUM PULSE TIME A minimum pulse width can be set at the "minimum pulse time" parameter.
BREAK_TM	TIME	≥ SAMPLE_T	T#3s	MINIMUM BREAK TIME A minimum break duration can be set at the "minimum break time" parameter.

Parameters	Data type	Value range	Default setting	Description
MTR_TM	TIME	≥ SAMPLE_T	T#30s	MOTOR MANIPULATED VALUE The time required by the control valve to move from limit stop to limit stop is entered at the "motor manipulated value" parameter.
DISV	REAL	-100.0 ... +100.0% or phys. value	0.0	DISTURBANCE VARIABLE For feedforward control, the disturbance variable is connected to the "disturbance variable" input.

Output parameters

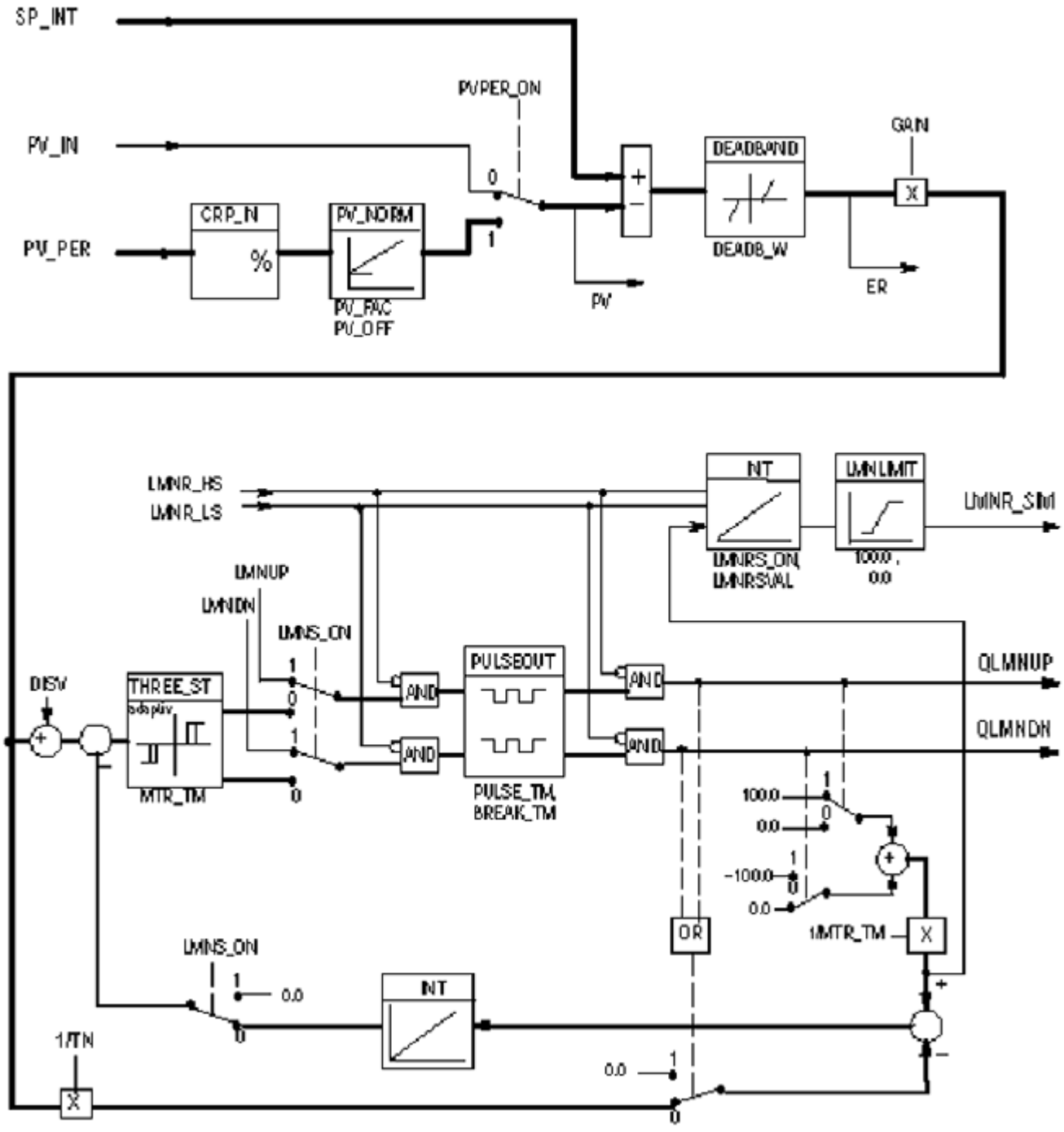
Parameters	Data type	Value range	Default setting	Description
QLMNUP	BOOL		FALSE	MANIPULATED SIGNAL UP If the "manipulated signal up" output is set, the control valve should be opened.
QLMNDN	BOOL		FALSE	MANIPULATED SIGNAL DOWN If the "manipulated signal down" output is set, the control valve should be closed.
PV	REAL		0.0	PROCESS VARIABLE The effective process variable is output at the "process variable" output.
ER	REAL		0.0	ERROR SIGNAL The effective error signal is output at the "error signal" output.

Additional information

For more information, refer to the section:

CONT_S: Block diagram (Page 174)

17.3.2 CONT_S: Block diagram



17.4 PULSEGEN

17.4.1 PULSEGEN: Pulse duration modulation for PID controller

Object name (type + number)

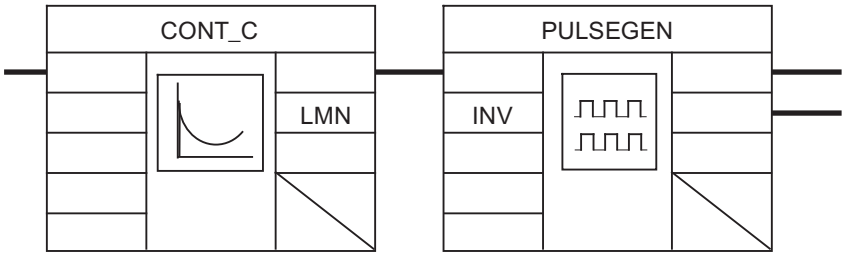
FB 3

Introduction

The PULSEGEN function block is used to structure a PID controller with pulse output for proportional final controlling elements.

Application

PID two-step or three-step controllers with pulse-duration modulation can be configured with the PULSEGEN function block. The function is normally used in conjunction with the continuous controller CONT_C (Page 164).



Description

The PULSEGEN function transforms the input variable INV (=LMN of the PID controller) by modulating the pulse duration in a pulse train with constant period time. It corresponds to the cycle time used to update the input variable and must be parameterized in PER_TM.

The duration of a pulse per period time is proportional to the input variable. The cycle assigned to PER_TM is not identical to the processing cycle of the PULSEGEN function block. Rather, a PER_TM cycle consists of several PULSEGEN function block processing cycles. In this context, the number of PULSEGEN calls per PER_TM cycle provides a measure of the accuracy of the pulse-duration modulation.

You will find more information on pulse duration modulation under: Pulse-duration modulation (Page 206)

An input variable of 30% and 10 PULSEGEN calls per PER_TM means:

- "One" at the QPOS output for the first three calls of PULSEGEN (30% of 10 calls)
- "Zero" at the QPOS output for seven further PULSEGEN calls (70% of 10 calls)

The pulse width is recalculated at the beginning of each period.

Accuracy of manipulated value

At a "sample ratio" of 1:10 (CONT_C calls to PULSEGEN calls) the accuracy of the manipulated value is restricted to 10% in this example. Default input values INV can only be mapped to a pulse width at the QPOS output with a tolerance of 10%.

Accuracy will, therefore, increase accordingly as the number of PULSEGEN calls per CONT_C call increases.

If PULSEGEN is called, for example, 100 times more often than CONT_C, a resolution of 1% of the manipulated value range is achieved.

Note

The call frequency must be programmed by the user.

Automatic synchronization

It is possible to synchronize the pulse output automatically with the block that updates the input variable INV (for example, CONT_C). This ensures that a changing input variable is output as quickly as possible as a pulse.

The pulse generator always evaluates the input variable INV at intervals corresponding to the period time PER_TM and converts the value into a pulse signal of corresponding length. Since, however, INV is usually calculated in a slower cyclic interrupt, the pulse generator should start the conversion of the discrete value into a pulse signal as soon as possible after the updating of INV.

To allow this, the block can synchronize the start of the period itself using the following procedure:

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

Automatic synchronization can be disabled by setting the "SYN_ON" input to FALSE.

Note

When a new period starts, the old value of INV (in other words of LMN) will be simulated in the pulse signal more or less inaccurately following synchronization.

Operating modes

Depending on the parameters assigned to the pulse generator, PID controllers with a three-step output or with a bipolar or unipolar two-step output can be configured. The following table illustrates the setting of the switch combinations for the possible modes.

Operating mode	Switch		
MAN_ON	STEP3_ON	ST2BI_ON	
Three-step control	FALSE	TRUE	Any

Operating mode	Switch		
Two-step control with bipolar control range (-100% ... +100%)	FALSE	FALSE	TRUE
Two-step control with monopolar control range (0% ... +100%)	FALSE	FALSE	FALSE
Manual operation	TRUE	Any	Any

Input parameters

Parameters	Data type	Value range	Default setting	Description
INV	REAL	-100.0 ... +100.0%	0.0	INPUT VARIABLE An analog manipulated value is connected to the "input variable" input parameter.
PER_TM	TIME	≥ 20 * SAMPLE_T	T#1s	PERIOD TIME The constant period time of the pulse width modulation is specified at the "Period time" input parameter. This corresponds to the sample time of the controller. The ratio between the sample time of the pulse generator and the sample time of the controller determines the accuracy of the pulse-duration modulation.
P_B_TM	TIME	≥ SAMPLE_T	T#0ms	MINIMUM PULSE/BREAK TIME A minimum pulse or minimum break time can be assigned at the "minimum pulse/break time" input parameter.
RATIO-FAC	REAL	0.1 ... 10.0	1.0	RATIO FACTOR The "ratio factor" input parameter can be used to change the ratio of the duration of negative to positive pulses. In a thermal process, this would, for example, allow different time constants for heating and cooling to be compensated (for example, in a process with electrical heating and water cooling).
STEP3_ON	BOOL		TRUE	THREE STEP CONTROL ON The "three step control on" input parameter activates this mode. In three-step control, both output signals are active.
ST2BI_ON	BOOL		FALSE	TWO STEP CONTROL FOR BIPOLAR MANIPULATED VALUE RANGE ON The "two step control for bipolar manipulated value range on" input parameter can be used to select between the "two-step control for bipolar manipulated value" and "two-step control for unipolar manipulated value range" modes. The STEP3_ON parameter must be set to FALSE.

Parameters	Data type	Value range	Default setting	Description
MAN_ON	BOOL		FALSE	MANUAL MODE ON By setting the input parameter "manual mode on", the output signals can be set manually.
POS_P_ON	BOOL		FALSE	POSITIVE PULSE ON In manual operation with three-step control, the QPOS_P output signal can be set at the "positive pulse on" input parameter. In manual operation with two-step control, QNEG_P is always set inversely to QPOS_P.
NEG_P_ON	BOOL		FALSE	NEGATIVE PULSE ON In manual operation with three-step control, the QNEG_P output signal can be set at the "negative pulse on" input parameter. In manual operation with two-step control, QNEG_P is always set inversely to QPOS_P.
SYN_ON	BOOL		TRUE	SYNCHRONISATION ON The "synchronization on" input parameter can be set in order to synchronize the pulse output automatically with the block that updates the INV input variable. This ensures that a changing input variable is output as quickly as possible as a pulse.
SAMPLE_T	REAL	≥ 0.001s	1	SAMPLE TIME [s] The time between block calls must be constant. The "sample time" input specifies the time between block calls.

Note

The values of the input parameters are not limited in the block. There is no parameter check.

Output parameters

Parameters	Data type	Value range	Default setting	Description
QPOS_P	BOOL		FALSE	OUTPUT POSITIVE PULSE The "output positive pulse" output parameter is set when a pulse is to be output. In three-step control, this is always the positive pulse. In two-step control, QNEG_P is always set inversely to QPOS_P.
QNEG_P	BOOL		FALSE	OUTPUT NEGATIVE PULSE The "output negative pulse" output parameter is set when a pulse is to be output. In three-step control, this is the negative pulse. In two-step control, QNEG_P is always set inversely to QPOS_P.

Complete restart/Restart operating states

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

Error information

The error-message word RET_VAL is not used.

Additional information

You can find additional information in the following sections:

PULSEGEN: Block diagram (Page 180)

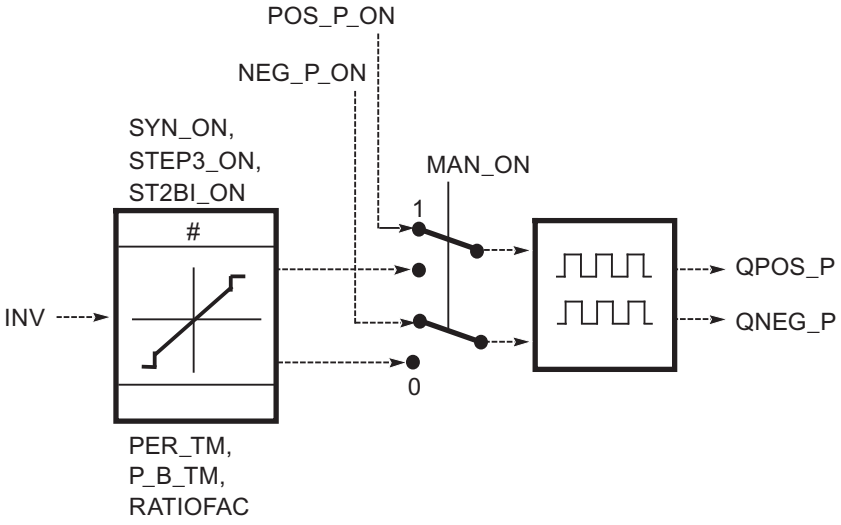
PULSEGEN: Three-step control (Page 180)

PULSEGEN: Three-step control, asymmetrical (Page 181)

PULSEGEN: Two-step control (Page 182)

PULSEGEN: Manual operation in two- or three-step control (Page 182)

17.4.2 PULSEGEN: Block diagram



17.4.3 PULSEGEN: Three-step control

Description

In "three-step control" mode, the actuating signal can adopt three states. The values of the binary output signals QPOS_P and QNEG_P are assigned to the relevant statuses of the final controlling element. The table shows the example of a temperature control:

	Final controlling element		
Output signal	**Heat**	**Off**	**Cool**
QPOS_P	TRUE	FALSE	FALSE
QNEG_P	FALSE	FALSE	TRUE

Based on the input variable, a characteristic curve is used to calculate a pulse width. The form of the characteristic curve is defined by the minimum pulse or minimum break time and the ratio factor. The normal value for the ratio factor is 1.

The "doglegs" in the curves are caused by the minimum pulse or minimum break times. You will find more information on characteristic curves under:

Symmetrical characteristic for three-step controller (Page 216)

Minimum pulse or minimum break time

A correctly assigned minimum pulse or minimum break time P_B_TM can prevent short on/off times that reduce the working life of switching elements and actuators.

Note

Small absolute values at the input variable LMN that would generate a pulse width shorter than P_B_TM are suppressed. Large input values that would generate a pulse width longer than (PER_TM - P_B_TM) are set to 100% or -100%.

The duration of the positive or negative pulses is calculated from the input variable (in %) multiplied by the period time. Pulse width = INV/100 * PER_TM

Additional information

You can find additional information in the following sections:

PULSEGEN: Three-step control, asymmetrical (Page 181)

PULSEGEN: Two-step control (Page 182)

PULSEGEN: Manual operation in two- or three-step control (Page 182)

17.4.4 PULSEGEN: Three-step control, asymmetrical

Description

The ratio factor RATIOFAC can be used to change the ratio of the duration of positive to negative pulses. In a thermal process, for example, this would allow different system time constants for heating and cooling.

The ratio factor also affects the minimum pulse time or period time. A ratio factor < 1 means that the threshold value for negative pulses is multiplied by the ratio factor.

Ratio factor < 1

The pulse width at the negative pulse output calculated from the input variable multiplied by the period time is reduced by the ratio factor:

Positive pulse width = INV/100 * PER_TM

Negative pulse width = INV/100 * PER_TM + RATIOFAC

Ratio factor > 1

The pulse width at the positive pulse output calculated from the input variable multiplied by the period time is reduced by the ratio factor:

Positive pulse width = INV/100 + PER_TM

Negative pulse width = INV/100 * PER_TM/RATIOFAC

Additional information

You can find additional information in the following sections:

Asymmetrical characteristic for three-step controller (Page 217)

PULSEGEN: Three-step control (Page 180)

PULSEGEN: Two-step control (Page 182)

PULSEGEN: Manual operation in two- or three-step control (Page 182)

17.4.5 PULSEGEN: Two-step control

Description

In two-step control, only the positive pulse output QPOS_P of PULSEGEN is connected to the on/off final controlling element. Depending on the manipulated-value range being used, the two-step controller will have a bipolar or a unipolar manipulated-value range.

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

Pulse	Final cont. el. on	Final cont. el. off
QPOS_P	TRUE	FALSE
QNEG_P	FALSE	TRUE

Additional information

You can find additional information in the following sections:

Characteristic with bipolar range of manipulated values (Page 208)

Characteristic with unipolar range of manipulated values (Page 209)

PULSEGEN: Three-step control (Page 180)

PULSEGEN: Three-step control, asymmetrical (Page 181)

PULSEGEN: Manual operation in two- or three-step control (Page 182)

17.4.6 PULSEGEN: Manual operation in two- or three-step control

Description

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

	POS_P_ON	NEG_P_ON	QPOS_P	QNEG_P
Three-step control	FALSE	FALSE	FALSE	FALSE
	TRUE	FALSE	TRUE	FALSE
	FALSE	TRUE	FALSE	TRUE
	TRUE	TRUE	FALSE	FALSE

	POS_P_ON	NEG_P_ON	QPOS_P	QNEG_P
Two-step control	FALSE	Any	FALSE	TRUE
	TRUE	Any	TRUE	FALSE

Additional information

You can find additional information in the following sections:

PULSEGEN: Three-step control (Page 180)

PULSEGEN: Three-step control, asymmetrical (Page 181)

PULSEGEN: Two-step control (Page 182)

System function blocks (SFBs) 18

18.1 EVENT: Start of the priority class

CFC blocks of the "SYSTEM" family

This family contains the following system calls provided by the M7-300/400 run-time system:

DELAY (Page 186)	Generation of a software interrupt whose name is forwarded as a parameter
DELAY: Delaying the start event (Page 187)	Delay for all start events until processing is enabled
EDELAY: Enable signal for delayed start events (Page 188)	Enable signal for delayed start events
DISCARD: All arising start events are discarded (Page 189)	Discards all (not yet started) start events, in order to run the called task without interruption
EDISCARD: Enable signal for all new start events (Page 190)	Enable signal for all new start events
LASTERR: Determining the error code in DELAY, EDELAY, DISCARD, EDISCARD,P_REASON (Page 191)	Determines the error code of the I/O error and errors in the SFBs DELAY, EDELAY, DISCARD, EDISCARD, P_REASON
SYSTIME: Determines the system time (Page 192)	Determines the system time
P_REASON: Determining the cause of the process interrupt (Page 193)	Determines the cause of process interrupts

18.2 DELAY

Note

The block can only be used for M7-300/400 run-time systems!

Function

The block generates a software interrupt. It starts the task whose name is set at the TN input.

If a name is specified to which no task has been assigned, an error will be reported during compilation and consistency checking.

I/Os

	Name	Data type	Description	Default setting
Input	TN	TASK	Task name	0

18.3 DELAY: Delaying the start event

Note

The block can only be used for M7-300/400 run-time systems!

Function

This block enables the operation of a calling task without interruptions by other tasks. All queued start events are delayed either until processing is enabled (via the EDELAY block) or until the task currently being executed has been completed.

All queued start events are then executed.

Any errors that occur during task processing can be queried via the LASTERR block.

Information on the EDELAY and LASTERR blocks is contained in the following sections:

- EDELAY: Enable signal for delayed start events (Page 188)
- LASTERR: Determining the error code in DELAY, EDELAY, DISCARD, EDISCARD,P_REASON (Page 191)

18.4 EDELAY: Enable signal for delayed start events

Note

The block can only be used for M7-300/400 run-time systems!

Function

This block enables delayed start events. These events must have been delayed initially via the DELAY block.

Any errors that occur during task processing can be queried via the LASTERR block.

Information on the DELAY and LASTERR blocks is contained in the following sections:

- DELAY: Delaying the start event (Page 187)
- LASTERR: Determining the error code in DELAY, EDELAY, DISCARD, EDISCARD,P_REASON (Page 191)

18.5 DISCARD: All arising start events are discarded

Note

The block can only be used for M7-300/400 run-time systems!

Function

This block enables the operation of a calling task without interruptions by other tasks. All queued start events are discarded, i.e., the tasks are not started. Events already registered (delayed) are processed. Process interrupts are acknowledged immediately.

All queued start events are discarded either until processing is enabled (via the EDISCARD block) or until the task currently being executed has been completed.

Any errors that occur during task processing can be queried via the LASTERR block.

Information on the EDISCARD and LASTERR blocks is contained in the following sections:

- EDISCARD: Enable signal for all new start events (Page 190)
- LASTERR: Determining the error code in DELAY, EDELAY, DISCARD, EDISCARD,P_REASON (Page 191)

18.6 EDISCARD: Enable signal for all new start events

Note

The block can only be used for M7-300/400 run-time systems!

Function

This block enables queued start events. These events must have been discarded initially via the DISCARD block.

Any errors that occur during task processing can be queried via the LASTERR block.

Information on the DISCARD and LASTERR blocks is contained in the following sections:

- DISCARD: All arising start events are discarded (Page 189)
- LASTERR: Determining the error code in DELAY, EDELAY, DISCARD, EDISCARD,P_REASON (Page 191)

18.7 LASTERR: Determining the error code in DELAY, EDELAY, DISCARD, EDISCARD,P_REASON

Note

The block can only be used for M7-300/400 run-time systems!

Function

This block returns the error code of the last error that occurred for the following error classes:

- I/O errors
- Errors in the system blocks:
  - DELAY: Delaying the start event (Page 187),
  - EDELAY: Enable signal for delayed start events (Page 188),
  - DISCARD: All arising start events are discarded (Page 189),
  - EDISCARD: Enable signal for all new start events (Page 190),
  - P_REASON: Determining the cause of the process interrupt (Page 193)

The possible values represent a subset of M7 system software error codes. For additional information, refer to file M7API.H or to the M7 system software documentation.

I/Os

	Name	Data type	Description	Default setting
Output	ERR	DINT	Error code	0

18.8 SYSTIME: Determines the system time

Note

The block can only be used for M7-300/400 run-time systems!

Function

This block can be used to determine the system time and display it in TIME format at its output.

I/Os

	Name	Data type	Description	Default setting
Output	TIME	TIME	System time	0

18.9 P_REASON: Determining the cause of the process interrupt

Function

This block can be used to locate the cause of a process interrupt.

The name of the task required is specified at input TN. The block will not have any effect if this task is not a process interrupt.

Additional information the process alarm had output during its last call is written to the STATE output. This additional information refers to specific modules and its byte sequence is, therefore, returned in INTEL format.

The MASK output also returns the interrupt mask configured in CFC for this process-interrupt task.

Any errors that occur during task processing can be queried via the LASTERR block.

You can find additional information on the LASTERR block in the section: LASTERR: Determining the error code in DELAY, EDELAY, DISCARD, EDISCARD,P_REASON (Page 191)

I/Os

	Name	Data type	Description	Default setting
Input	TN	TASK	Task name	0
Outputs	STATE	STATE	Alarm state	0
	MASK	DWORD	Alarm mask	0

18.10 FRC_CFC: Internal Block

Object name (type + number)

FB 136

This block is a system block and is only used internally. There is therefore no help available for it.

Blocks for AS-wide connections 19

19.1 IK_STATE

IK_STATE: Display status of an AS-wide connection

Object name (type + number)

FC 157

Application

The block is used in a CFC chart when AS-wide connections are used and when the error status of these connections is to be evaluated in the user program.

Insert one instance of the block in the CFC chart on the sending and receiving end for each AS-wide connection.

Remove the instances when you have deleted the AS-wide connection.

Function

The block will output the error status of the AS-wide connection whose ID was configured at the NETPRO_ID input.

The block outputs a separate bit for each error. A group error is also available. An error will also be issued if the specified NETPRO_ID does not exist.

Troubleshooting

The return value RET_VAL is not used.

Input parameters

Parameter	Data type	Default setting	Description
NETPRO_ID	INT	0	NETPRO_ID of the connection
MasterDB	INT	0	DB number of the Master DB. Automatically entered by the CFC

Output parameters

Parameter	Data type	Default setting	Description
SendErr	BOOL	FALSE	Send error Display in BSEND
RcvErr	BOOL	FALSE	Receive error Display in BRCV
SendOvl	BOOL	FALSE	Sender overload This error can be created when processing of the intermediate send buffers is too slow.
RcvOv	BOOL	FALSE	Receiver overload This error can be created when processing of the intermediate receive buffers is too slow.
RcvChg	BOOL	FALSE	Inconsistent change at receiver 1. The AS-wide connection has been reconfigured at the receiver end, which also has an effect on the sender. This could not be reached at that time. 2. The AS-wide connection has been reconfigured at the sender end, which also has an effect on the receiver. This could not be reached at that time. 3. Sender and receiver have different AS-wide data structures due to different charging states.
RcvTmout	BOOL	FALSE	Receiver timeout Possible reasons: 1. STOP the sending AS 2. Sender has not loaded BSEND (yet)
SysErr	BOOL	FALSE	System error AS-wide data block does not exist, for example
GroupErr	BOOL	FALSE	Group error of the 8 errors above
ConnNA	BOOL	FALSE	Configured NETPRO_ID does not exist

Additional information on AS-wide connection is available in the help "CFC for SIMATIC S7 > Creating runtime structures > Creating and working with interconnections > Creating AS-wide interconnections".

19.2 IK_MANAG

IK_MANAG

Object name (type+number)

FC152

Application

The "IK_MANAG block is part of the runtime system to support the AS-wide interconnections; it is copied during initial creation of an AS-wide interconnection to the block folder of the S7 program.

Therefore this block will not be described in detail.

Additional information on AS-wide connection is available in the help "CFC for SIMATIC S7 > Creating runtime structures > Creating and working with interconnections > Creating AS-wide interconnections".

19.3 IK_SEND

IK_SEND

Object name (type+number)

FC155

Application

The "IK_SEND block is part of the runtime system to support the AS-wide interconnections; it is copied during initial creation of an AS-wide interconnection to the block folder of the S7 program.

Therefore this block will not be described in detail.

Additional information on AS-wide connection is available in the help "CFC for SIMATIC S7 > Creating runtime structures > Creating and working with interconnections > Creating AS-wide interconnections".

19.4 IK_RCV

IK_RCV

Object name (type+number)

FC156

Application

The "IK_RCV block is part of the runtime system to support the AS-wide interconnections; it is copied during initial creation of an AS-wide interconnection to the block folder of the S7 program.

Therefore this block will not be described in detail.

Additional information on AS-wide connection is available in the help "CFC for SIMATIC S7 > Creating runtime structures > Creating and working with interconnections > Creating AS-wide interconnections".

19.5 IK_CP_OU

IK_CP_OU

Object name (type+number)

FC154

Application

The "IK_CP_OU block is part of the runtime system to support the AS-wide interconnections; it is copied during initial creation of an AS-wide interconnection to the block folder of the S7 program.

Therefore this block will not be described in detail.

Additional information on AS-wide connection is available in the help "CFC for SIMATIC S7 > Creating runtime structures > Creating and working with interconnections > Creating AS-wide interconnections".

19.6 IK_CP_IN

IK_CP_IN

Object name (type+number)

FC153

Application

The "IK_CP_IN block is part of the runtime system to support the AS-wide interconnections; it is copied during initial creation of an AS-wide interconnection to the block folder of the S7 program.

Therefore this block will not be described in detail.

Additional information on AS-wide connection is available in the help "CFC for SIMATIC S7 > Creating runtime structures > Creating and working with interconnections > Creating AS-wide interconnections".

19.7 IK_ALARM

IK_ALARM

Object name (type+number)

FB244

Application

The "IK_ALARM block is part of the runtime system to support the AS-wide interconnections; it is copied during initial creation of an AS-wide interconnection to the block folder of the S7 program.

Therefore this block will not be described in detail.

Additional information on AS-wide connection is available in the help "CFC for SIMATIC S7 > Creating runtime structures > Creating and working with interconnections > Creating AS-wide interconnections".

"@SYSTEM" block family

20

20.1 PA_CPU: Monitoring block for license information

Function

This block compares during runtime the number of process objects (POs) that are recorded under the "AS RT PO" license with the number of POs that are loaded in the CPU 410-5H Process Automation ("CPU 410-5H PA").

The block is automatically inserted in CFC "@PA-CPU" during compilation and this CFC is installed in OB1. The block and CFC are loaded in the "CPU 410-5H PA" during the download to the AS.

If the number of recorded licenses for POs in the Automation License Manager is lower than the number of POs that are loaded in the AS, insufficient licensing is indicated by:

- An entry in the diagnostics buffer of the CPU
- Cyclic triggering of a corresponding message in SIMATIC WinCC.
  The message is always triggered immediately during the loading process for which insufficient licensing is detected and when the difference is changed by the loading process. Afterwards, the message is triggered in intervals of 6 hours.
  When the process objects are sufficiently licensed once again, for example, due to a reduction of the POs required in the AS program or the purchase of additional licenses, the message is no longer triggered. No "OUTGOING" message is generated.

Note

For the "CPU 410-5H PA", the maximum number of loadable POs are licensed by the hardware. It is not possible to download a greater number of POs than is licensed by the hardware into the AS. However, this limit is not relevant for the comparison in block "PA-CPU".

Appendix

21

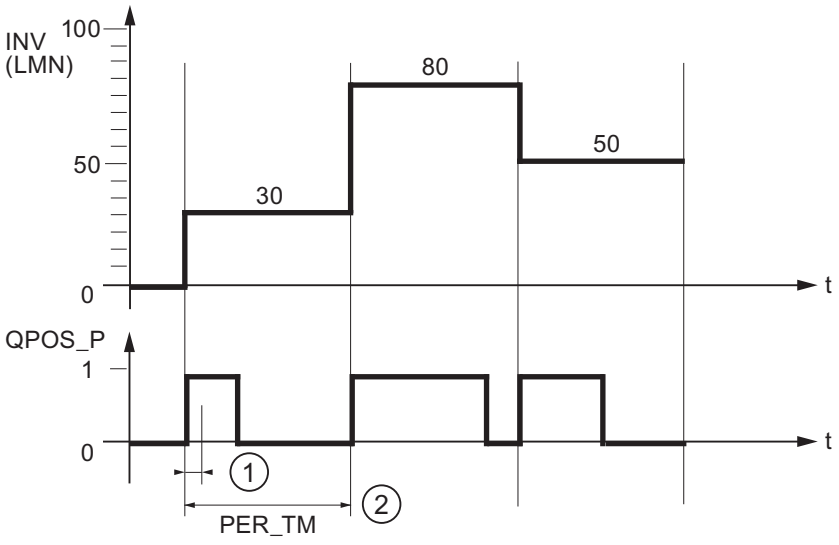
21.1 Manual-value processing

It is possible to switch between manual and automatic operation. In the case of manual operation the manipulated variable is corrected to a manual value.

The integrator (INT) is set internally to LMN - LMN_P - DISV and the differentiator (DIF) to 0, and then adjusted internally. This means that switching over to automatic operation will be smooth.

21.2 Pulse-duration modulation

Pulse-duration modulation



Key	
1	PULSEGEN cycle
2	Cycle CONT_C

21.3 Process variable branch

The process variable can be read in as a peripheral or floating-point value. The function CRP_IN converts the peripheral value PV_PER into a floating-point format from -100 to +100% in accordance with the following rule:

Output of CPR_IN = PV_PER * 100/27648

The function PV_NORM normalizes the output of CRP_IN in accordance with the following rule:

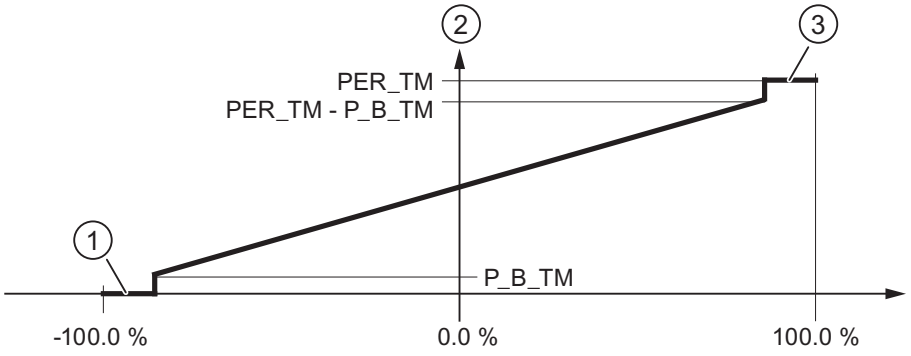
Output of PV_NORM = (output of CRP_IN) * PV_FAC + PV_OFF

The default value of PV_FAC is 1 and that of PV_OFF is 0.

21.4 Characteristic with bipolar range of manipulated values

Characteristic with bipolar range of manipulated values

Range of manipulated values -100% to 100%

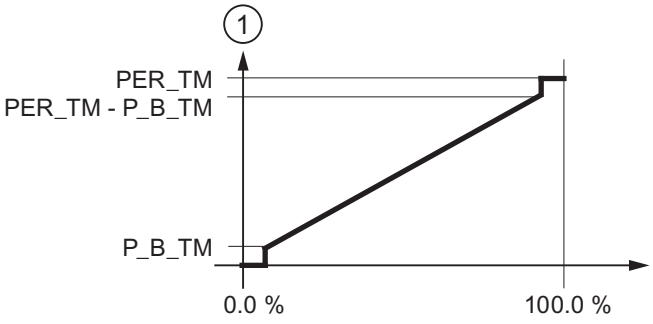


Key	
1	Continuous "Off"
2	Duration of the positive pulse
3	Continuous "On"

21.5 Characteristic with unipolar range of manipulated values

Characteristic with unipolar range of manipulated values

Range of manipulated values 0% to 100%



Key	
1	Duration of the positive pulse

21.6 PI step algorithm

This function block runs without a repeated manipulated value. The I component of the PI algorithm and the assumed repeated manipulated value are calculated in one integrator (INT) and compared as a feedback value with the remaining P component. The difference then goes to a three-step element (THREE_ST) and to a pulse generator (PULSEOUT), which generates the pulses for the control valve. The switching frequency of the controller is reduced by adapting the response threshold of the three-step element.

21.7 PID algorithm

The PID algorithm functions as a manipulated variable algorithm. The proportional, integral (INT) and differential (DIF) components are connected in parallel and can be switched in or out individually. In this way P, PI, PD and PID controllers and even pure I and D controllers can be configured,

21.8 Error signal

The difference between setpoint value and process variable constitutes the error signal. In order to suppress a minor sustained oscillation due to quantization of manipulated values (for example, in the case of pulse duration modulation with PULSEGEN or limited resolution of the manipulated value by the control valve) the error signal is routed via a dead zone (DEADBAND). When DEADB_W = 0 the dead band is switched out.

21.9 Setpoint branch

The setpoint branch is input at input SP_INT in floating-point format.

21.10 Manipulated-value processing

The manipulated value is limited to pre-definable values using the LMNLIMIT function. Signaling bits are set whenever the limits are exceeded.

The function LMN_NORM normalizes the output of LMNLIMIT in accordance with the following rule:

LMN = (output of LMNLIMIT) * LMN_FAC + LMN_OFF

The default value of LMN_FAC is 1 and that of LMN_OFF is 0.

The manipulated value is also available in periphery format. The function CRP_OUT converts the floating-point value LMN into a peripheral value in accordance with the following rule:

LMN_PER = LMN * 27648/100

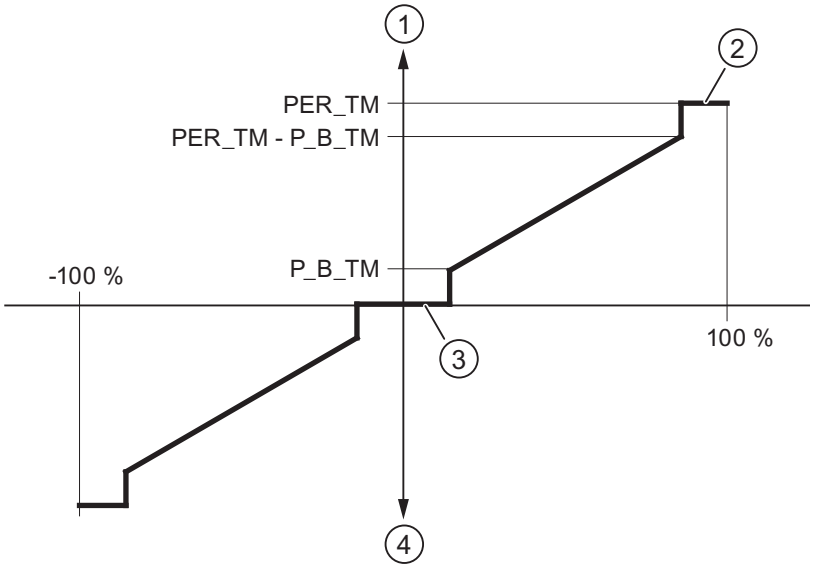
21.11 Disturbance variable input

A disturbance variable can be feedforward-controlled additively at the DISV input.

21.12 Symmetrical characteristic for three-step controller

Symmetrical characteristic for three-step controller

Ratio factor = 1

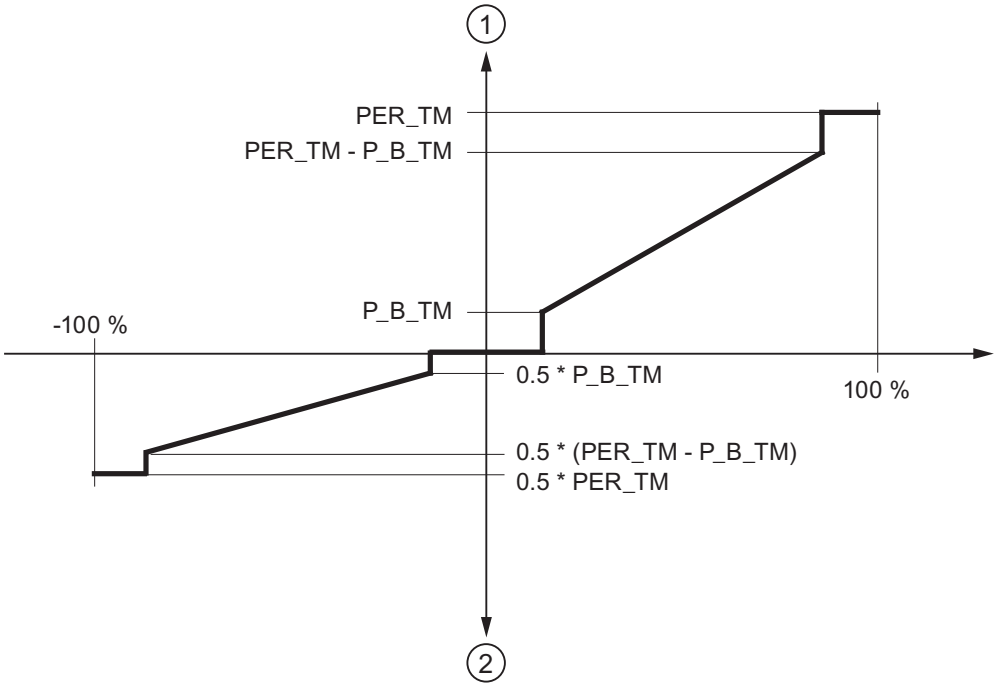


Key	
1	Duration of the positive pulse
2	Continuous "On"
3	Continuous "Off"
4	Duration of the negative pulse

21.13 Asymmetrical characteristic for three-step controller

Asymmetrical characteristic for three-step controller

Ratio factor = 0.5



Key	
1	Duration of the positive pulse
2	Duration of the negative pulse

Index

A

B

C

D

E

F

I

J

L

M

N

O

P

R

S

T

U

W

X