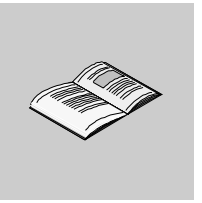
Concept 2.6
IEC block library
Part: IEC

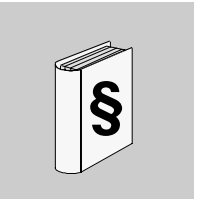
01/2007

33002217.03

Table of Contents



Safety Information



Important Information

NOTICE

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

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

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

⚠ DANGER

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

⚠ WARNING

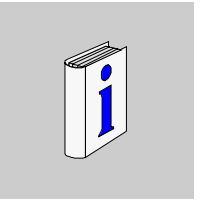
WARNING indicates a potentially hazardous situation, which, if not avoided, **can result** in death, serious injury, or equipment damage.

⚠ CAUTION

CAUTION indicates a potentially hazardous situation, which, if not avoided, **can result** in injury or equipment damage.

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

About the Book



At a Glance

Document Scope

This documentation is designed to help with the configuration of functions and function blocks.

Validity Note

This documentation applies to Concept 2.6 under Microsoft Windows 98, Microsoft Windows 2000, Microsoft Windows XP and Microsoft Windows NT 4.x.

Note: There is additional up to date tips in the README data file in Concept.

Related Documents

Title of Documentation	Reference Number
Concept Installation Instructions	840 USE 502 00
Concept User Manual	840 USE 503 00
Concept EFB User Manual	840 USE 505 00
Concept LL984 Block Library	840 USE 506 00

You can download these technical publications and other technical information from our website at *www.telemecanique.com*

User Comments

We welcome your comments about this document. You can reach us by e-mail at techpub@schneider-electric.com

General information about IEC block library

I

Overview

Introduction

This section contains general information about IEC block library.

What's in this Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
1	Parameterizing functions and function blocks	19

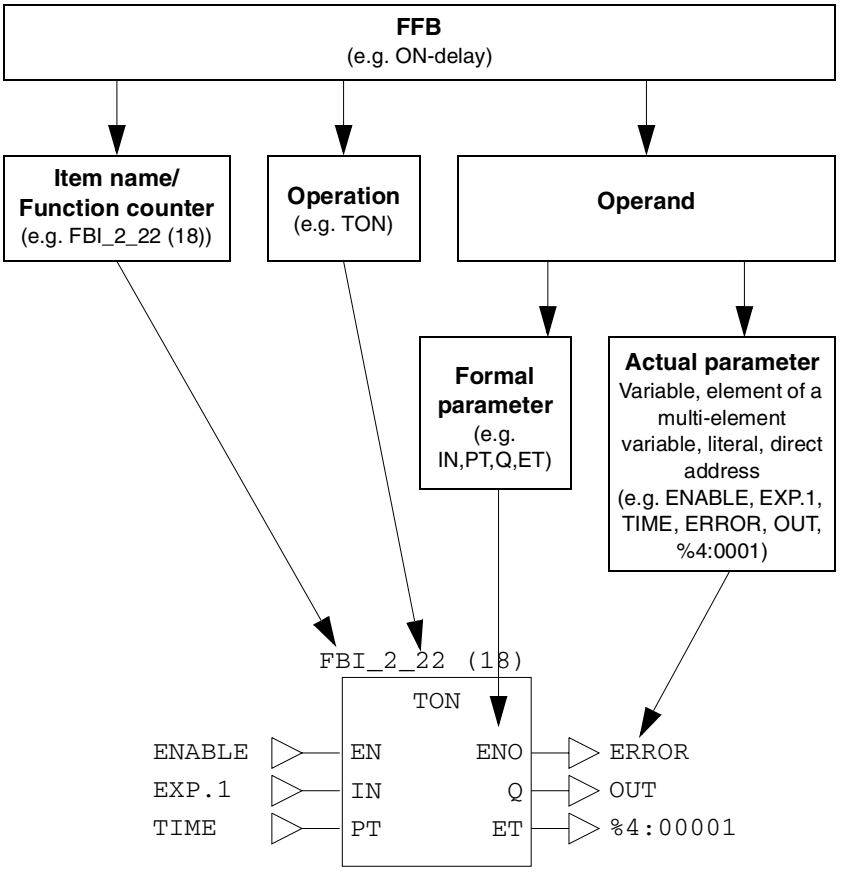
Parameterizing functions and function blocks

1

Parameterizing functions and function blocks

General

Each FFB consists of an operation, the operands needed for the operation and an instance name or function counter.



Operation

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

Operand

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

Formal/actual parameters

The formal parameter holds the place for an operand. During parameterization, an actual parameter is assigned to the formal parameter.

The actual parameter can be a variable, a multi-element variable, an element of a multi-element variable, a literal or a direct address.

Conditional/ unconditional calls

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

- Displayed EN
  conditional calls (the FFB is only processed if EN = 1)
- EN not displayed
  unconditional calls (FFB is always processed)

Note: If the EN input is not parameterized, it must be disabled. Any input pin that is not parameterized is automatically assigned a "0" value. Therefore, the FFB should never be processed.

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

Calling functions and function blocks in IL and ST

Information on calling functions and function blocks in IL (Instruction List) and ST (Structured Text) can be found in the relevant chapters of the user manual.

EFB descriptions

II

Overview

Introduction

These EFB descriptions are documented in alphabetical order.

Note: The number of inputs of individual EFBs can be increased to a maximum of 32 by changing the size of the FFB symbol vertically. As to which EFBs are concerned, please refer to the description of the individual EFBs.

What's in this Part?

This part contains the following chapters:

ABS_***: Absolute value computation

2

Overview

Introduction

This chapter describes the ABS_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function computes the absolute value of the input value and assigns the result to the output.

Data types of the ANY_NUM group can be processed.

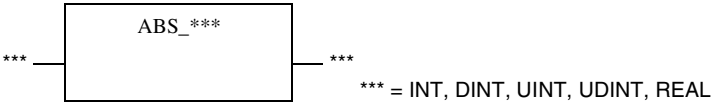
The Data types of the input and output values must be identical. A specific function is available for processing each different data type.

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

Representation

Symbol

Block representation:



Formula

$$OUT = |IN|$$

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN	INT, DINT, UINT, UDINT, REAL	Input value
OUT	INT, DINT, UINT, UDINT, REAL	Output value

Runtime error

Error message

If an invalid floating point number is applied to an input parameter from data type REAL, an Error message appears.

ACOS_REAL: Arc Cosine in Radians

3

Overview

Introduction

This chapter describes the ACOS_REAL block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

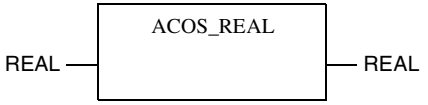
This Function computes the arc cosine of the input value and assigns the result in radians to the output.

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

Representation

Symbol

Block representation:



Formula

OUT = arc cos IN

Prerequisites:

$-1 \leq IN \leq 1$

The following applies:

$0 \leq OUT \leq \pi$

Parameter description

Description of the block parameters

Parameter	Data type	Meaning
IN	REAL	Input value
OUT	REAL	Output value in radians

Runtime error

Error message

If during execution of the Function there is a violation of the value range at the input or an unauthorized floating point number is present, an Error message appears.

ADD_***: Addition

4

Overview

Introduction

This chapter describes the ADD_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function adds up input values of the ANY_NUM group or the data type TIME and assigns the result to the output.

The data types of all input values and output values must be identical. A specific function is available for processing each different data type.

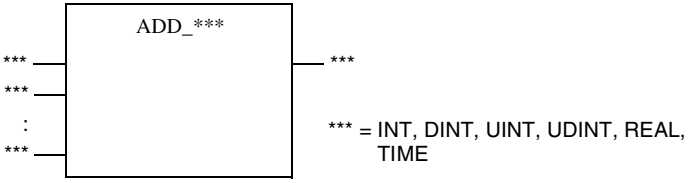
The number of inputs can be increased for all functions, except in ADD_TIME.

EN and ENO can be configured as additional parameters.

Representation

Symbol

Block representation:



Formula

INT, DINT, UINT, UDINT, REAL:

OUT = IN1 + IN2 + .. INn

TIME:

OUT = IN1 + IN2

Parameter description

Description of the block parameters

Parameter	Data type	Meaning
IN1	INT, DINT, UINT, UDINT, REAL, TIME	Summand
IN2	INT, DINT, UINT, UDINT, REAL, TIME	Summand
INn	INT, DINT, UINT, UDINT, REAL	Summand
OUT	INT, DINT, UINT, UDINT, REAL, TIME	Sum

Runtime error

Error message

An Error message appears if,

- the value range at the output is exceeded,
- an unauthorized floating point number is placed at an input parameter from data type REAL.

AND_***: AND function

5

Overview

Introduction

This chapter describes the AND_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function links (acc. to AND logic) the bit sequences at the inputs and assigns the result to the output. The AND operation is performed bit-by-bit.

Data types of the ANY_BIT group can be processed.

Note: This function is not available with Boolean variables in the LD (Ladder Diagram) programming language, since the same functionality can be achieved there with contacts.

The Data types of the input values and output values must be identical. A specific function is available for processing each different data type.

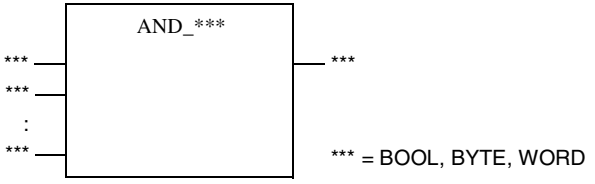
The number of inputs can be increased.

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

Representation

Symbol

Block representation:



Formula

OUT = IN1 & IN2 & INn

Parameter description

Description of the block parameters

Parameter	Data type	Meaning
IN1	BOOL, BYTE, WORD	Input bit sequence
IN2	BOOL, BYTE, WORD	Input bit sequence
INn	BOOL, BYTE, WORD	Input bit sequence
OUT	BOOL, BYTE, WORD	Output bit sequence

ASIN_REAL: Arc sine in radians

6

Overview

Introduction

This chapter describes the ASIN_REAL block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

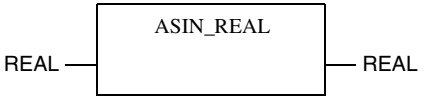
This Function calculates the arc sine of the input value and assigns the result in radians to the output.

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

Representation

Symbol

Block representation:



Formula

OUT = arc sin (IN)

Assuming:

IN -1 ≤ IN ≤ 1

Hence:

$$-\frac{\pi}{2} \leq \mathrm{OUT} \leq \frac{\pi}{2}$$

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN	REAL	Input value
OUT	REAL	Output value in radians

Runtime error

Error message

If during execution of the Function there is a violation of the authorized value range at the input or an unauthorized floating point number is present, an Error message appears.

ATAN_REAL: Arc tangent in radians

7

Overview

Introduction

This chapter describes the ATAN_REAL block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

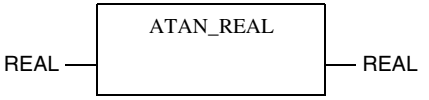
This Function calculates the arc tangent of the input value and assigns the result in radians to the output.

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

Representation

Symbol

Block representation:



Formula

OUT = arc tan (IN)

Hence:

$$-\frac{\pi}{2} \leq \mathrm{OUT} \leq \frac{\pi}{2}$$

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN	REAL	Input value
OUT	REAL	Output value in radians

Runtime error

Error message

If there is an unauthorized floating point number at the input, an Error message appears.

BOOL_TO_***: Type conversion

8

Overview

Introduction

This chapter describes the BOOL_TO_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This Function converts an input value from the Data type BOOL into a data type of the ANY_NUM group, or the data types BYTE, WORD or TIME.

A specific function is available for processing each different data type.

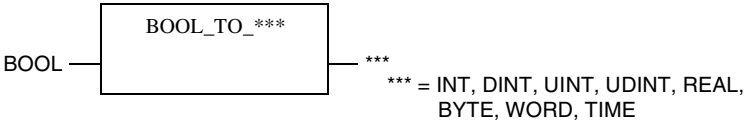
The least significant bit of the output value is set to the input value. All other bits of the output value are set to zero.

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

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN	BOOL	Input value
OUT	INT, DINT, UINT, UDINT, REAL, BYTE, WORD, TIME	Output value

BYTE_TO_***: Type conversion

9

Overview

Introduction

This chapter describes the BYTE_TO_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function converts an input value from the Data type BYTE into a data type of the ANY_NUM group, or the data types BOOL, WORD or TIME.

A specific function is available for processing each different data type.

Note: The EFB converts adhering strictly to IEC rules. Since this EFB has been realized as a generic function, there will also be some non-logic conversions, e.g. BYTE_ TO_TIME. It must be noted here, that the input byte pattern will be transferred into the most significant word of the output word.

When converting the data type BYTE into a data type of the ANY_NUM group or WORD, the bit pattern of the input is transferred to the least significant bits of the output. The high bits of the output are set to zero.

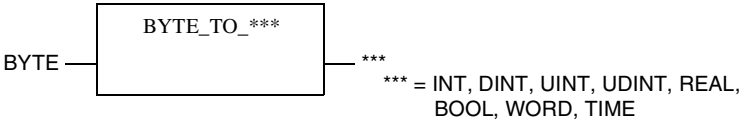
When converting the data type BYTE into the data type BOOL, the least significant bit of the input value is transferred to the output.

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

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN	BYTE	Input value
OUT	INT, DINT, UINT, UDINT, REAL, BOOL, WORD, TIME	Output value

Runtime error

Error message

If an unauthorized floating point number is created during the conversation into Data type REAL, an Error message appears.

COS_REAL: Cosine

10

Overview

Introduction

This chapter describes the COS_REAL block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This Function computes the cosine of the input value and assigns the result to the output. The input value must be entered in radians.

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

Representation

Symbol

Block representation:

Formula

OUT = cos (IN)

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN	REAL	Input value in radians
OUT	REAL	Output value

Runtime error

Error message

If there is an unauthorized floating point number at the input, an Error message appears.

CTD: Down counter

11

Overview

Introduction

This chapter describes the CTD block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function Block is used to count down INT values.

The Function Blocks for the counting of DINT, UDINT and UINT values can be found in the Extended library.

A "1"-signal at the LD input causes the value of the PV input to be assigned to the CV output. With each transition from "0" to "1" at the CD input, the value of CV is decreased by 1.

When CV is ≤ 0, Q output becomes "1".

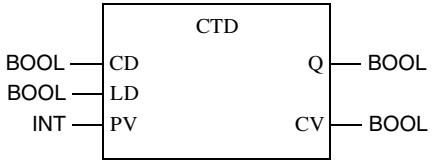
Note: The counter operates only up to the minimum value of output CV. No overflow occurs.

The parameters EN and ENO can be additionally configured.

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
CD	BOOL	Trigger input
LD	BOOL	Load data
PV	INT	Default value
Q	BOOL	Output
CV	INT	Count value (actual value)

CTU: Up counter

12

Overview

Introduction

This chapter describes the CTU block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function Block is used to count up INT values.

The Function Blocks for the counting of DINT, UDINT and UINT values can be found in the Extended library.

A "1" signal at the R input causes the value "0" to be assigned to the CV output. With each transition from "0" to "1" at the CU input, the value of CV is increased by 1.

When CV is ≥ PV, the Q output is set to "1".

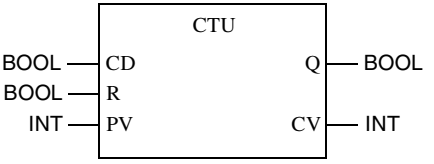
Note: The counter only operates up to the maximum value of output CV. No overflow occurs.

The parameters EN and ENO can be additionally configured.

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
CU	BOOL	Trigger input
R	BOOL	Reset
PV	INT	Default value
Q	BOOL	Output
CV	INT	Count value (actual value)

CTUD: Up/Down counter

13

Overview

Introduction

This chapter describes the CTUD block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function Block is used to count up and down INT values.

The Function Blocks for the counting of DINT, UDINT and UINT values can be found in the Extended library.

A "1" signal at the R input causes the value "0" to be assigned to the CV output. A "1"-signal at the LD input causes the value of the PV input to be assigned to the CV output. With each transition from "0" to "1" at the CU input, the value of CV is increased by 1. With each transition from "0" to "1" at the CD input, the value of CV is decreased by 1.

If there is a simultaneous "1"-signal at the CU and CD inputs, input CU (up counter) has precedence.

If there is a simultaneous "1"-signal at input R and input LD, input R has precedence.

When CV is ≥ PV, output QU becomes "1".

When CV is ≤ 0, output QD becomes "1".

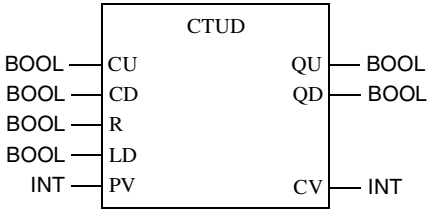
Note: The counter only operates up to the minimum value (down counting) or the maximum value (up counting) of output CV. No overflow occurs.

The parameters EN and ENO can be additionally configured.

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
CU	BOOL	Up counter trigger input
CD	BOOL	Down counter trigger input
R	BOOL	Reset
LD	BOOL	Load data
PV	INT	Default value
QU	BOOL	Up display
QD	BOOL	Down display
CV	INT	Count value (actual value)

DINT_EXPT_REAL: Exponentiation

14

Overview

Introduction

This chapter describes the DINT_EXPT_REAL block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

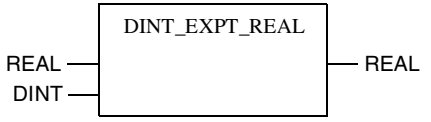
This Function is used to compute the exponentiation. The value at the input IN1 (base) is powered as an exponent using the value at input IN2 and the power is assigned to the output.

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

Representation

Symbol

Block representation:



Formula

OUT = IN1 exp IN2

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN1	REAL	Base
IN2	DINT	Exponent
OUT	REAL	Power

Runtime error

Error message

An Error message appears if,

- (IN1 = 0) & (IN2 < 0),
- an unauthorized floating point number is set at IN1 or
- the value range at the output has been exceeded.

DINT_TO_***: Type conversion

15

Overview

Introduction

This chapter describes the DINT_TO_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function converts an input value of Data type DINT into an output of data type INT, UDINT, UINT, REAL, TIME, BOOL, BYTE or WORD.

A specific function is available for processing each different data type.

Note: The EFB converts adhering strictly to IEC rules. Since this EFB has been realized as a generic function, there will also be a few non-logic conversions, e.g. DINT_ TO_BOOL.

When converting the data type DINT into the data type BOOL, BYTE or WORD, the least significant bits of the input value are transferred to the output.

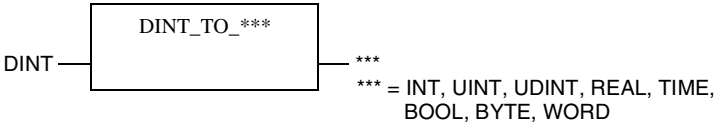
Negative input values can not be converted into data types UDINT, or UINT.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN	DINT	Input value
OUT	INT, UDINT, UINT, REAL, TIME, BOOL, BYTE, WORD	Output value

Runtime error

Error message

An Error message appears if,

- the value range of the output is exceeded or
- a negative input value is to be converted into an UDINT or UINT output value.

DIV_***: Division

16

Overview

Introduction

This chapter describes the DIV_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function divides the value at input IN1 with the value at input IN2 and assigns the result to the output.

The Data types of the input values and the output values must be identical. A specific function is available for processing each different data type.

When dividing data types of the ANY_INT group, any decimal point that might be in the result, will be truncated towards zero, e.g.

$7 \div 3 = 2$

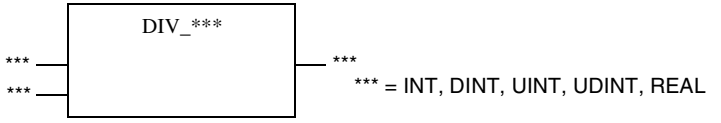
$(-7) \div 3 = -2$

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Formula

$OUT = ((IN1) \div (IN2))$

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN1	INT, DINT, UINT, UDINT, REAL	Dividend
IN2	INT, DINT, UINT, UDINT, REAL	Divisor
OUT	INT, DINT, UINT, UDINT, REAL	Quotient

Runtime error

Error message

An Error message appears if,

- $IN2 = 0$ or
- an unauthorized floating point number is set at an input parameter from data type REAL.

EQ_***: Equal to

17

Overview

Introduction

This chapter describes the EQ_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This Function checks the inputs for equality, i.e. the output becomes "1" if there is equality at all inputs; otherwise, the output remains at "0".

The Data types of the input values must be identical. A specific function is available for processing each different data type.

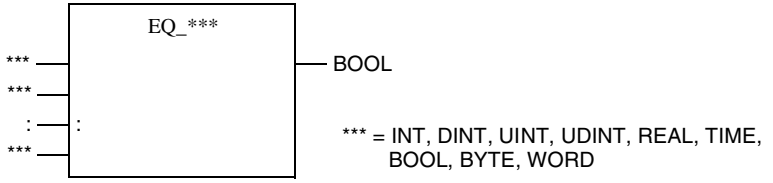
The number of inputs can be increased.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Formula

OUT = 1, if (IN1 = IN2) & (IN2 = IN3) & .. & ($IN_{(n-1)}$ = IN_n)

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN1	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	1. Input
IN2	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	2. Input
INn	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	n. input
OUT	BOOL	Output

Runtime error

Error message

Should an invalid floating point number be applied to an input parameter from data type REAL, an Error message appears.

EXP_REAL: Exponential function

18

Overview

Introduction

This chapter describes the EXP_REAL block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

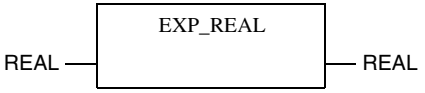
This Function is used to compute the exponential function. The value at the input is used as exponent to base e and the power is assigned to the output.

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

Representation

Symbol

Block representation:



Formula

OUT = e exp IN

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN	REAL	Exponent to base e
OUT	REAL	Power

Runtime error

Error message

An Error message appears if,

- an unauthorized floating point number is set at the input or
- the value range at the output has been exceeded.

F_TRIG: Falling edge detection

19

Overview

Introduction

This chapter describes the F_TRIG block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This Function block is used for the detection of falling edges 1 -> 0.

Note: This function block is not available in the LD (Ladder Diagram) programming language, since the same functionality can be achieved there with the "Contact - negative edge".

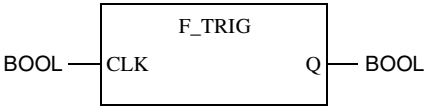
Output Q becomes "1" if there is a transition from "1" to "0" at the CLK input. The output will remain at "1" from one function block execution to the next; the output subsequently returns to "0".

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
CLK	BOOL	Timing input
Q	BOOL	Output

GE_***: Greater than or equal to

20

Overview

Introduction

This chapter describes the GE_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function checks the values of successive inputs for a falling sequence or equality.

The Data types of the input values must be identical. A specific function is available for processing each different data type.

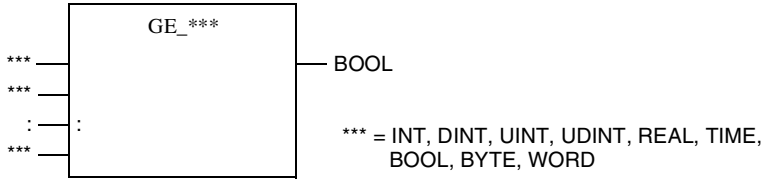
The number of inputs can be increased.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Formula

OUT = 1, if (IN1 ≥ IN2) & (IN2 ≥ IN3) & .. & ($IN_{(n-1)}$ ≥ IN_n)

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN1	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	1. Input
IN2	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	2. Input
INn	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	n. input
OUT	BOOL	Output

Runtime error

Error message

Should an invalid floating point number be applied to an input parameter from data type REAL, an Error message appears.

GT_***: Greater than

21

Overview

Introduction

This chapter describes the GT_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function checks the values of successive inputs for a falling sequence.

The Data types of the input values must be identical. A specific function is available for processing each different data type.

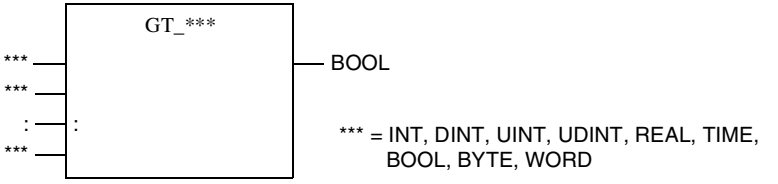
The number of inputs can be increased.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Formula

OUT = 1, if (IN1 > IN2) & (IN2 > IN3) & .. ($IN_{(n-1)}$ > IN_n)

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN1	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	1. Input
IN2	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	2. Input
INn	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	n. input
OUT	BOOL	Output

Runtime error

Error message

Should an invalid floating point number be applied to an input parameter from data type REAL, an Error message appears.

INT_EXPT_REAL: Exponentiation

22

Overview

Introduction

This chapter describes the INT_EXPT_REAL block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

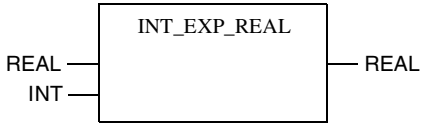
This Function is used to compute the exponentiation. The value at the input IN1 (base) is powered as an exponent using the value at input IN2 and the power is assigned to the output.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Formula

OUT = IN1 exp (IN2)

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN1	REAL	Base
IN2	INT	Exponent
OUT	REAL	Power

Runtime error

Error message

An error message appears if,

- (IN1 = 0) & (IN2 < 0),
- an unauthorized floating point number is set at IN1 or
- the value range at the output has been exceeded.

INT_TO_***: Type conversion

23

Overview

Introduction

This chapter describes the INT_TO_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function converts an input value of Data type INT into an output value of data type BOOL, BYTE, WORD, DINT, UDINT, UINT, REAL or TIME.

A specific function is available for processing each different data type.

Note: The EFB converts adhering strictly to IEC rules. Since this EFB has been realized as a generic function, there will also be a few non-logic conversions, e.g. INT_ TO_BOOL.

Negative input values can not be converted into data types UDINT, UINT, or TIME.

When converting an input value from the data type INT into data type WORD, the bit pattern from the input is transferred to the output without being modified.

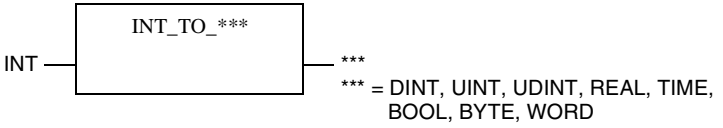
When converting an input value from the data type INT into data types BOOL or BYTE, the least significant bits of the input are transferred to the output.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN	INT	Input value
OUT	BOOL, BYTE, WORD, DINT, UDINT, UINT, REAL, TIME	Output value

Runtime error

Error message

An Error message appears if,

- the value range of the output is exceeded or
- a negative input value is to be converted into an UDINT, UINT, or TIME output value.

LE_***: Less than or equal to

24

Overview

Introduction

This chapter describes the LE_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function checks the values of successive inputs for a rising sequence or equality.

The Data types of the input values must be identical. A specific function is available for processing each different data type.

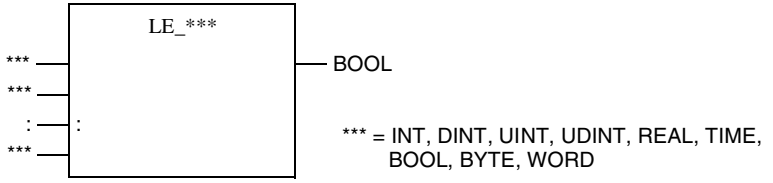
The number of inputs can be increased.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Formula

OUT = 1, if (IN1 ≤ In2) & (IN2 ≤ In3) & .. & ($IN_{(n-1)} \leq IN_n$)

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN1	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	1. Input
IN2	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	2. Input
INn	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	n. input
OUT	BOOL	Output

Runtime error

Error message

Should an invalid floating point number be applied to an input parameter from data type REAL, an Error message appears.

LIMIT_***: Limit

25

Overview

Introduction

This chapter describes the LIMIT_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This Function transfers the unchanged input value (IN) to the output if the input value is not less than the minimum value (MN) and not more than the maximum value (MX). If the input value (IN) is less than the minimum value (MN), the minimum value is transferred to the output. If the input value (IN) exceeds the maximum value (MX), the maximum value is transferred to the output.

Data types of the ANY_ELEM group can be processed.

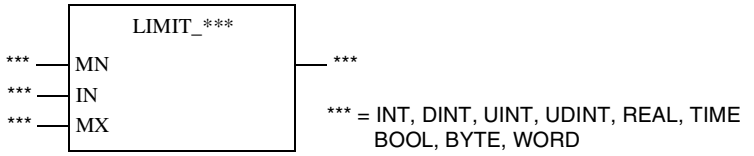
The Data types of the input values and output values must be identical. A specific function is available for processing each different data type.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Formula

OUT = IN, if (IN ≥ MN) & (IN ≤ MX)

OUT = MN, if (IN < MN)

OUT = MX, if (IN > MX)

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
MN	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	lower limit
IN	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	Input
MX	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	upper limit
OUT	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	Output

Runtime error

Error message	If there is an unauthorized floating point number at the input, an Error message appears.

LN_REAL: Natural logarithm

26

Overview

Introduction

This chapter describes the LN_REAL block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

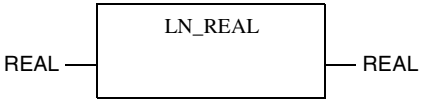
The Function calculates the natural logarithm of the input value and assigns the result to the output.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Formula

OUT = ln (IN)

Assuming:

IN > 0

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN	REAL	Input value
OUT	REAL	Output value

Runtime error

Error message

If during execution of the Function there is a violation of the value range at the input or an unauthorized floating point number is present, an Error message appears.

LOG_REAL: Base 10 Logarithm

27

Overview

Introduction

This chapter describes the LOG_REAL block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function calculates the logarithm to base 10 of the input value and assigns the result to the output.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:

Formula

OUT = log (IN)

Assuming:

IN > 0

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN	REAL	Input value
OUT	REAL	Output value

Runtime error

Error message

If during execution of the Function there is a violation of the value range at the input or an unauthorized floating point number is present, an Error message appears.

LT_***: Less than

28

Overview

Introduction

This chapter describes the LT_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function checks the values of successive inputs for a rising sequence.

The Data types of the input values must be identical. A specific function is available for processing each different data type.

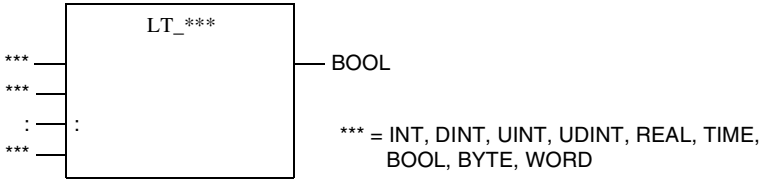
The number of inputs can be increased.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Formula

OUT = 1, if (IN1 < IN2) & (IN2 < IN3) & .. & ($IN_{(N-1)}$ < IN_n)

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN1	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	1. Input value
IN2	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	2. Input value
INn	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	n. input value
OUT	BOOL	Output value

Runtime error

Error message

Should an invalid floating point number be applied to an input parameter from data type REAL, an Error message appears.

MAX_***: Maximum value function

29

Overview

Introduction

This chapter describes the MAX_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function assigns the largest input value to the output.

Data types of the ANY_ELEM group can be processed.

The Data types of the input values and output values must be identical. A specific function is available for processing each different data type.

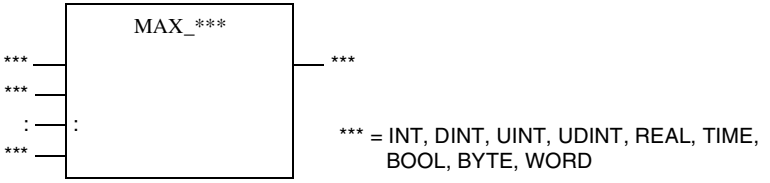
The number of inputs can be increased.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Formula

OUT = MAX {IN1, In2, .., In}

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN1	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	1. Input value
IN2	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	2. Input value
INn	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	n. input value
OUT	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	Maximum value

Runtime error

Error message	Should an invalid floating point number be applied to an input parameter from data type REAL, an Error message appears.

MIN_***: Minimum value function

30

Overview

Introduction

This chapter describes the MIN_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function assigns the smallest input value to the output.

Data types of the ANY_ELEM group can be processed.

The Data types of the input values and output values must be identical. A specific function is available for processing each different data type.

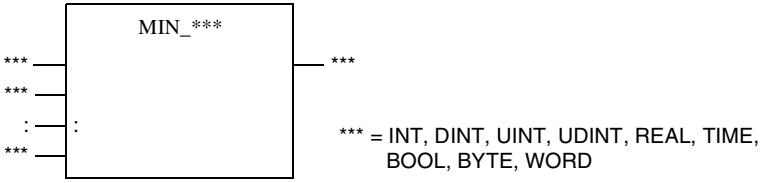
The number of inputs can be increased.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Formula

OUT = MIN {IN1, IN2, .., INn}

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN1	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	1. Input value
IN2	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	2. Input value
INn	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	n. input value
OUT	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	Minimum value

Runtime error

Error message	Should an invalid floating point number be applied to an input parameter from data type REAL, an Error message appears.

MOD_***: Modulo

31

Overview

Introduction

This chapter describes the MOD_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function divides the value at input IN1 by the value at input IN2 and assigns the rest of the division (modulo) to the output.

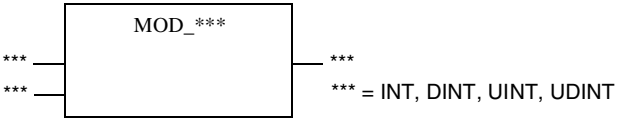
The Data types of the input values and output values must be identical. A specific function is available for processing each different data type.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Formula

OUT = IN1 mod IN2

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN1	INT, DINT, UINT, UDINT	Dividend
IN2	INT, DINT, UINT, UDINT	Divisor
OUT	INT, DINT, UINT, UDINT	Modulo

MOVE: Assignment

32

Overview

Introduction

This chapter describes the MOVE_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function assigns the input value to the output.

This is a generic function, i.e. the data type to be processed will be determined by the variable that was first assigned to the function.

If a direct address of a variable is to be assigned or vice versa, always assign the variable to the function first. A direct address at input and output of the function is not authorized since this does not allow a clear definition of the data type.

The Data types of the input and output values must be identical.

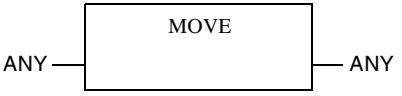
Note: This function can not be used in the LD (Ladder Diagram) programming language with the BOOL data type, since the same functionality can be achieved there with contacts and coils.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Formula

OUT = IN

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN	ANY	Input value
OUT	ANY	Output value

MUL_***: Multiplication

33

Overview

Introduction

This chapter describes the MUL_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function multiplies the input values and assigns the result to the output.

The Data types of the input values and output values must be identical. A specific function is available for processing each different data type.

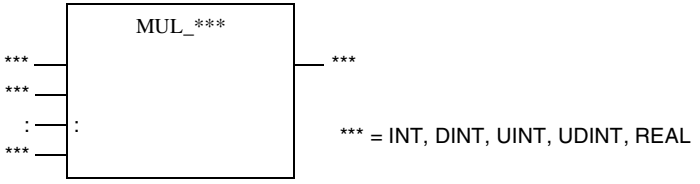
The number of inputs can be increased.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Formula

$OUT = IN1 \times IN2 \times .. \times IN_n$

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN1	INT, DINT, UINT, UDINT, REAL	Multiplicand (factor)
IN2	INT, DINT, UINT, UDINT, REAL	Multiplier (factor)
INn	INT, DINT, UINT, UDINT, REAL	Multiplier (Factor)
OUT	INT, DINT, UINT, UDINT, REAL	Product

Runtime error

Error message

An Error message appears if,

- an unauthorized floating point number is set at an input parameter of REAL data type or
- the value range at the output has been exceeded.

MUX_***: Multiplexer

34

Overview

Introduction

This chapter describes the MUX_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This Function transfers the respective input to the output depending on the value at input K.

The number of inputs can be increased.

The parameters EN and ENO can additionally be configured.

Example

K = 0: Input IN0 is transferred to the output

K = 1: Input IN1 is transferred to the output

K = 5: Input IN5 is transferred to the output

K = n: Input INn is transferred to the output

Data types

Data types of the ANY group can be processed.

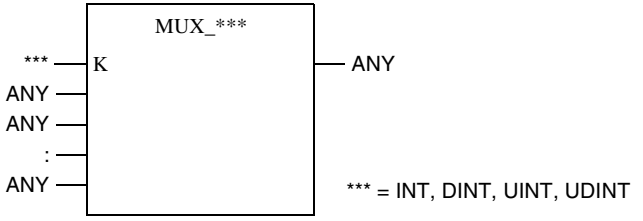
The Data types at the inputs IN0 through INn and at the output must be identical.

At input K, a specific function is available for processing each different data type (ANY_INT).

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
K	INT, DINT, UINT, UDINT	Selection input
IN0	ANY	0. Input
IN1	ANY	1. Input
IN2	ANY	2. Input
INn	ANY	n. input
OUT	ANY	Output

NE_***: Not equal to

35

Overview

Introduction

This chapter describes the NE_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function checks the input values for inequality.

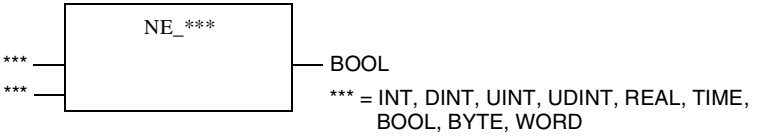
The Data types of the input values must be identical. A specific function is available for processing each different data type.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Formula

OUT = 1, if IN1 < > IN2

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN1	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	1. Input
IN2	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	2. Input
OUT	BOOL	Output

Runtime error

Error message

Should an invalid floating point number be applied to an input parameter from data type REAL, an Error message appears.

NOT_***: Negation

36

Overview

Introduction

This chapter describes the NOT_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function negates the input bit sequence bit-by-bit and assigns the result to the output.

Data types of the ANY_BIT group can be processed.

Note: This function is not available with Boolean variables in LD (Ladder Diagram) programming language, since the same functionality can be achieved there with (N.O.) contacts.

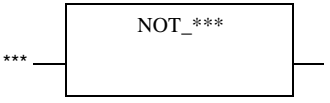
The Data types of the input and output values must be identical. A specific function is available for processing each different data type.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



*** = BOOL, BYTE, WORD

Formula

OUT = NOT IN

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN	BOOL, BYTE, WORD	Input bit sequence
OUT	BOOL, BYTE, WORD	negated bit sequence

OR_***: OR function

37

Overview

Introduction

This chapter describes the OR_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function links (acc. to OR logic) the bit sequences at the inputs and assigns the result to the output. The link is established bit-by-bit.

Data types of the ANY_BIT group can be processed.

Note: This function is not available with Boolean variables in LD (Ladder Diagram) programming language, since the same functionality can be achieved there with contacts.

The Data types of the input values and output values must be identical. A specific function is available for processing each different data type.

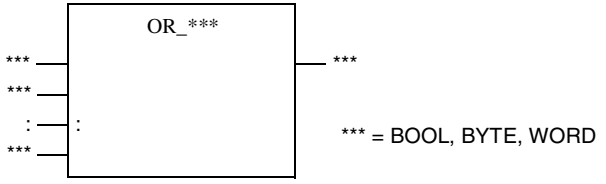
The number of inputs can be increased.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Formula

OUT = IN1 OR IN2 OR .. OR INn

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN1	BOOL, BYTE, WORD	Input bit sequence
IN2	BOOL, BYTE, WORD	Input bit sequence
INn	BOOL, BYTE, WORD	Input bit sequence
OUT	BOOL, BYTE, WORD	Output bit sequence

R_TRIG: Rising edge detection

38

Overview

Introduction

This chapter describes the R_TRIG block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This Function block is used for the detection of rising edges 0 -> 1.

Note: This function block is not available in the LD (Ladder Diagram), programming language, since the same functionality can be achieved there with the "Contact - positive edge".

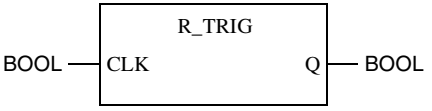
Output Q becomes "1" if there is a transition from "0" to "1" at the CLK input. The output remains at "1" from one function block execution to the next (one cycle); the output subsequently returns to "0".

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
CLK	BOOL	Timing input
Q	BOOL	Output

REAL_EXPT_REAL: Exponentiation

39

Overview

Introduction

This chapter describes the REAL_EXPT_REAL block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

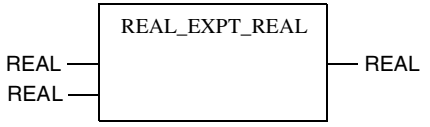
This Function is used to compute the exponentiation. The value at the input IN1 (base) is powered as an exponent using the value at input IN2 and the power is assigned to the output.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Formula

OUT = IN1 exp IN2

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN1	REAL	Base
IN2	REAL	Exponent
OUT	REAL	Power

Runtime error

Error message

An Error message appears if,

- an unauthorized floating point number is set at one of the inputs or
- the value range at the output has been exceeded.

REAL_TO_***: Type conversion

40

Overview

Introduction

This chapter describes the REAL_TO_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This Function converts an input value from the Data type REAL into a data type of the ANY_BIT, or ANY_INT group, or the data type TIME.

Note: The EFB converts adhering strictly to IEC rules. Since this EFB has been realized as a generic function, there will also be a few non-logic conversions, e.g. REAL_ TO_BOOL.

When converting to ANY_BIT, the least significant bits of the input value are transferred to the output.

Note: When converting from REAL to WORD the R_INT_WORD, and R_UINT_WORD blocks in the ANA_IO block library, and the REAL_AS_WORD block in the EXTENDED block library can also be used.

When converting to ANY_INT, and TIME, the IEC 559 rules for rounding are applied.

The parameters EN and ENO can additionally be configured.

Example

The following example shows how the IEC 559 rounding is applied.

1,4 -> 1

1,5 -> 2

2,5 -> 2

3,5 -> 4

4,5 -> 4

4,6 -> 5

Data type

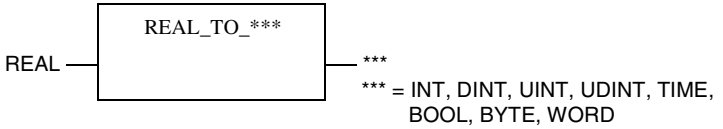
Negative input values can not be converted into data types UDINT, UINT, or TIME.

A specific function is available for processing each different data type.

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN	REAL	Input value
OUT	INT, DINT, UINT, UDINT, TIME, BOOL, BYTE, WORD	Output value

Runtime error

Error message

An Error message appears if,

- an unauthorized floating point number is set at the input,
- the value range of the output is exceeded or
- a negative input value is to be converted into an UDINT, UINT, or TIME output value.

REAL_TRUNC_***: Type conversion

41

Overview

Introduction

This chapter describes the REAL_TRUNC_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function converts (by truncating towards zero) an input value of the Data type REAL to a data type of the ANY_INT group.

The parameters EN and ENO can additionally be configured.

Example

The following example shows the converting procedure.

1,6 -> 1

-1,6 -> -1

1,4 -> 1

-1,4 -> -1

Data type

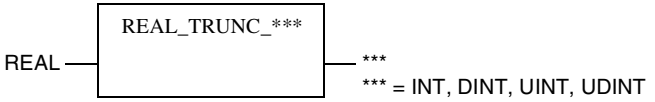
Negative input values can not be converted into data types UDINT, or UINT.

A specific function is available for processing each different data type.

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN	REAL	Input value
OUT	INT, DINT, UINT, UDINT	Output value

Runtime error

Error message

An Error message appears if,

- a negative input value is to be converted into an UDINT, or UINT,
- an unauthorized floating point number is set at the input or
- the value range of the output is exceeded.

ROL_***: Rotate left

42

Overview

Introduction

This chapter describes the ROL_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This Function rotates the bit sequence at the IN input circularly to the left by n bits (value at the N input).

Data types of the ANY_BIT group can be processed.

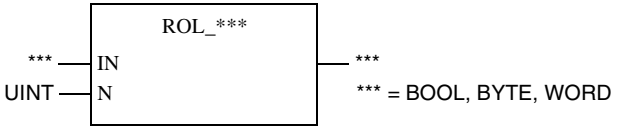
The Data types of the input value IN and the output value OUT must be identical. A specific function is available for processing each different data type.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN	BOOL, BYTE, WORD	Bit sequence for rotation
N	UINT	Number of spaces to be rotated
OUT	BOOL, BYTE, WORD	rotated bit sequence

ROR_***: Rotate right

43

Overview

Introduction

This chapter describes the ROR_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This Function rotates the bit sequence at the IN input circularly to the right by n bits (value at the N input).

Data types of the ANY_BIT group can be processed.

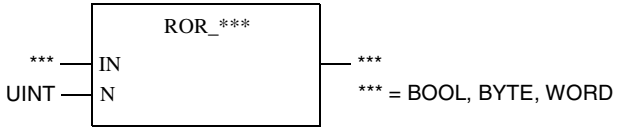
The Data types of the input value IN and the output value OUT must be identical. A specific function is available for processing each different data type.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN	BOOL, BYTE, WORD	Bit sequence for rotation
N	UINT	Number of spaces to be rotated
OUT	BOOL, BYTE, WORD	rotated bit sequence

RS: Bistable function block, reset dominant

44

Overview

Introduction

This chapter describes the RS_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block is used as RS memory with the property "Reset dominant".

Output Q1 becomes "1" when input S becomes "1". This state remains even if input S reverts back to "0". Output Q1 changes back to "0" when input R1 becomes "1". If the inputs S and R1 are "1" simultaneously, the dominating input R1 will set the output Q1 to "0".

When the function block is called for the first time, the initial state of Q1 is "0".

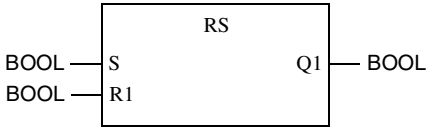
EN and ENO can be projected as additional parameters.

Note: This function block works with an internal unlocated variable and therefore has stored behavior. That means if output "Q1" is connected with a hardware output, the value of the output remains "1" when the PLC is switched off and then on again.

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
S	BOOL	Set
R1	BOOL	Reset (dominant)
Q1	BOOL	Output

SEL: Binary selection

45

Overview

Introduction

This chapter describes the SEL block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function is used for binary selection between two input values.

Depending on the state of input G, either input IN0 or input IN1 is transferred to output OUT.

G = 0 -> OUT = IN0

G = 1 -> OUT = IN1

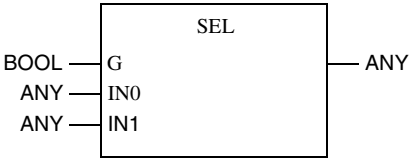
The Data types of the input values IN0 and IN1 and that of the output value OUT must be identical.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
G	BOOL	Selection input
IN0	ANY	Input 0
IN1	ANY	Input 1
OUT	ANY	Output

SHL_***: Shift left

46

Overview

Introduction

This chapter describes the SHL_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function shifts the bit sequence at input IN by n bits (value at input N) to the left.

Zeros are filled in from the right.

Data types of the ANY_BIT group can be processed.

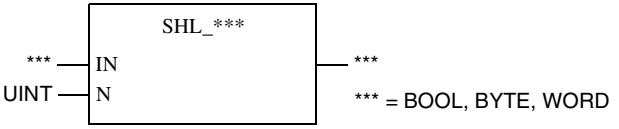
The Data types of the input value IN and the output value OUT must be identical. A specific function, respectively, is available for processing the different data types.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN	BOOL, BYTE, WORD	Bit sequence for shifting
N	UINT	Number of spaces to be shifted
OUT	BOOL, BYTE, WORD	shifted bit sequence

SHR_***: Shift right

47

Overview

Introduction

This chapter describes the SHR_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function shifts the bit sequence at input IN by n bits (value at input N) to the right.

Zeros are filled in from the left.

Data types of the ANY_BIT group can be processed.

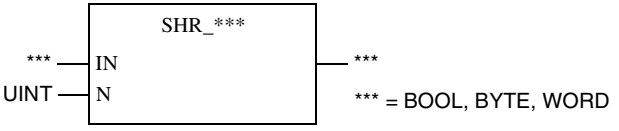
The Data types of the input value IN and the output value OUT must be identical. A specific function is available for processing each different data type.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN	BOOL, BYTE, WORD	Bit sequence for shifting
N	BOOL	Number of spaces to be shifted
OUT	BOOL, BYTE, WORD	shifted bit sequence

SIN_REAL: Sine

48

Overview

Introduction

This chapter describes the SIN_REAL block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

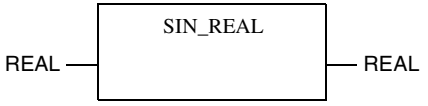
The Function calculates the sine of the input value and assigns the result to the output. The input value must be entered in radians.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Formula

OUT = sin IN

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN	REAL	Input value in radians
OUT	REAL	Output value

Runtime error

Error message

If there is an unauthorized floating point number at the input, an Error message appears.

SQRT_REAL: Square root

49

Overview

Introduction

This chapter describes the SQRT_REAL block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

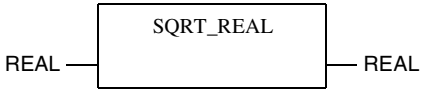
The Function calculates the square root of the input value and assigns the result to the output.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Formula

$$OUT = \sqrt{IN}$$

Assuming:

$IN \geq 0$

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN	REAL	Input value
OUT	REAL	Output value

Runtime error

Error message

If during execution of the Function there is a violation of the value range at the input or an unauthorized floating point number is present, an Error message appears.

SR: Bistable function block, set dominant

50

Overview

Introduction

This chapter describes the SR block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block is used as SR memory with the property "Set dominant".

Output Q1 becomes "1" when input S1 becomes "1". This state remains even if input S1 reverts again to "0". Output Q1 goes back to "0" again when input R becomes "1". If the inputs S1 and R are both "1" simultaneously, the dominating input S1 will set the output Q1 to "1".

When the function block is called for the first time, the initial state of Q1 is "0".

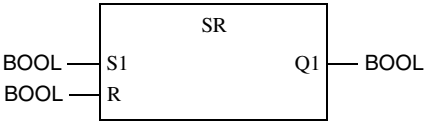
EN and ENO can be projected as additional parameters.

Note: This function block works with an internal unlocated variable and therefore has stored behavior. That means if output "Q1" is connected with a hardware output, the value of the output remains "1" when the PLC is switched off and then on again.

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
S1	BOOL	Set (dominant)
R	BOOL	Reset
Q	BOOL	Output

SUB_***: Subtraction

51

Overview

Introduction

This chapter describes the SUB_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function subtracts the value at input IN2 from the value at input IN1 and assigns the result to the output.

Data types of the ANY_NUM group and data type TIME can be processed.

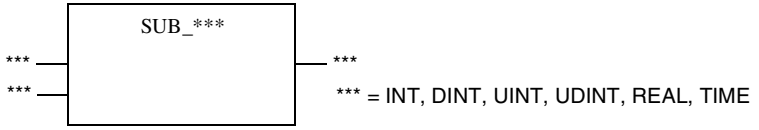
The Data types of the input values and output values must be identical. For processing the different data types, there is a specific function available.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Formula

OUT = IN1 - IN2

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN1	INT, DINT, UINT, UDINT, REAL, TIME	Minuend
IN2	INT, DINT, UINT, UDINT, REAL, TIME	Subtrahend
OUT	INT, DINT, UINT, UDINT, REAL, TIME	Difference

Runtime error

Error message

An Error message appears if,

- an unauthorized floating point number is set at an input parameter from data type REAL or
- the value range at the output has been exceeded.

TAN_REAL: Tangent

52

Overview

Introduction

This chapter describes the TAN_REAL block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

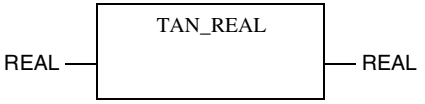
The Function calculates the tangent of the input value and assigns the result to the output. The input value must be entered in radians.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Formula

OUT = tan IN

Assuming:

$$IN \neq \frac{\langle 2n + 1 \rangle \times \pi}{2}$$

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN	REAL	Input value in radians
OUT	REAL	Output value

Runtime error

Error message

If during execution of the Function there is a violation of the value range at the input or an unauthorized floating point number is present, an Error message appears.

TIME_DIV_***: Division of time values

53

Overview

Introduction

This chapter describes the TIME_DIV_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function divides the value at input IN1 with the value at input IN2 and assigns the result to the output.

The parameters EN and ENO can additionally be configured.

Example

If there is a decimal point in the result, it will be truncated towards zero:

67 / 3 = 22

-67 / 3 = -22

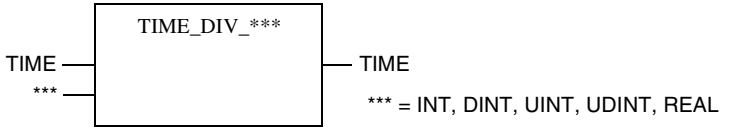
Data type

A specific function is available for processing each of the different Data types.

Representation

Symbol

Block representation:



Formula

OUT = IN1 / IN2

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN1	TIME	Dividend
IN2	INT, DINT, UINT, UDINT, REAL	Divisor
OUT	TIME	Quotient

Runtime error

Error message

Should the value range be exceeded at the output during the execution of the Function, an Error message appears.

TIME_MUL_***: Multiplication of time values

54

Overview

Introduction

This chapter describes the TIME_MUL_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function multiplies the value at input IN1 with the value at input IN2 and assigns the result to the output.

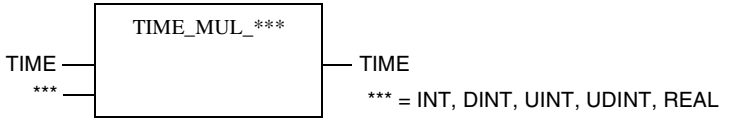
A specific function is available for processing each of the different Data types.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Formula

OUT = IN1 x IN2

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN1	TIME	Multiplicand (factor)
IN2	INT, DINT, UINT, UDINT, REAL	Multiplier (factor)
OUT	TIME	Product

Runtime error

Error message

Should the value range be exceeded at the output during the execution of the Function, an Error message appears.

TIME_TO_***: Type conversion

55

Overview

Introduction

This chapter describes the TIME_TO_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This Function converts an input value of the Data type TIME into a data type of the ANY_BIT, or ANY_NUM group.

A specific function is available for processing each different data type.

Note: The EFB converts adhering strictly to IEC rules. Since this EFB has been realized as a generic function, there will also be a few non-logic conversions, e.g. TIME_ TO_BOOL.

While converting an input value of data type TIME into an output value of data type BOOL, BYTE, WORD, INT or UINT, the least significant bits, respectively, are transferred from the input to the output.

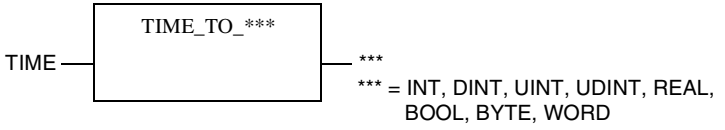
Note: When converting from TIME to WORD the TIME_AS_WORD block from the EXTENDED block library can also be used.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN	TIME	Input value
OUT	INT, DINT, UINT, UDINT, REAL, BOOL, BYTE, WORD	Output value

Runtime error

Error message

Should the value range be exceeded at the output during the execution of the Function, an Error message appears.

TOF: Off delay

56

Overview

Introduction

This chapter describes the TOF block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block is used as an Off delay.

A 0 -> 1 edge on input IN triggers a reset.

A 1 -> 0 edge on input IN starts the timer function.

If the elapsed time (output ET) reaches the value defined on input PT, output Q is set to "0".

When the function block is first called the initial state of ET is "0".

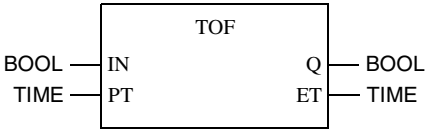
EN and ENO can be configured as additional parameters.

Note: Input EN cannot be used as pause function for the function block.
The elapsed time is still measured even when input EN becomes "0". If input EN becomes "1" again, output ET is updated and executes a jump.
If a pause function is required, function block TOF_P from the EXTENDED block library can be used.

Representation

Symbol

Block representation:



Parameter description

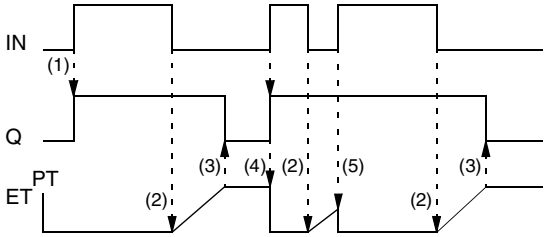
Description of the block parameters:

Parameter	Data type	Meaning
IN	BOOL	0 -> 1: Reset 1 -> 0: Start timer
PT	TIME	Preset delay time
Q	BOOL	Output
ET	TIME	Elapsed time

Detailed description

Timing diagram

Representation of the Off delay TOF:



(1) If IN becomes "1", Q becomes "1".
(2) If IN becomes "0", the internal time (ET) is started.
(3) If the internal time reaches the value of PT, Q becomes "0".
(4) If IN becomes "1", Q becomes "1", and the internal time is stopped/reset.
(5) If IN becomes "1" before the internal time has reached the value of PT, the internal time is stopped/reset without Q being set back to "0".

TON: On delay

57

Overview

Introduction

This chapter describes the TON block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block is used as an On delay.

A 1 -> 0 edge on input IN triggers a reset.

A 0 -> 1 edge on input IN starts the timer function.

If the elapsed time (output ET) reaches the value defined on input PT, output Q is set to "1".

When the function block is first called the initial state of ET is "0".

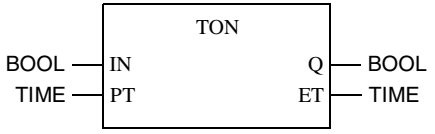
EN and ENO can be configured as additional parameters.

Note: Input EN cannot be used as pause function for the function block.
The elapsed time is still measured even when input EN becomes "0". If input EN becomes "1" again, output ET is updated and executes a jump.
If a pause function is required, function block TON_P from the EXTENDED block library can be used.

Representation

Symbol

Block representation:



Parameter description

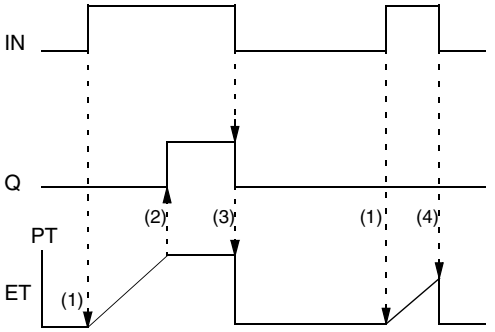
Description of the block parameters:

Parameter	Data type	Meaning
IN	BOOL	0 -> 1: Start timer 1 -> 0: Reset
PT	TIME	Preset delay time
Q	BOOL	Output
ET	TIME	Elapsed time

Detailed description

Timing diagram

Representation of the On delay TON:



(1) If IN becomes "1", the internal time (ET) is started.
(2) If the internal time reaches the value of PT, Q becomes "1".
(3) If IN becomes "0", Q becomes "0" and the internal time is stopped/reset.
(4) If IN becomes "0" before the internal time has reached the value of PT, the internal time is stopped/reset without Q being set to "1".

TP: Pulse

58

Overview

Introduction

This chapter describes the TP block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block is used for the generation of a pulse with defined duration.

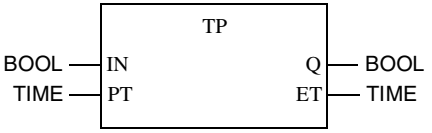
When the function block is first called the initial state of ET is "0".

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Parameter description

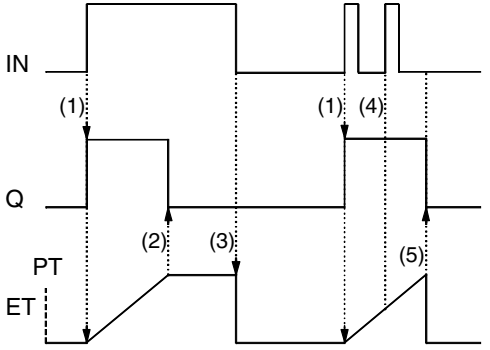
Description of the block parameters:

Parameter	Data type	Meaning
IN	BOOL	Trigger pulse
PT	TIME	Pulse duration presetting
Q	BOOL	Output
ET	TIME	internal time

Detailed description

Timing diagram

Representation of the TP pulse:



(1) If IN becomes "1", Q becomes "1" and the internal time (ET) starts.
(2) If the internal time reaches the value of PT, Q becomes "0" (independent of IN).
(3) The internal time stops/is reset if IN becomes "0".
(4) If the internal time has not reached the value of PT yet, the internal time is not affected by a clock at IN.
(5) If the internal time has reached the value of PT and IN is "0", the internal time stops/is reset and Q becomes "0".

UDINT_EXPT_REAL: Exponentiation

59

Overview

Introduction

This chapter describes the UDINT_EXPT_REAL block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

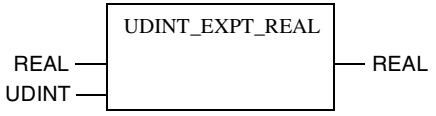
This Function is used to compute the exponentiation. The value at the input IN1 (base) is powered as an exponent using the value at input IN2 and the power is assigned to the output.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Formula

OUT = IN1 exp IN2

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN1	REAL	Base
IN2	UDINT	Exponent
OUT	REAL	Power

Runtime error

Error message

An Error message appears if,

- (IN1 = 0) & (IN2 < 0),
- an unauthorized floating point number is set at IN1 or
- the value range at the output has been exceeded.

UDINT_TO_***: Type conversion

60

Overview

Introduction

This chapter describes the UDINT_TO_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function converts an input value from Data type UDINT, into an output value from data type DINT, INT, UINT, REAL, TIME, BOOL, BYTE or WORD.

A specific function is available for processing each different data type.

Note: The EFB converts adhering strictly to IEC rules. Since this EFB has been realized as a generic function, there will also be a few non-logic conversions, e.g. UDINT_ TO_BOOL.

When converting the data type UDINT into the data type BOOL, BYTE or WORD, the least significant bits of the input value are transferred to the output.

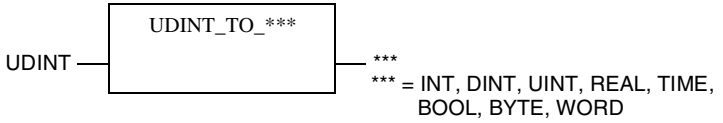
Note: When converting from UDINT to WORD the UDINT_AS_WORD block from the EXTENDED block library can also be used.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Parameter description

Block description:

Parameter	Data type	Meaning
IN	UDINT	Input value
OUT	DINT, INT, UINT, REAL, TIME, BOOL, BYTE, WORD	Output value

Runtime error

Error message

There will be an Error message if the value range of the output is exceeded.

UINT_EXPT_REAL: Exponentiation

61

Overview

Introduction

This chapter describes the UINT_EXPT_REAL block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

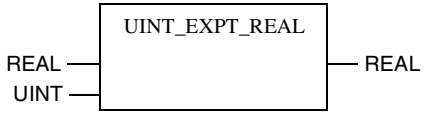
This Function is used to compute the exponential function. The value at the input IN1 (base) is powered as an exponent using the value at input IN2 and the power is assigned to the output.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Formula

OUT = IN1 exp IN2

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN1	REAL	Base
IN2	UINT	Exponent
OUT	REAL	Power

Runtime error

Error message

An Error message appears if,

- (IN1 = 0) & (IN2 < 0),
- an unauthorized floating point number appears at IN1 or
- the value range at the output has been exceeded.

UINT_TO_***: Type conversion

62

Overview

Introduction

This chapter describes the UINT_TO_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function converts an input value from Data type UINT to an output value of data type BOOL, BYTE, WORD, DINT, INT, UDINT, REAL or TIME.

A specific function is available for the processing of each different data type.

Note: The EFB converts strictly in accordance with IEC rules. Since this EFB has been realized as a generic function, there will also be a few non-logic conversions, e.g. UINT_ TO_BOOL.

When converting an input value of data type UINT into data type WORD, the bit pattern of the input is transferred to the output without being modified.

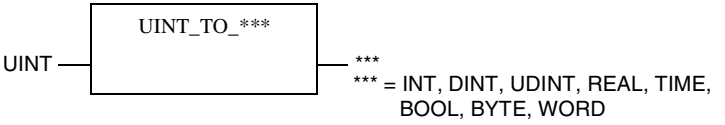
When converting an input value of data type UINT into the data types BOOL or BYTE, the least significant bits of the input are transferred to the output.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN	UINT	Input value
OUT	BOOL, BYTE, WORD, DINT, INT, UDINT, REAL, TIME	Output value

Runtime error

Error message

There will be an Error message if the value range of the output is exceeded.

WORD_TO_***: Type conversion

63

Overview

Introduction

This chapter describes the WORD_TO_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This Function converts an input value from the Data type WORD into a data type of the ANY_NUM group, or the data types BOOL, BYTE or TIME.

A specific function is available for the processing of each different data type.

Note: The EFB converts strictly in accordance with IEC rules. Since this EFB has been realized as a generic function, there will also be a few non-logic conversions, e.g. WORD_ TO_TIME. It must be noted here, that the input byte pattern will be transferred into the most significant word of the output word.

When converting the data WORD to the data type DINT, UDINT or REAL, the bit pattern of the input is transferred to the least significant bits of the output. The most significant bits of the output are set to zero.

When converting the data type WORD to the data type BOOL or BYTE, the least significant bits of the input value are transferred to the output.

Note: When converting from WORD to REAL the W_INT_REAL and W_UINT_REAL blocks from the ANA_IO block library and the WORD_AS_REAL block from the EXTENDED block library can also be used.

Note: When converting from WORD to TIME the WORD_AS_TIME block from the EXTENDED block library can also be used.

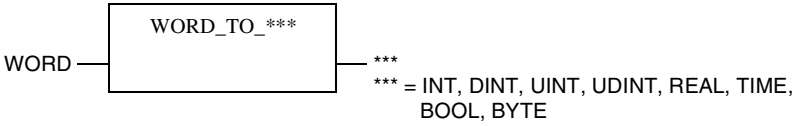
Note: When converting from WORD to UDINT the WORD_AS_UDINT block from the EXTENDED block library can also be used.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN	WORD	Input value
OUT	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE	Output value

Runtime error

Error message

If an unauthorized floating point number is created during the conversion into Data type REAL, an Error message will appear.

XOR_***: Exclusive OR function

64

Overview

Introduction

This chapter describes the XOR_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function links (acc. to exclusive OR logic) the bit sequences at the inputs and returns the result at the output. The AND operation is performed bit-by-bit.

Data types of the ANY_BIT group can be processed.

The Data types of the input values and output values must be identical. A specific function is available for the processing of each different data type.

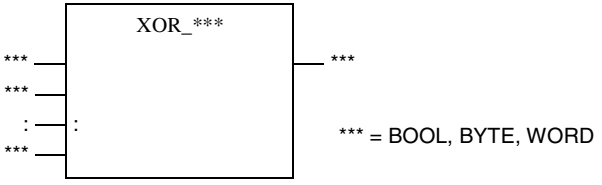
The number of inputs can be increased.

The parameters EN and ENO can additionally be configured.

Representation

Symbol

Block representation:



Formula

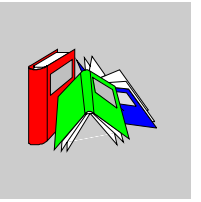
OUT = IN1 XOR IN2 XOR .. XOR INn

Parameter description

Description of the block parameters:

Parameter	Data type	Meaning
IN1	BOOL, BYTE, WORD	Input bit sequence
IN2	BOOL, BYTE, WORD	Input bit sequence
INn	BOOL, BYTE, WORD	Input bit sequence
OUT	BOOL, BYTE, WORD	Output bit sequence

Glossary



A

active Window — The window, which is currently selected. Only one window can be active at any given time. When a window is active, the color of the title bar changes, so that it is distinguishable from the other windows. Unselected windows are inactive.

Actual Parameters — Current connected Input / Output Parameters.

Addresses — (Direct) addresses are memory ranges on the PLC. They are located in the State RAM and can be assigned Input/Output modules.
The display/entry of direct addresses is possible in the following formats:

- Standard Format (400001)
- Separator Format (4:00001)
- Compact format (4:1)
- IEC Format (QW1)

ANL_IN — ANL_IN stands for the "Analog Input" data type and is used when processing analog values. The 3x-References for the configured analog input module, which were specified in the I/O component list, are automatically assigned to the data type and should therefore only be occupied with Unlocated Variables.

ANL_OUT — ANL_OUT stands for the "Analog Output" data type and is used when processing analog values. The 4x-References for the configured analog output module, which were specified in the I/O component list, are automatically assigned to the data type and should therefore only be occupied with Unlocated Variables.

ANY — In the present version, "ANY" covers the BOOL, BYTE, DINT, INT, REAL, UDINT, UINT, TIME and WORD elementary data types and related Derived Data Types.

ANY_BIT In the present version, "ANY_BIT" covers the BOOL, BYTE and WORD data types.

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

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

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

ANY_REAL In the present version, "ANY_REAL" covers the REAL data type.

Application Window The window contains the workspace, menu bar and the tool bar for the application program. The name of the application program appears in the title bar. An application window can contain several Document windows. In Concept the application window corresponds to a Project.

Argument Synonymous with Actual parameters.

ASCII-Mode The ASCII (American Standard Code for Information Interchange) mode is used to communicate with various host devices. ASCII works with 7 data bits.

Atrium The PC based Controller is located on a standard AT board, and can be operated within a host computer in an ISA bus slot. The module has a motherboard (requires SA85 driver) with two slots for PC104 daughter-boards. In this way, one PC104 daughter-board is used as a CPU and the other as the INTERBUS controller.

B

Backup file (Concept-EFB) The backup file is a copy of the last Source coding file. The name of this backup file is "backup??.c" (this is assuming that you never have more than 100 copies of the source coding file). The first backup file has the name "backup00.c". If you have made alterations to the Definitions file which do not cause any changes to the EFB interface, the generation of a backup file can be stopped by editing the source coding file (**Objects** → **Source**). If a backup file is created, the source file can be entered as the name.

Base 16 literals

Base 16 literals are used to input whole number values into the hexadecimal system. The base must be denoted using the prefix 16#. The values can not have any signs (+/-). Single underscores (_) between numbers are not significant.

Example
16#F_F or 16#FF (decimal 255)
16#E_0 or 16#E0 (decimal 224)

Base 2 literals

Base 2 literals are used to input whole number values into the dual system. The base must be denoted using the prefix 2#. The values can not have any signs (+/-). Single underscores (_) between numbers are not significant.

Example
2#1111_1111 or 2#11111111 (decimal 255)
2#1110_0000 or 2#11100000 (decimal 224)

Base 8 literals

Base 8 literals are used to input whole number values in the octosystem. The base must be denoted using the prefix 8#. The values can not have any signs (+/-). Single underscores (_) between numbers are not significant.

Example
8#3_77 or 8#377 (decimal 255)
8#34_0 or 8#340 (decimal 224)

Binary Connections

Connections between FFB outputs and inputs with the data type BOOL.

Bit sequence

A data element, which consists of one or more bits.

BOOL

BOOL stands for the data type "boolean". The length of the data element is 1 bit (occupies 1 byte in the memory). The value range for the variables of this data type is 0 (FALSE) and 1 (TRUE).

Bridge

A bridge is a device which connects networks. It enables communication between nodes on two networks. Each network has its own token rotation sequence - the token is not transmitted via the bridge.

BYTE

BYTE stands for the data type "bit sequence 8". Entries are made as base 2 literal, base 8 literal or base 16 literal. The length of the data element is 8 bits. A numerical value range can not be assigned to this data type.

C

Clipboard
The clipboard is a temporary memory for cut or copied objects. These objects can be entered in sections. The contents of the clipboard are overwritten with each new cut or copy.

Coil
A coil is a LD element which transfers the status of the horizontal connection on its left side, unchanged, to the horizontal connection on its right side. In doing this, the status is saved in the relevant variable/direct address.

Compact format (4:1)
The first digit (the Reference) is separated from the address that follows by a colon (:) where the leading zeros are not specified.

Constants
Constants are Unlocated variables, which are allocated a value that cannot be modified by the logic program (write protected).

Contact
A contact is a LD element, which transfers a status on the horizontal link to its right side. This status comes from the boolean AND link of the status of the horizontal link on the left side, with the status of the relevant variable/direct address. A contact does not change the value of the relevant variable/direct address.

D

Data transfer settings
Settings which determine how information is transferred from your programming device to the PLC.

Data Types
The overview shows the data type hierarchy, as used for inputs and outputs of functions and function blocks. Generic data types are denoted using the prefix "ANY".

- ANY_ELEM
  - ANY_NUM
    ANY_REAL (REAL)
    ANY_INT (DINT, INT, UDINT, UINT)
  - ANY_BIT (BOOL, BYTE, WORD)
  - TIME
- System Data types (IEC Extensions)
- Derived (from "ANY" data types)

DCP I/O drop

A remote network with a super-ordinate PLC can be controlled using a Distributed Control Processor (D908). When using a D908 with remote PLC, the super-ordinate PLC considers the remote PLC as a remote I/O drop. The D908 and the remote PLC communicate via the system bus, whereby a high performance is achieved with minimum effect on the cycle time. The data exchange between the D908 and the super-ordinate PLC takes place via the remote I/O bus at 1.5Mb per second. A super-ordinate PLC can support up to 31 D908 processors (addresses 2-32).

DDE (Dynamic Data Exchange)

The DDE interface enables a dynamic data exchange between two programs in Windows. The user can also use the DDE interface in the extended monitor to call up their own display applications. With this interface, the user (i.e. the DDE client) can not only read data from the extended monitor (DDE server), but also write data to the PLC via the server. The user can therefore alter data directly in the PLC, while monitoring and analyzing results. When using this interface, the user can create their own "Graphic Tool", "Face Plate" or "Tuning Tool" and integrate it into the system. The tools can be written in any language, i.e. Visual Basic, Visual C++, which supports DDE. The tools are invoked when the user presses one of the buttons in the Extended Monitor dialog field. Concept Graphic Tool: Configuration signals can be displayed as a timing diagram using the DDE connection between Concept and Concept Graphic Tool.

Declaration

Mechanism for specifying the definition of a language element. A declaration usually covers the connection of an identifier to a language element and the assignment of attributes such as data types and algorithms.

Definitions file (Concept-EFB)

The definitions file contains general descriptive information on the selected EFB and its formal parameters.

Defragmenting

With defragmenting, unanticipated gaps (e.g. resulting from deleting unused variables) are removed from memory.

Derived Data Type

Derived data types are data types, which are derived from Elementary Data Types and/or other derived data types. The definition of the derived data types is found in the Concept data type editor.
A distinction is made between global data types and local data types.

Derived Function Block (DFB)
A derived function block represents the invocation of a derived function block type. Details of the graphic form of the invocation can be found in the "Functional block (instance)". In contrast to the invocation of EFB types, invocations of DFB types are denoted by double vertical lines on the left and right hand side of the rectangular block symbol.
The output side of a derived function block is created in FBD language, LD language, ST language, IL language, but only in the current version of the programming system. Derived functions can also not be defined in the current version.
A distinction is made between local and global DFBs.

DFB Code
The DFB code is the section's DFB code which can be executed. The size of the DFB code is mainly dependent upon the number of blocks in the section.

DFB instance data
The DFB instance data is internal data from the derived function blocks used in the program.

DINT
DINT stands for the data type "double length whole number (double integer)". Entries are made as integer literal, base 2 literal, base 8 literal or base 16 literal. The length of the data element is 32 bits. The value range for variables of this data type reaches from -2 exp (31) to 2 exp (31) -1.

Direct Representation
A method of displaying variables in the PLC program, from which the assignment to the logical memory can be directly - and indirectly to the physical memory - derived.

Document Window
A window within an application window. Several document windows can be open at the same time in an application window. However, only one document window can ever be active. Document windows in Concept are, for example, sections, the message window, the reference data editor and the PLC configuration.

DP (PROFIBUS)
DP = Remote Peripheral

Dummy
An empty file, which consists of a text heading with general file information, such as author, date of creation, EFB designation etc. The user must complete this dummy file with further entries.

DX Zoom
This property enables the user to connect to a programming object, to monitor and, if necessary change, its data value.

E

EFB code

The EFB code is the executable code of all EFBs used. In addition the used EFBs count in DFBs.

Elementary functions/ function blocks (EFB)

Identifier for Functions or Function blocks, whose type definitions are not formulated in one of the IEC languages, i.e. whose body for example can not be modified with the DFB editor (Concept-DFB). EFB types are programmed in "C" and are prepared in a pre-compiled form using libraries.

EN / ENO (Enable / Error signal)

If the value of EN is equal to "0" when the FFB is invoked, the algorithms that are defined by the FFB will not be executed and all outputs keep their previous values. The value of ENO is in this case automatically set to "0". If the value of EN is equal to "1", when the FFB is invoked, the algorithms which are defined by the FFD will be executed. After the error-free execution of these algorithms, the value of ENO is automatically set to "1". If an error occurs during the execution of these algorithms, ENO is automatically set to "0". The output behavior of the FFB is independent of whether the FFBs are invoked without EN/ENO or with EN=1. If the EN/ENO display is switched on, it is imperative that the EN input is switched on. Otherwise, the FFB is not executed. The configuration of EN and ENO is switched on or off in the Block Properties dialog box. The dialog box can be invoked with the **Objects** → **Properties...**menu command or by double-clicking on the FFB.

Error

If an error is recognized during the processing of a FFB or a step (e.g. unauthorized input values or a time error), an error message appears, which can be seen using the **Online** → **Event Viewer...**menu command. For FFBs, the ENO output is now set to "0".

Evaluation

The process, through which a value is transmitted for a Function or for the output of a Function block during Program execution.

Expression

Expressions consist of operators and operands.

F

FFB (Functions/ Function blocks)

Collective term for EFB (elementary functions/function blocks) and DFB (Derived function blocks)

Field variables A variable, which is allocated a defined derived data type with the key word ARRAY (field). A field is a collection of data elements with the same data type.

FIR Filter (Finite Impulse Response Filter) a filter with finite impulse answer

Formal parameters Input / Output parameters, which are used within the logic of a FFB and led out of the FFB as inputs/outputs.

Function (FUNC) A program organization unit, which supplies an exact data element when processing. a function has no internal status information. Multiple invocations of the same function using the same input parameters always supply the same output values.
Details of the graphic form of the function invocations can be found in the definition "Functional block (instance)". In contrast to the invocations of the function blocks, function invocations only have a single unnamed output, whose name is the same as the function. In FBD each invocation is denoted by a unique number via the graphic block, this number is automatically generated and can not be altered.

Function block (Instance) (FB) A function block is a program organization unit, which correspondingly calculates the functionality values that were defined in the function block type description, for the outputs and internal variable(s), if it is invoked as a certain instance. All internal variable and output values for a certain function block instance remain from one function block invocation to the next. Multiple invocations of the same function block instance with the same arguments (input parameter values) do not therefore necessarily supply the same output value(s).
Each function block instance is displayed graphically using a rectangular block symbol. The name of the function block type is stated in the top center of the rectangle. The name of the function block instance is also stated at the top, but outside of the rectangle. It is automatically generated when creating an instance, but, depending on the user's requirements, it can be altered by the user. Inputs are displayed on the left side of the block and outputs are displayed on the right side. The names of the formal input/output parameters are shown inside the rectangle in the corresponding places.
The above description of the graphic display is especially applicable to the function invocations and to DFB invocations. Differences are outlined in the corresponding definitions.

Function Block Dialog (FBD) One or more sections, which contain graphically displayed networks from Functions, Function blocks and Connections.

Function block type A language element, consisting of: 1. the definition of a data structure, divided into input, output and internal variables; 2. a set of operations, which are performed with elements of the data structure, when a function block type instance is invoked. This set of operations can either be formulated in one of the IEC languages (DFB type) or in "C" (EFB type). A function block type can be instanced (invoked) several times.

Function Number

The function number is used to uniquely denote a function in a program or DFB. The function number can not be edited and is automatically assigned. The function number is always formed as follows: .n.m

n = Number of the section (consecutive numbers)
m = Number of the FFB object in the section (current number)

G

Generic Data Type

A data type, which stands in place of several other data types.

Generic literals

If the literal's data type is not relevant, simply specify the value for the literal. If this is the case, Concept automatically assigns the literal a suitable data type.

Global Data

Global data are Unlocated variables.

Global derived data types

Global derived data types are available in each Concept project and are occupied in the DFB directory directly under the Concept directory.

Global DFBs

Global DFBs are available in each Concept project. The storage of the global DFBs is dependant upon the settings in the CONCEPT.INI file.

Global macros

Global macros are available in each Concept project and are stored in the DFB directory directly under the Concept directory.

Groups (EFBs)

Some EFB libraries (e.g. the IEC library) are divided into groups. This facilitates locating the EFBs especially in expansive libraries.

H

Host Computer

Hardware and software, which support programming, configuring, testing, operating and error searching in the PLC application as well as in a remote system application, in order to enable source documentation and archiving. The programming device can also be possibly used for the display of the process.

I

I/O Map
The I/O and expert modules from the various CPUs are configured in the I/O map.

Icon
Graphical representation of different objects in Windows, e.g. drives, application programs and document windows.

IEC 61131-3
International standard: Programmable Logic Controls - Part 3: Programming languages.

IEC Format (QW1)
There is an IEC type designation in initial position of the address, followed by the five-figure address.

- %0x12345 = %Q12345
- %1x12345 = %I12345
- %3x12345 = %IW12345
- %4x12345 = %QW12345

IEC name conventions (identifier)
An identifier is a sequence of letters, numbers and underscores, which must begin with either a letter or underscore (i.e. the name of a function block type, an instance, a variable or a section). Letters of a national typeface (i.e.: ö,ü, é, õ) can be used, except in project and DFB names.
Underscores are significant in identifiers; e.g. "A_BCD" and "AB_CD" are interpreted as two separate identifiers. Several leading and multiple successive underscores are not allowed.
Identifiers should not contain any spaces. No differentiation is made between upper and lower case, e.g. "ABCD" and "abcd" are interpreted as the same identifier.
Identifiers should not be Keywords.

IEC Program Memory
The IEC program memory consists of the program code, EFB code, the section data and the DFB instance data.

IIR Filter
(Infinite Impulse Response Filter) a filter with infinite impulse answer

Initial step
The first step in a sequence. A step must be defined as an initial step for each sequence. The sequence is started with the initial step when first invoked.

Initial value
The value, which is allocated to a variable when the program is started. The values are assigned in the form of literals.

Input bits (1x references)

The 1/0 status of the input bits is controlled via the process data, which reaches from an input device to the CPU.

> **Note:** The x, which follows the initial reference type number, represents a five-figure storage location in the user data memory, i.e. the reference 100201 signifies an output or marker bit at the address 201 in the State RAM.

Input parameter (Input)

Upon invocation of a FFB, this transfers the corresponding argument.

Input words (3x references)

An input word contains information, which originates from an external source and is represented by a 16 bit number. A 3x register can also contain 16 sequential input bits, which were read into the register in binary or BCD (binary coded decimal) format. Note: The x, which follows the initial reference type number, represents a five-figure storage location in the user data memory, i.e. the reference 300201 signifies a 16-bit input word at the address 201 in the State RAM.

Instance Name

An identifier, which belongs to a certain function block instance. The instance name is used to clearly denote a function block within a program organization unit. The instance name is automatically generated, but it can be edited. The instance name must be unique throughout the whole program organization unit, and is not case sensitive. If the name entered already exists, you will be warned and you will have to choose another name. The instance name must comply with the IEC name conventions otherwise an error message appears. The automatically generated instance name is always formed as follows: FBI_n_m

FBI = Function Block Instance
n = Number of the section (consecutive numbers)
m = Number of the FFB object in the section (current number)

Instancing

Generating an Instance.

Instruction (IL)

Instructions are the "commands" of the IL programming language. Each instruction begins on a new line and is performed by an operator with a modifier if necessary, and if required for the current operation, by one or more operands. If several operands are used, they are separated by commas. A character can come before the instruction, which is then followed by a colon. The comment must, if present, be the last element of the line.

Instruction (LL984)

When programming electrical controls, the user must implement operation-coded instructions in the form of picture objects, which are divided into a recognizable contact form. The designed program objects are, on a user level, converted to computer usable OP codes during the download process. The OP codes are decoded in the CPU and processed by the firmware functions of the controller in a way that the required control is implemented.

Instruction (ST)

Instructions are "commands" of the ST programming language. Instructions must be completed by semicolons. Several instructions can be entered in one line (separated by semicolons).

Instruction list (IL)

IL is a text language according to IEC 1131, which is shown in operations, i.e. conditional or unconditional invocations of Functions blocks and Functions, conditional or unconditional jumps etc. through instructions.

INT

INT stands for the data type "whole number (integer)". Entries are made as integer literal, base 2 literal, base 8 literal or base 16 literal. The length of the data element is 16 bits. The value range for variables of this datatype reaches from -2 exp (15) to 2 exp (15) -1.

Integer literals

Integer literals are used to input whole number values into the decimal system. The values can have a preceding sign (+/-). Single underscores (_) between numbers are not significant.

Example
-12, 0, 123_456, +986

INTERBUS (PCP)

The new INTERBUS (PCP) I/O drop type is entered into the Concept configurator, to allow use of the INTERBUS PCP channel and the INTERBUS process data pre-processing (PDV). This I/O drop type is assigned the INTERBUS switching module 180-CRP-660-01.
The 180-CRP-660-01 differs from the 180-CRP-660-00 only in the fact that it has a clearly larger I/O range in the control state RAM.

Invocation

The process by which the execution of an operation is initiated.

J

Jump

Element of the SFC language. Jumps are used to skip zones in the sequence.

K

Keywords

Keywords are unique combinations of characters, which are used as special syntactical components, as defined in Appendix B of the IEC 1131-3. All keywords which are used in the IEC 1131-3 and therefore in Concept, are listed in Appendix C of the IEC 1131-3. These keywords may not be used for any other purpose, i.e. not as variable names, section names, instance names etc.

L

Ladder Diagram (LD)

Ladder Diagram is a graphic programming dialog according to IEC1131, which is optically oriented to the "rung" of a relay contact plan.

Ladder Logic 984 (LL)

The terms Ladder Logic and Ladder Diagram refer to the word Ladder being executed. In contrast to a circuit diagram, a ladder diagram is used by electrotechnicians to display an electrical circuit (using electrical symbols), which should show the course of events and not the existing wires, which connect the parts with each other. A usual user interface for controlling the actions of automation devices permits a Ladder Diagram interface, so that electrotechnicians do not have to learn new programming languages to be able to implement a control program.
The structure of the actual Ladder Diagram enables the connection of electric elements in such a way that generates a control output, which is dependent upon a logical power flow through used electrical objects, which displays the previously requested condition of a physical electrical device.
In simple form, the user interface is a video display processed by the PLC programming application, which sets up a vertical and horizontal grid in which programming objects are classified. The diagram contains the power grid on the left side, and when connected to activated objects, the power shifts from left to right.

Landscape

Landscape means that when looking at the printed text, the page is wider than it is high.

Language Element

Every basic element in one of the IEC programming languages, e.g. a step in SFC, a function block instance in FBD or the initial value of a variable.

Library

Collection of software objects, which are intended for re-use when programming new projects, or even building new libraries. Examples are the libraries of the Elementary function block types.
EFB libraries can be divided up into Groups.

Link

A control or data flow connection between graphical objects (e.g. steps in the SFC Editor, function blocks in the FBD Editor) within a section, represented graphically as a line.

Literals

Literals are used to provide FFB inputs, and transition conditions etc with direct values. These values can not be overwritten by the program logic (write-protected). A distinction is made between generic and standardized literals.
Literals are also used to allocate, to a constant, a value or a variable, an initial value. Entries are made as base 2 literal, base 8 literal, base 16 literal, integer literal, real literal or real literal with exponent.

Local derived data types

Local derived data types are only available in a single Concept project and the local DFBs and are placed in the DFB directory under the project directory.

Local DFBs

Local DFBs are only available in a single Concept project and are placed in the DFB directory under the project directory.

Local Link

The local network is the network, which connects the local nodes with other nodes either directly or through bus repeaters.

Local macros

Local macros are only available in a single Concept project and are placed in the DFB directory under the project directory.

Local network nodes

The local node is the one which is currently being configured.

Located variable

A state RAM address (reference addresses 0x, 1x, 3x,4x) is allocated to located variables. The value of these variables is saved in the state RAM and can be modified online using the reference data editor. These variables can be addressed using their symbolic names or their reference addresses.

All inputs and outputs of the PLC are connected to the state RAM. The program can only access peripheral signals attached to the PLC via located variables. External access via Modbus or Modbus Plus interfaces of the PLC, e.g. from visualization systems, is also possible via located variables.

M

Macro

Macros are created with the help of the Concept DFB software.
Macros are used to duplicate frequently used sections and networks (including their logic, variables and variable declaration).
A distinction is made between local and global macros.

Macros have the following properties:
- Macros can only be created in the FBD and LD programming languages.
- Macros only contain one section.
- Macros can contain a section of any complexity.
- In programming terms, there is no difference between an instanced macro, i.e. a macro inserted into a section and a conventionally created section.
- DFB invocation in a macro
- Declaring variables
- Using macro-specific data structures
- Automatic transfer of the variables declared in the macro.
- Initial values for variables
- Multiple instancing of a macro in the entire program with differing variables
- The name of the section, variable names and data structure names can contain up to 10 different exchange marks (@0 to @9).

MMI

Man-Machine-Interface

Multi element variables

Variables to which a Derived data type defined with STRUCT or ARRAY is allocated. A distinction is made here between field variables and structured variables.

N

Network

A network is the collective switching of devices to a common data path, which then communicate with each other using a common protocol.

Network node

A node is a device with an address (1...64) on the Modbus Plus network.

Node

Node is a programming cell in a LL984 network. A cell/node consists of a 7x11 matrix, i.e. 7 rows of 11 elements.

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

O

Operand An operand is a literal, a variable, a function invocation or an expression.

Operator An operator is a symbol for an arithmetic or boolean operation which is to be executed.

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

Output/Marker bits (0x references) An output/marker bit can be used to control real output data using an output unit of the control system, or to define one or more discrete outputs in the state RAM. Note: The x, which follows the initial reference type number, represents a five-figure storage location in the user data memory, i.e. the reference 000201 signifies an output or marker bit at the address 201 in the State RAM.

Output/marker words (4x references) An output / marker word can be used to save numerical data (binary or decimal) in the state RAM, or to send data from the CPU to an output unit in the control system. Note: The x, which follows the initial reference type number, represents a five-figure storage location in the user data memory, i.e. the reference 400201 signifies a 16 bit output or marker word at the address 201 in the State RAM.

P

Peer CPU The Peer CPU processes the token execution and the data flow between the Modbus Plus network and the PLC user logic.

PLC Memory programmable controller

Portrait Portrait means that the sides are larger than the width when printed.

Program The uppermost program organization unit. A program is closed on a single PLC download.

Program organization unit
A function, a function block, or a Program. This term can refer to either a type or an instance.

Program redundancy system (Hot Standby)
A redundancy system consists of two identically configured PLC machines, which communicate with one another via redundancy processors. In the case of a breakdown of the primary PLC, the secondary PLC takes over the control check. Under normal conditions, the secondary PLC does not take over the control function, but checks the status information, in order to detect errors.

Project
General description for the highest level of a software tree structure, which specifies the super-ordinate project name of a PLC application. After specifying the project name you can save your system configuration and your control program under this name. All data that is created whilst setting up the configuration and program, belongs to this super-ordinate project for this specific automation task.
General description for the complete set of programming and configuration information in the project database, which represents the source code that describes the automation of a system.

Project database
The database in the host computer, which contains the configuration information for a project.

Prototype file (Concept-EFB)
The prototype file contains all the prototypes of the assigned functions. In addition, if one exists, a type definition of the internal status structure is specified.

R

REAL
REAL stands for the data type "floating point number". The entry can be real-literal or real-literal with an exponent. The length of the data element is 32 bits. The value range for variables of this data type extends from +/-3.402823E+38.

> **Note:** Dependent on the mathematical processor type of the CPU, different ranges within this permissible value range cannot be represented. This applies to values that are approaching ZERO and for values that approach INFINITY. In these cases NAN (**N**ot **A** **N**umber) or INF (**INF**inite) will be displayed in the animation mode instead of a number value.

Real literals
Real literals are used to input floating point values into the decimal system. Real literals are denoted by a decimal point. The values can have a preceding sign (+/-). Single underscores (_) between numbers are not significant.

Example
-12.0, 0.0, +0.456, 3.14159_26

Real literals with exponents

Real literals with exponents are used to input floating point values into the decimal system. Real literals with exponents are identifiable by a decimal point. The exponent indicates the power of ten, with which the existing number needs to be multiplied in order to obtain the value to be represented. The base can have a preceding negative sign (-). The exponent can have a preceding positive or negative sign (+/-). Single underscores (_) between numbers are not significant. (Only between characters, not before or after the decimal point and not before or after "E", "E+" or "E-")

Example
-1.34E-12 or -1.34e-12
1.0E+6 or 1.0e+6
1.234E6 or 1.234e6

Reference

Every direct address is a reference that begins with an indicator, which specifies whether it is an input or an output and whether it is a bit or a word. References that begin with the code 6, represent registers in the extended memory of the state RAM.
0x range = Output/Marker bits
1x range = Input bits
3x range = Input words
4x range = Output registers
6x range = Register in the extended memory

Note: The x, which follows each initial reference type number, represents a five-digit storage location in the user data memory, i.e. the reference 400201 signifies a 16 bit output or marker word at the address 201 in the State RAM.

Register in the extended memory (6x-reference)

6x references are holding registers in the extended memory of the PLC. They can only be used with LL984 user programs and only with a CPU 213 04 or CPU 424 02.

Remote Network (DIO)

Remote programming in the Modbus Plus network enables maximum performance when transferring data and dispenses with the need for connections. Programming a remote network is simple. Setting up a network does not require any additional ladder logic to be created. All requirements for data transfer are fulfilled via corresponding entries in the Peer Cop Processor.

RIO (Remote I/O)

Remote I/O indicates a physical location of the I/O point controlling devices with regard to the CPU controlling them. Remote inp./outputs are connected to the controlling device via a twisted communication cable.

RTU-Mode

Remote Terminal Unit
The RTU mode is used for communication between the PLC and an IBM compatible personal computer. RTU works with 8 data bits.

Runtime error Errors, which appear during program processing on the PLC, in SFC objects (e.g. Steps) or FFBs. These are, for example, value range overflows for numbers or timing errors for steps.

S

SA85 module The SA85 module is a Modbus Plus adapter for IBM-AT or compatible computers.

Scan A scan consists of reading the inputs, processing the program logic and outputting the outputs.

Section A section can for example be used to describe the functioning mode of a technological unit such as a motor.
A program or DFB consists of one or more sections. Sections can be programmed with the IEC programming languages FBD and SFC. Only one of the named programming languages may be used within a section at any one time.
Each section has its own document window in Concept. For reasons of clarity, however, it is useful to divide a very large section into several small ones. The scroll bar is used for scrolling within a section.

Section Code Section Code is the executable code of a section. The size of the Section Code is mainly dependent upon the number of blocks in the section.

Section Data Section data is the local data in a section such as e.g. literals, connections between blocks, non-connected block inputs and outputs, internal status memory of EFBs.

Note: Data which appears in the DFBs of this section is not section data.

Separator Format (4:00001) The first digit (the reference) is separated from the five-digit address that follows by a colon (:).

Sequence language (SFC) The SFC Language Elements enable a PLC program organization unit to be divided up into a number of Steps and Transitions, which are connected using directional Links. A number of actions belong to each step, and transition conditions are attached to each transition.

Serial Connections With serial connections (COM) the information is transferred bit by bit.

Source code file (Concept-EFB)

The source code file is a normal C++ source file. After executing the **Library** → **Create files** menu command, this file contains an EFB-code frame, in which you have to enter a specific code for the EFB selected. To do this invoke the **Objects** → **Source** menu command.

Standard Format (400001)

The five-digit address comes directly after the first digit (the reference).

Standardized literals

If you would like to manually determine a literal's data type, this may be done using the following construction: 'Data type name'#'value of the literal'.

Example
INT#15 (Data type: integer, value: 15),
BYTE#00001111 (Data type: byte, value: 00001111)
REAL#23.0 (Data type: real, value: 23.0)

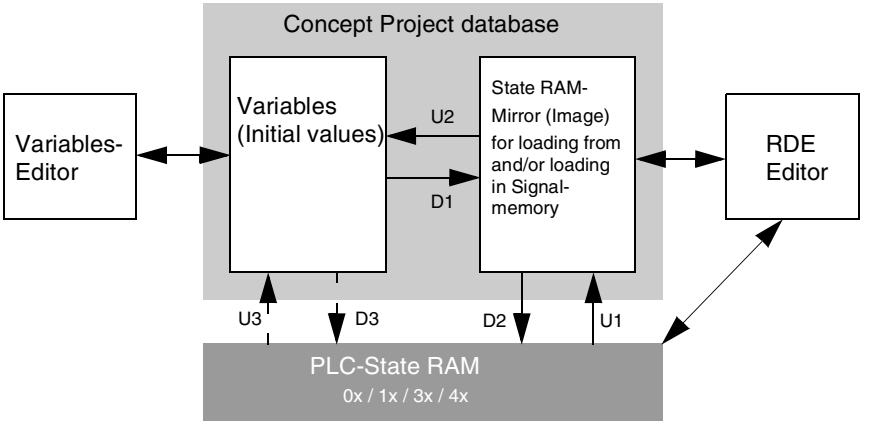
To assign the data type REAL, the value may also be specified in the following manner: 23.0.
Entering a comma will automatically assign the data type REAL.

State RAM

The state RAM is the memory space for all variables, which are accessed via References (Direct representation) in the user program. For example, discrete inputs, coils, input registers, and output registers are located in the state RAM.

State RAM overview for uploading and downloading

Overview:



Status Bits

For every device with global inputs or specific inputs/outputs of Peer Cop data, there is a status bit. If a defined group of data has been successfully transferred within the timeout that has been set, the corresponding status bit is set to 1. If this is not the case, this bit is set to 0 and all the data belonging to this group is deleted (to 0).

Step SFC-language element: Situation, in which the behavior of a program, in reference to its inputs and outputs, follows those operations which are defined by the actions belonging to the step.

Step name The step name is used to uniquely denote a step in a program organization unit. The step name is generated automatically, but it can be edited. The step name must be unique within the entire program organization unit, otherwise an error message will appear.
The automatically generated step name is always formed as follows: S_n_m

S = step
n = Number of the section (consecutive numbers)
m = Number of the step in the section (current number)

Structured text (ST) ST is a text language according to IEC 1131, in which operations, e.g. invocations of Function blocks and Functions, conditional execution of instructions, repetitions of instructions etc. are represented by instructions.

Structured variables Variables to which a Derived data type defined with STRUCT (structure) is allocated. A structure is a collection of data elements with generally different data types (elementary data types and/or derived data types).

SY/MAX In Quantum control devices, Concept includes the preparation of I/O-map SY/MAX-I/O modules for remote controlling by the Quantum PLC. The SY/MAX remote backplane has a remote I/O adapter in slot 1, which communicates via a Modicon S908 R I/O System. The SY/MAX-I/O modules are executed for you for labeling and inclusion in the I/O map of the Concept configuration.

T

Template file (Concept-EFB) The template file is an ASCII file with layout information for the Concept FBD Editor, and the parameters for code creation.

TIME TIME stands for the data type "time". The entry is time literal. The length of the data element is 32 bits. The value range for variables of this data type extends from 0 to 2exp(32)-1. The unit for the data type TIME is 1 ms.

Time literals

Permissible units for times (TIME) are days (D), hours (H), minutes (M), seconds (S) and milliseconds (MS) or combinations of these. The time must be marked with the prefix t#, T#, time# or TIME#. The "overflow" of the unit with the highest value is permissible, e.g. the entry T#25H15M is allowed.

Example
t#14MS, T#14.7S, time#18M, TIME#19.9H, t#20.4D, T#25H15M, time#5D14H12M18S3.5MS

Token

The network "token" controls the temporary possession of the transfer right via a single node. The token passes round the nodes in a rotating (increasing) address sequence. All nodes follow the token rotation and can receive all the possible data that is sent with it.

Total IEC memory

The total IEC memory consists of the IEC program memory and the global data.

Traffic Cop

The traffic cop is an IO map, which is generated from the user-IO map. The traffic cop is managed in the PLC and in addition to the user IO map, contains e.g. status information on the I/O stations and modules.

Transition

The condition, in which the control of one or more predecessor steps passes to one or more successor steps along a directed link.

U

UDEFB

User-defined elementary functions/function blocks
Functions or function blocks, which were created in the C programming language, and which Concept provides in libraries.

UDINT

UDINT stands for the data type "unsigned double integer". Entries are made as integer literal, base 2 literal, base 8 literal or base 16 literal. The length of the data element is 32 bits. The value range for variables of this data type extends from 0 to 2exp(32)-1.

UINT

UINT stands for the data type "unsigned integer". Entries are made as integer literal, base 2 literal, base 8 literal or base 16 literal. The length of the data element is 16 bits. The value range for variables of this data type extends from 0 to (2exp 16)-1.

Unlocated variable

Unlocated variables are not allocated a state RAM address. They therefore do not occupy any state RAM addresses. The value of these variables is saved in the internal system and can be changed using the reference data editor. These variables are only addressed using their symbolic names.

Signals requiring no peripheral access, e.g. intermediate results, system tags etc., should be primarily declared as unlocated variables.

V

Variables

Variables are used to exchange data within a section, between several sections and between the program and the PLC.
Variables consist of at least one variable name and one data type.
If a variable is assigned a direct address (reference), it is called a located variable.
If the variable has no direct address assigned to it, it is called an unlocated variable.
If the variable is assigned with a derived data type, it is called a multi element variable.
There are also constants and literals.

W

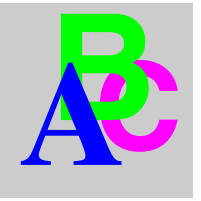
Warning

If a critical status is detected during the processing of a FFB or a step (e.g. critical input values or an exceeded time limit), a warning appears, which can be seen using the **Online** → **Event Viewer...**menu command. For FFBs, the ENO remains set to "1".

WORD

WORD stands for the data type "bit sequence 16". Entries are made as base 2 literal, base 8 literal or base 16 literal. The length of the data element is 16 bits. A numerical value range can not be assigned to this data type.

Index



D

E

F

G

I

L

M

N

O

P

R

S

T

U

W

X