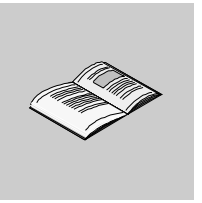
Concept 2.6
IEC Block Library
Part: EXTENDED

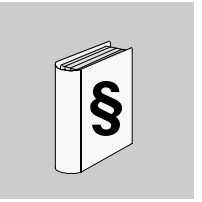
01/2007

33002214.04

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

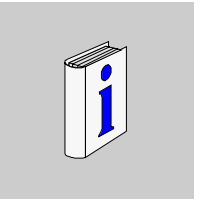
WARNING indicates a potentially hazardous situation, which, if not avoided, **can result** in death, serious injury, or equipment damage.

⚠ 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 on the block library EXTENDED

I

Overview

Introduction

This section contains general information on the block library EXTENDED.

What's in this Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
1	Parameterizing functions and function blocks	15

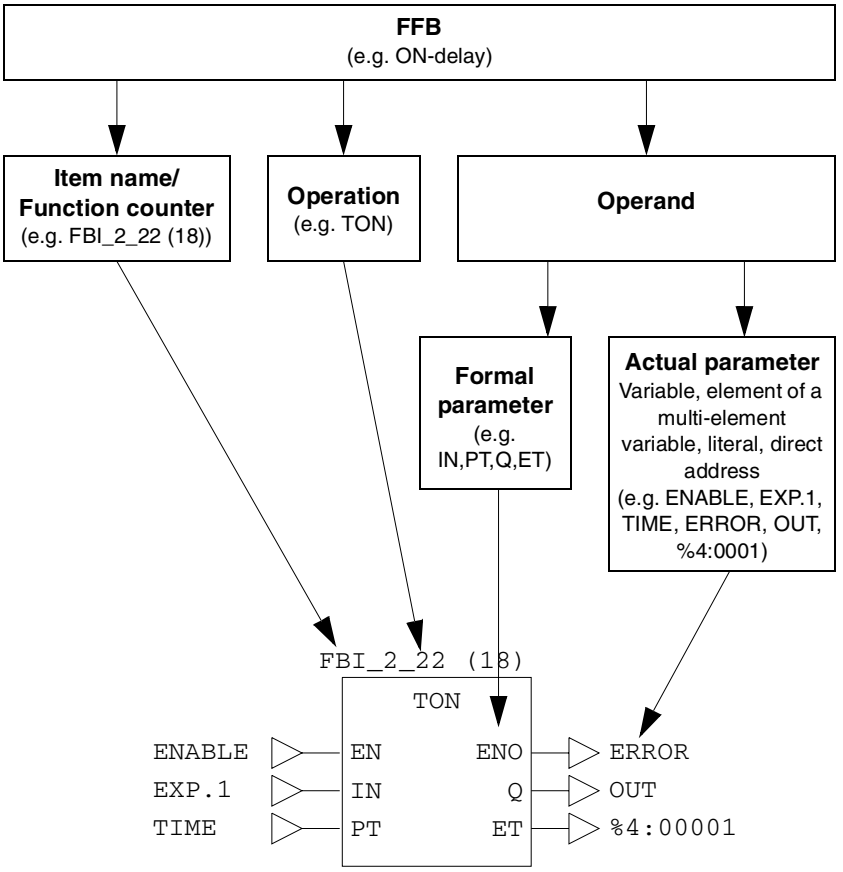
Parameterizing functions and function blocks

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 the EFBs can be increased to a maximum of 32 by changing the size of the FFB symbols vertically. Please consult the individual EFB descriptions to know which EFBs are concerned.

What's in this Part?

This part contains the following chapters:

AVE_***: Averaging

2

Overview

Introduction

This chapter describes the AVE_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function calculates the mean of weighted input values, and the result is then given at the output.

Each two successive inputs (K_Xn) represent one pair of values. The first K_Xn input corresponds to K1, the next to X1, the one after that to K2, etc.

The number of K_Xn inputs can be increased to 32 by vertically modifying the size of the block frame. This corresponds to a maximum of 16 value pairs.

The number of inputs must be even.

Data types of the ANY_NUM group can be processed.

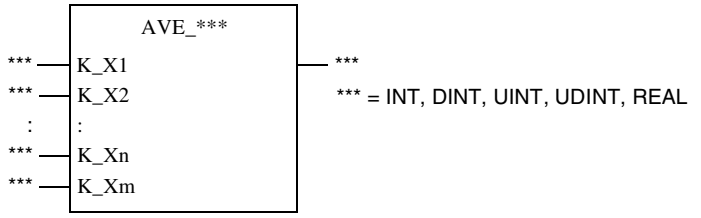
The data types of all input and output values must be identical. A specific function is available to process each of the different data types.

It is possible to project EN and ENO as additional parameters.

Representation

Symbol

Block representation:



Formula

Block formula:

$$OUT = \frac{\Sigma(Ki \times Xi)}{\Sigma(Ki)}$$

Parameter description

Block parameter description:

Parameters	Data type	Meaning
K_X1	INT, DINT, UINT, UDINT, REAL	Factor (K1) for first value
K_X2	INT, DINT, UINT, UDINT, REAL	First value (X1)
K_X3	INT, DINT, UINT, UDINT, REAL	Factor (K2) for second value
K_X4	INT, DINT, UINT, UDINT, REAL	Second value (X2)
:		
K_Xn	INT, DINT, UINT, UDINT, REAL	Factor ($K_{m/2}$) for m/2 value
K_Xm	INT, DINT, UINT, UDINT, REAL	m/2 value ($X_{m/2}$)
OUT	INT, DINT, UINT, UDINT, REAL	Mean value

AVGMV: Floating mean with fixed window size

3

Overview

Introduction

This chapter describes the AVGMV block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block forms a floating mean from a fixed number of input values (Input X). The output is the mean of all values between the current X value and the oldest X-value (N-1). It is possible to save up to 100 input values (N).

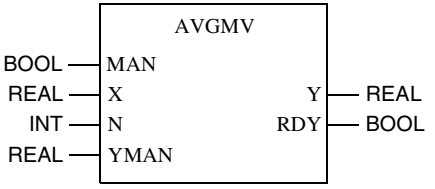
The function block can operate in manual and automatic mode.

It is possible to project EN and ENO as additional parameters.

Representation

Symbol

Block representation:



Formula

With RDY = 1:

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

or

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

Explanation of variables

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

Parameter description

Block parameter description:

Parameter	Data type	Meaning
MAN	BOOL	"0" = automatic operating mode "1" = manual operating mode
X	REAL	Input
N	INT	Window size (number of input values that are loaded into the buffer; 100 max.)
YMAN	REAL	Manual value
Y	REAL	Mean value
RDY	BOOL	"1" = n values in buffer, i.e. buffer ready "0" = buffer not ready

Detailed description

Automatic operating mode

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

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

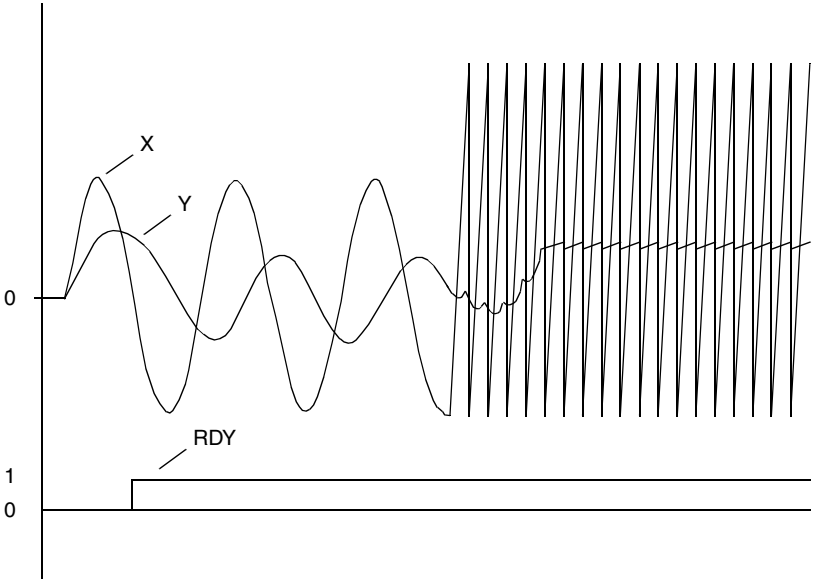
After a modification of the N value or after a cold/warm start, the internal buffer is deleted. The output is set to the input value X and RDY to "0". The buffer is filled during the next N cycles. The Y output contains an mean of the values accumulated so far. RDY remains "0" until the buffer is filled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

Manual operating mode

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

Diagram

Floating mean with limited memory of N = 50 values



Runtime error

Runtime error

An Error message appears if,

- N = 0 or N>100

AVGMV_K: Floating mean with frozen correction factor

4

Overview

Introduction

This chapter describes the AVGMV_K block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block establishes a floating mean (with frozen correction factor) of up to 10 000 input values.

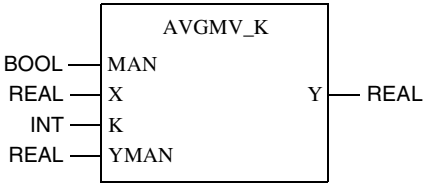
The function block can operate in manual and automatic mode.

It is possible to project EN and ENO as additional parameters.

Representation

Symbol

Block representation:



Formula

Block formula:

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

Explanation of variables

Variable	Meaning
$Y_{(new)}$	Y value in current program cycle
$Y_{(old)}$	Y value from last program cycle
K	Correction factor
X	X value in current program cycle

Parameter description

Block parameter description:

Parameter	Data type	Meaning
MAN	BOOL	"0" = Automatic Mode; "1" = Manual Mode
X	REAL	Input
K	INT	Correction factor (max. 10 000)
YMAN	REAL	Manual value
Y	REAL	Mean value

Detailed description

Automatic operating mode

One X value is read in every program cycle. 1/N is deducted from the Y value of the last program cycle, and then 1/N of the current X value is added. The result is delivered at the Y output.

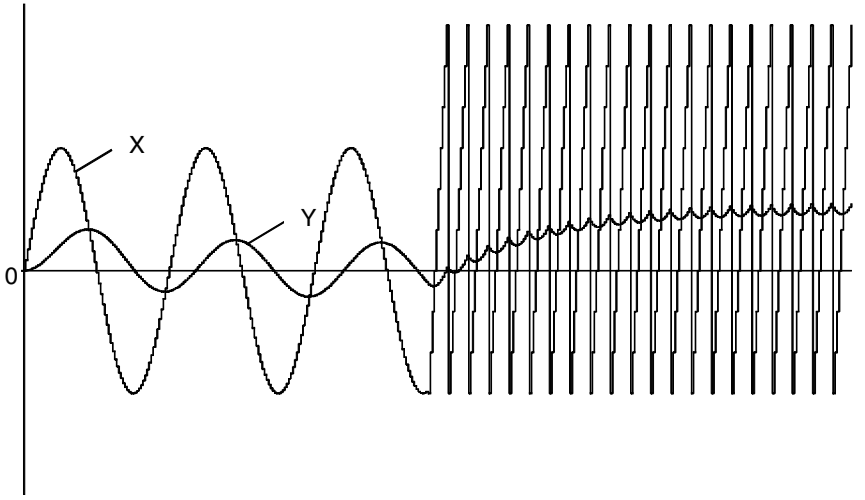
After a cold or warm (re)start, the X value is assigned to output Y.

Manual operating mode

The value YMAN is transferred to the Y output.

Diagram

Floating mean with frozen correction factor (K = 50)



BCD_TO_INT: Conversion from 16 Bit BCD to INT

5

Overview

Introduction

This chapter describes the BCD_TO_INT block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function converts a 16 Bit BCD input value (8-4-2-1-Code) into an INT data type output value.

If no valid BCD coded value is created at input, a runtime error is reported and the input value passes unchanged to the output.

EN and ENO can be configured as additional parameters.

Example

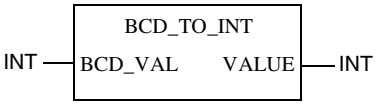
Example of a BCD -> INT conversion

	Data type	Dec. value	Hex. value (= BCD-value)
Input value	INT	-26 797	9753
Output value	INT	9 753	-

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
BCD_VAL	INT	16 Bit BCD input value
VALUE	INT	INT output value

BIT_TO_BYTE: Type conversion

6

Overview

Introduction

This chapter describes the BIT_TO_BYTE block.

What's in this Chapter?

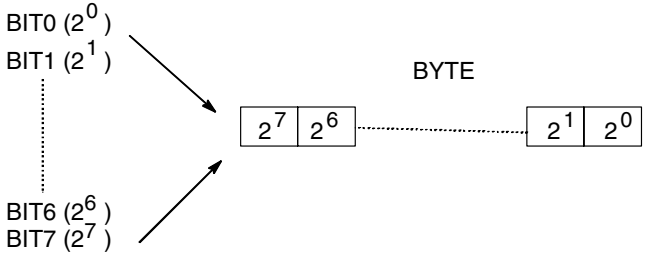
This chapter contains the following topics:

Brief description

Function description

The Function converts 8 input values of the Data type BOOL to an output of the data type BYTE.

The input values are assigned to the individual bits of the byte at the output according to the input names.

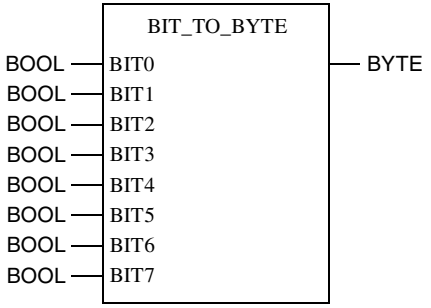


It is possible to project EN and ENO as additional parameters.

Representation

Symbol

Block representation:



Formula

Block formula:

OUT = {BIT7,BIT6,...,BIT0}

Parameter description

Block parameter description:

Parameter	Data type	Meaning
BIT0	BOOL	Input bit 0
BIT1	BOOL	Input bit 1
:	:	:
BIT7	BOOL	Input bit 7
OUT	BYTE	Output value

BIT_TO_WORD: Type conversion

7

Overview

Introduction

This chapter describes the BIT_TO_WORD block.

What's in this Chapter?

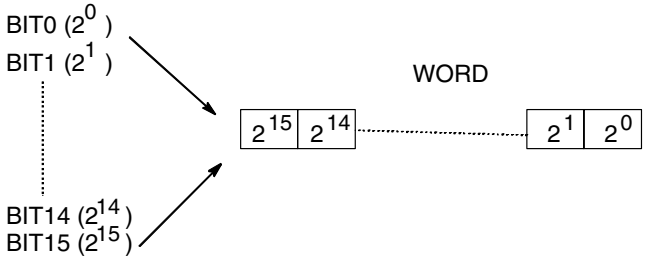
This chapter contains the following topics:

Brief description

Function description

The Function converts 16 input words from Data type BOOL to an output value of data type WORD.

The input values are assigned to the individual bits of the word at the output according to the input names.

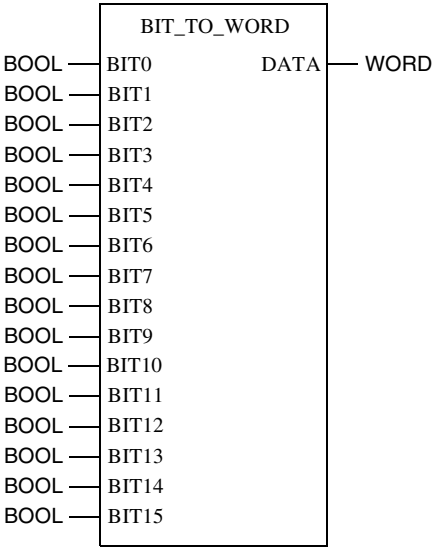


It is possible to project EN and ENO as additional parameters.

Representation

Symbol

Block representation:



Formula

Block formula:

$$OUT = \{BIT15,BIT14,\ldots,BIT0\}$$

Parameter description

Block parameter description:

Parameter	Data type	Meaning
BIT0	BOOL	Input bit 0
BIT1	BOOL	Input bit 1
:	:	:
BIT15	BOOL	Input bit 15
OUT	WORD	Output value

BYTE_AS_WORD: Type conversion

8

Overview

Introduction

This chapter describes the BYTE_AS_WORD block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function converts 2 input words from Data type BYTE to an output value of data type WORD.

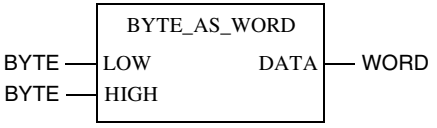
The input values are assigned to the word at the output according to the input names.

It is possible to project EN and ENO as additional parameters.

Representation

Symbol

Block representation:



Formula

Block formula:

$$OUT = \{HIGH, LOW\}$$

Parameter description

Block parameter description:

Parameter	Data type	Meaning
LOW	BYTE	less significant byte
HIGH	BYTE	more significant byte
OUT	WORD	Output value

BYTE_TO_BIT: Type conversion

9

Overview

Introduction

This chapter describes the BYTE_TO_BIT block.

What's in this Chapter?

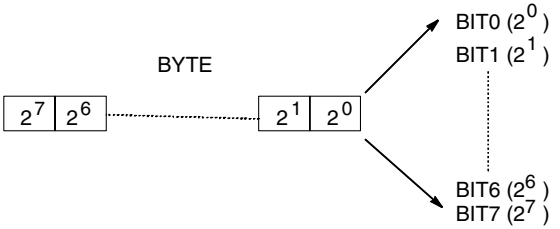
This chapter contains the following topics:

Brief description

Function description

Thisfunction block converts one input word from Data type BYTE into 8 output values of data type BOOL.

The individual bits of the byte at the input are assigned to the outputs according to the output names.

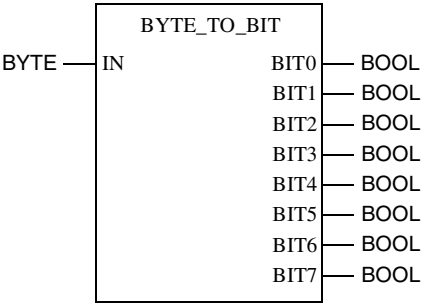


It is possible to project EN and ENO as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
IN	BYTE	Input
BIT0	BOOL	Output bit 0
BIT1	BOOL	Output bit 1
:	:	:
BIT7	BOOL	Output bit 7

CTD_***: Down counter

10

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 for counting down.

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

When CV is ≤ 0, the output Q becomes "1".

The Data types of the PV input and the CV output must be identical. A specific function block is available to process each of the different data types.

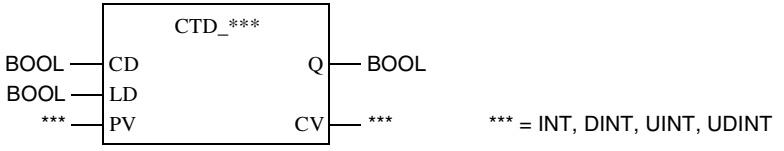
Note: The counter only works up to the minimum values of the data type being used. No overflow occurs.

The parameters EN and ENO can additionally be projected.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
CD	BOOL	Trigger input
LD	BOOL	Load data
PV	INT, DINT, UINT, UDINT	Presettings value
Q	BOOL	Output
CV	INT, DINT, UINT, UDINT	Count value (actual value)

CTU_***: Up counter

11

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 for counting up.

A "1" signal at the R input causes the value "0" to be allocated 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 ≥ PV, the Q output is set to "1".

The Data types of the PV input and the CV output must be identical. A specific function block is available to process each of the different data types.

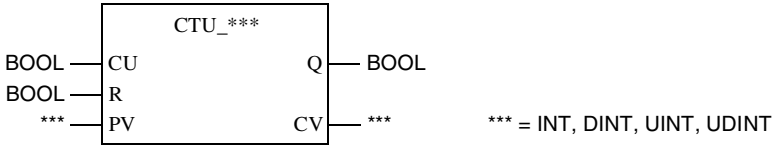
Note: The counter only works up to the maximum values of the data type being used. No overflow occurs.

The parameters EN and ENO can additionally be projected.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
CU	BOOL	Trigger input
R	BOOL	Reset
PV	INT, DINT, UINT, UDINT	Presettings value
Q	BOOL	Output
CV	INT, DINT, UINT, UDINT	Count value (actual value)

CTUD_***: Up/Down counter

12

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 for counting up and down.

A "1" signal at the R input causes the value "0" to be allocated to the CV output. A "1" signal at the LD input causes the value of the PV input to be allocated 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 reduced by 1.

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

When CV ≥ PV, output QU is "1".

When CV is ≤ 0, output QD is "1".

Note: The down counter only works up to the minimum values of the data type being used, and the up counter only up to the maximum values of the data type being used. No overflow occurs.

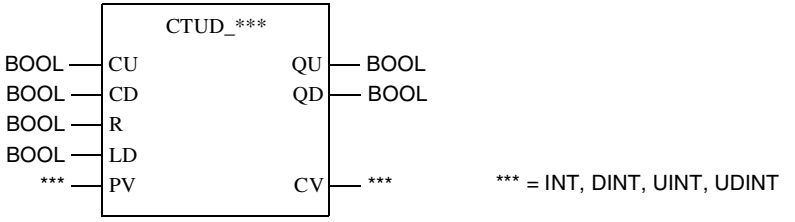
The Data types of input PV and input CV must be identical. A specific Function block is available to process each of the different data types.

The parameters EN and ENO can additionally be projected.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

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, DINT, UINT, UDINT	Presettings value
QU	BOOL	Up display
QD	BOOL	Down display
CV	INT, DINT, UINT, UDINT	Count value (actual value)

DBCD_TO_DINT: Conversion from 32 Bit BCD to DINT

13

Overview

Introduction

This chapter describes the DBCD_TO_DINT block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function converts a 32 Bit BCD input value (8-4-2-1-Code) into a DINT data type output value.

If no valid BCD coded value is created at input, a runtime error is reported and the input value passes unchanged to the output.

EN and ENO can be configured as additional parameters.

Example

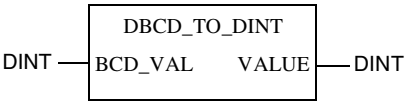
Example of a DBCD -> DINT conversion

	Data type	Dec. value	Hex. value (= BCD-value)
Input value	DINT	-2 023 406 815	8765 4321
Output value	DINT	87 654 321	-

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
BCD_VAL	DINT	32 Bit BCD input value
VALUE	DINT	DINT output value

DBCD_TO_INT: Conversion from 32 Bit BCD to INT

14

Overview

Introduction

This chapter describes the DBCD_TO_INT block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function converts a 32 Bit BCD input value (8-4-2-1-Code) into an INT data type output value.

The valid input value range is 0 32 767 (BCD.).

The following runtime errors are generated:

- If no valid BCD coded value is created at input, a runtime error is reported and the input value passes unchanged to the output.
- If the BCD format is correct, but too large (> 32 767) a runtime error is reported and the output value is set to -1.

EN and ENO can be configured as additional parameters.

Example

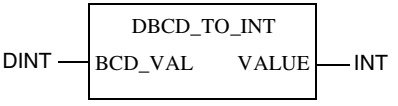
Example of a DBCD -> INT conversion

	Data type	Dec. value	Hex. value (= BCD-value)
Input value	DINT	22 083	0000 5643
Output value	INT	5 643	-

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
BCD_VAL	DINT	32 Bit BCD input value (valid value range 0 - 32767)
VALUE	INT	INT output value

DEAD_ZONE_REAL: Dead zone

15

Overview

Introduction

This chapter describes the DEAD_ZONE_REAL block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

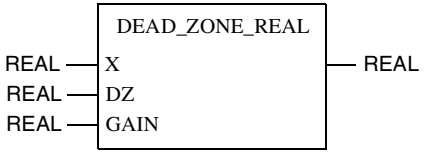
The Function is used to specify a deadzone for control variables.

It is possible to project EN and ENO as additional parameters.

Representation

Symbol

Block representation:



Formula

Block formula:

Assuming: DZ≥0

Y = GAIN x X for -DZ ≤ X ≤ DZ

Y = (X - DZ) + GAIN x DZ for X > DZ

Y = (X + DZ) - GAIN x DZ for X < DZ

Parameter description

Block parameter description:

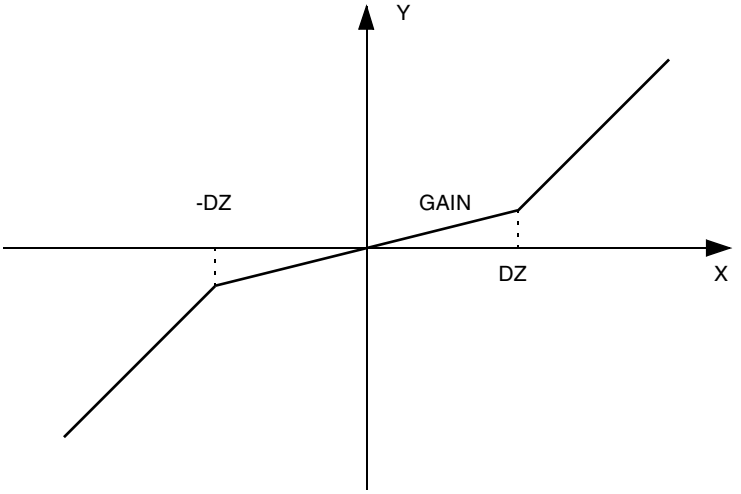
Parameter	Data type	Meaning
X	REAL	Input variable
DZ	REAL	Half width of the deadzone
GAIN	REAL	Gradient within deadzone
Y	REAL	Output variable

Detailed description

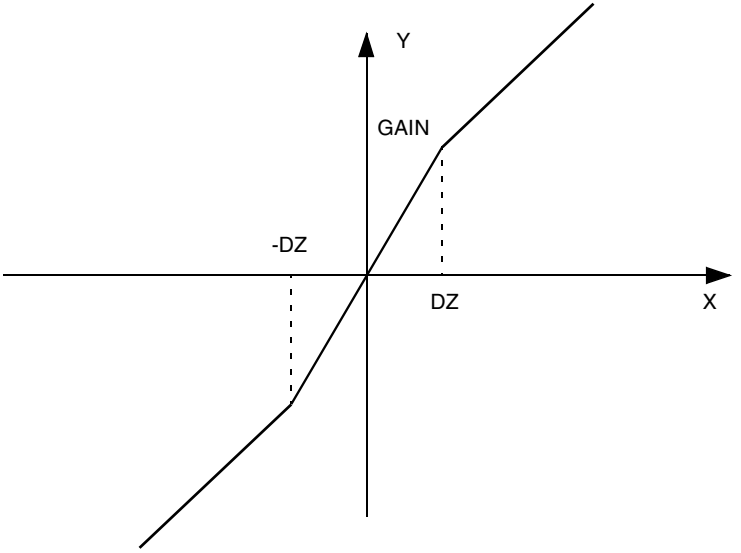
Characteristic Curves

The function block has the following characteristic curve:

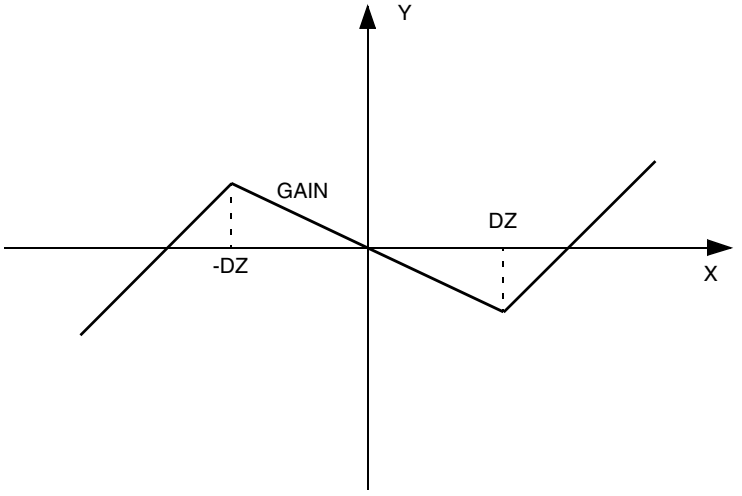
Deadzone with 0 < GAIN < 1



Deadzone with GAIN > 1



Deadzone with GAIN < 0



Note: Outside the dead zone, a gradient of 1 has been specified.

DINT_AS_WORD: Type conversion

16

Overview

Introduction

This chapter describes the DINT_AS_WORD block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function block converts one input value of Data type DINT to 2 output values of data type WORD.

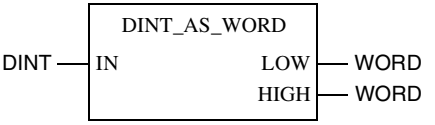
The individual words of the DINT input are assigned to the outputs corresponding to the output names.

It is possible to project EN and ENO as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
IN	DINT	Input
LOW	WORD	less significant word
HIGH	WORD	more significant word

DINT_TO_DBCD: Conversion from DINT to 32 Bit BCD

17

Overview

Introduction

This chapter describes the DINT_TO_DBCD block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function converts a DINT data type input value into a 32 Bit BCD output value (8-4-2-1-Code).

The valid input value range is 0 99 999 999 (Dec.).

If no valid value is created at input, a runtime error is reported and the input value passes unchanged to the output.

EN and ENO can be configured as additional parameters.

Example

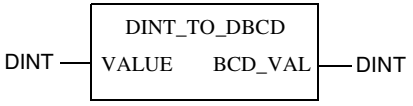
Example of a DINT -> DBCD conversion

	Data type	Dec. value	Hex. value (= BCD-value)
Input value	DINT	87 654 321	-
Output value	DINT	-2 023 406 815	8765 4321

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
VALUE	DINT	DINT input value (valid value range 0 - 99 999 999)
BCD_VAL	DINT	32 Bit BCD output value

DIVMOD_***: Division and Modulo

18

Overview

Introduction

This chapter describes the DIVMOD_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function block divides the value at input IN1 by the value at input IN2. The result of the division (quotient) is delivered at output DV. The remainder of the division (Modulo) is delivered at output MD.

If there is a decimal place in the division result, the division will truncate it.

Data types of the ANY_INT group can be processed.

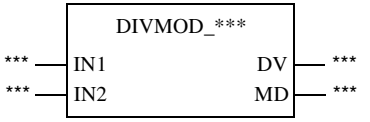
The data types of all input and output values must be identical. A specific function is available to process each of the different data types.

It is possible to project EN and ENO as additional parameters.

Representation

Symbol

Block representation:



*** = INT, DINT, UINT, UDINT

Formula

Block formula:

DV = IN1 / IN2

MD = IN1 mod IN2

Parameter description

Block parameter description:

Parameter	Data type	Meaning
IN1	INT, DINT, UINT, UDINT	Dividend
IN2	INT, DINT, UINT, UDINT	Divisor
DV	INT, DINT, UINT, UDINT	Quotient
MD	INT, DINT, UINT, UDINT	Modulo

Runtime error

Runtime error

An Error message appears, when

- IN2=0

HYST_***: Indicator signal for maximum value delimiter with hysteresis

19

Overview

Introduction

This chapter describes the HYST_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block monitors the input variable X for violation of the upper threshold.

Note: If the lower threshold should be monitored as well, use the INDLIM function block.

Data types of the ANY_NUM group can be processed.

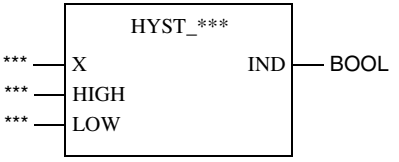
The data types of all input values must be identical. A specific function is available process each of the different data types.

It is possible to project EN and ENO as additional parameters.

Representation

Symbol

Block representation:



*** = INT, DINT, UINT, UDINT, REAL

Parameter description

Block parameter description:

Parameter	Data type	Meaning
X	INT, DINT, UINT, UDINT, REAL	Input variable
HIGH	INT, DINT, UINT, UDINT, REAL	Maximum upper threshold
LOW	INT, DINT, UINT, UDINT, REAL	Minimum upper threshold
IND	BOOL	Anzeige: reached upper threshold

Detailed Description

Parameter description

If X violates the upper HIGH limit, the function block reports this condition with IND = 1.

If, subsequently, X violates the LOW threshold, IND will be reset to "0".

Function

Function description

INDi = 1, if X > HIGH

INDi = 0, if X < LOW

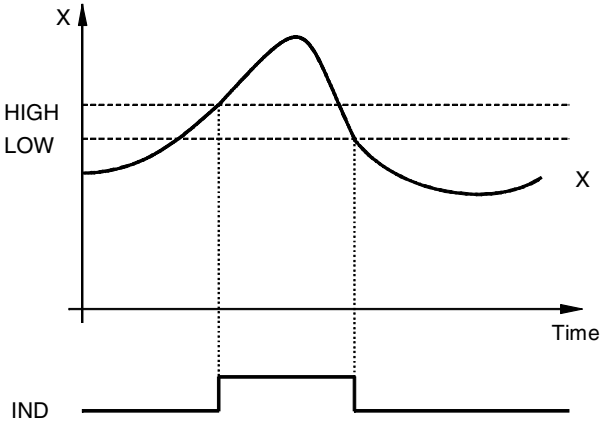
otherwise

INDi = INDi-1

If LOW is greater than HIGH, there is no hysteresis, and IND shows X is greater than HIGH.

Diagram

Maximum value delimiter with hysteresis



INDLIM_***: Indicator signal for delimiters with hysteresis

20

Overview

Introduction

This chapter describes the INDLIM_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block monitors the input variable X for violation of the upper threshold and violation of the lower threshold.

Note: If only the upper threshold should be monitored as well, use the HYST function block.

It is possible to process data types of the ANY_NUM group.

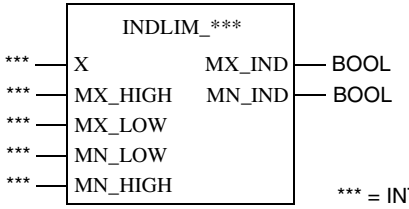
The data types of all input values must be identical. A specific function is available to process each of the different data types.

It is possible to project EN and ENO as additional parameters.

Representation

Symbol

Block representation:



*** = INT, DINT, UINT, UDINT, REAL

Parameter description

Block parameter description:

Parameter	Data type	Meaning
X	INT, DINT, UINT, UDINT, REAL	Input variable
MX_HIGH	INT, DINT, UINT, UDINT, REAL	Maximum upper limit
MX_LOW	INT, DINT, UINT, UDINT, REAL	Minimum lower limit
MN_LOW	INT, DINT, UINT, UDINT, REAL	Minimum lower limit
MN_HIGH	INT, DINT, UINT, UDINT, REAL	Minimum upper limit
MX_IND	BOOL	Display: reached upper limit
MN_IND	BOOL	Display: reached lower limit

Detailed description

Parameter description

If X exceeds the upper limit MX_HIGH, the function block reports this condition with MX_IND = 1.

If, subsequently, X is less than the limit MX_LOW, MX_IND will be reset to "0".

Function

Function description:

MX_INDi = 1, if X > MX_HIGH

MX_INDi = 0, if X < MX_LOW

otherwise

MX_INDi = MX_IND(i-1)

If X is less than the lower limit MN_HIGH, the function block reports this condition with MN_IND =1.

If, subsequently, X exceeds the limit MN_LOW, MX_IND is reset to "0".

Function

Function description:

MX_INDi = 1, if X < MX_HIGH

MX_INDi = 0, if X > MX_LOW

otherwise

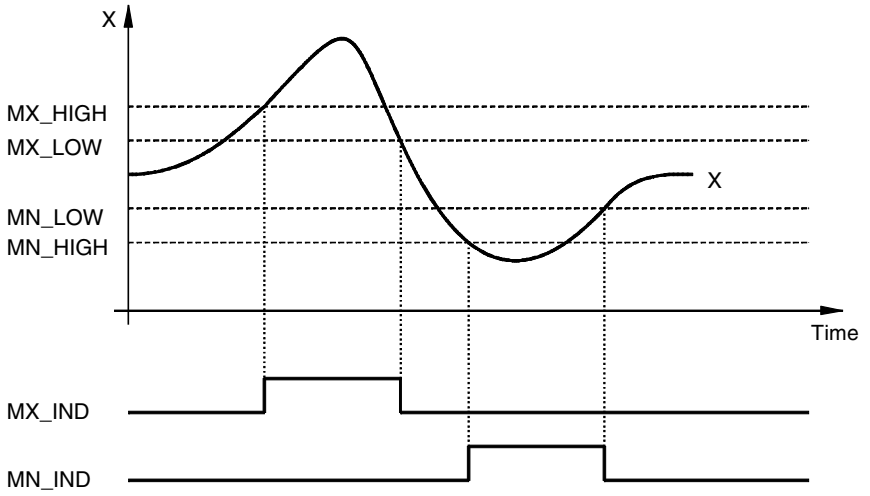
MX_INDi = MX_IND(i-1)

If MX_LOW is greater than MX_HIGH, there will be no hysteresis, and MX_IND displays the fact that X is greater than MX_HIGH.

If MN_HIGH is greater than MN_LOW, there will be no hysteresis, and MN_IND displays the fact that X is smaller than MN_HIGH.

Diagram

Delimiters with hysteresis INDLIM



INT_TO_BCD: Conversion from INT to 16 Bit BCD

21

Overview

Introduction

This chapter describes the INT_TO_BCD block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function converts an INT data type input value into a 16 Bit BCD output value (8-4-2-1-Code).

The valid input value range is 0 9 999 (Dec.).

If no valid BCD coded value is created at input, a runtime error is reported and the input value passes unchanged to the output.

EN and ENO can be configured as additional parameters.

Example

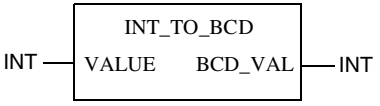
Example of an INT -> BCD conversion

	Data type	Dec. value	Hex. value (= BCD-value)
Input value	INT	9 753	-
Output value	INT	-26 797	9753

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
VALUE	INT	INT input value (valid value range 0 - 9 999)
BCD_VAL	INT	16 Bit BCD output value

INT_TO_DBCD: Conversion from INT to 32 Bit BCD

22

Overview

Introduction

This chapter describes the INT_TO_DBCD block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function converts an INT data type input value into a 32 Bit BCD output value (8-4-2-1-Code).

The valid input value range is 0 32 767 (Dec.).

If no valid value is created at input, a runtime error is reported and the input value passes unchanged to the output.

EN and ENO can be configured as additional parameters.

Example

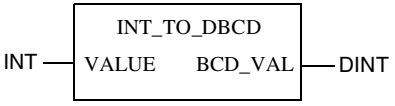
Example of an INT -> DBCD conversion

	Data type	Dec. value	Hex. value (= BCD-value)
Input value	INT	13 579	-
Output value	DINT	79 225	0001 3579

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
VALUE	INT	INT input value (valid value range 0 - 32767)
BCD_VAL	DINT	32 Bit BCD output value

LIMIT_IND_***: Limit with indicator

23

Overview

Introduction

This chapter describes the LIMIT_IND_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function block transfers the unchanged input value (IN) to the output OUT, if the input value is not less than the minimum value (MN) and does not exceed the maximum value (MX). If the input value (IN) is less than the minimum value (MN), the minimum value will be transferred to the output. If the input value (IN) exceeds the maximum value (MX), the maximum value is transferred to the output.

Furthermore, a signal indicates the violation of a minimum or maximum value. If the value at input IN is less than the value at input MN, output MN_IND becomes "1". If the value at input IN exceeds the value at input MX, the output MX_IND becomes "1".

Data types of the ANY_ELEM group can be processed.

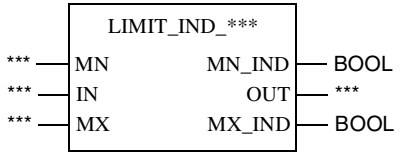
The data types of the input values MN, IN, MX as well as of the output value OUT must be identical. A specific function is available to process each of the different data types.

It is possible to project EN and ENO as additional parameters.

Representation

Symbol

Block representation:



*** = INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD

Formula

Block formula:

OUT = IN, if (IN ≤ MX) & IN ≥ MN

OUT = MN, if (IN < MN)

OUT = MX, if (IN > MX)

MN_IND = 0, if IN ≥ MN

MN_IND = 1, if IN < MN

MX_IND = 0, if IN ≤ MX

MX_IND = 1, if IN > MX

Parameter description

Block parameter description:

Parameter	Data type	Meaning
MN	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	Limit of minimum value
IN	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	Input
MX	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	Limit of maximum value
MN_IND	BOOL	Display of minimum value violation
OUT	INT, DINT, UINT, UDINT, REAL, TIME, BOOL, BYTE, WORD	Output
MX_IND	BOOL	Display of maximum value violation

LOOKUP_TABLE1: Traverse progression with 1st degree interpolation

24

Overview

Introduction

This chapter describes the LOOKUP_TABLE1 block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function block linearizes characteristic curves by means of interpolation. The function block works with variable width.

The number of XiYi inputs can be increased to 30 by modifying the size of the block frame vertically. This corresponds to a maximum of 15 support point pairs.

The number of inputs must be even.

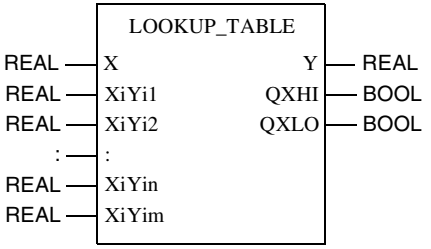
The X values must be in ascending order.

It is possible to project EN and ENO as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
XiYi1	REAL	X-coordinate 1st support point
XiYi2	REAL	Y-coordinate 1st support point
XiYin	REAL	X-coordinate mth support point
XiYim	REAL	Y-coordinate mth support point
X	REAL	Input variable
Y	REAL	Output variable
QXHI	BOOL	Display: X > Xm
QXLO	BOOL	Indicator signal X < X1

Detailed description

Parameter description

Each two sequential inputs (XiYi) represent a support point pair. The first input XiYi corresponds to X1, the next one to Y1, the one after that to X2, etc.

For all types of input value in X found between these support points, the corresponding Y output value is interpolated, while the traverse progression between the support points is viewed linearly.

For $X < X_1$ Y is = Y_1

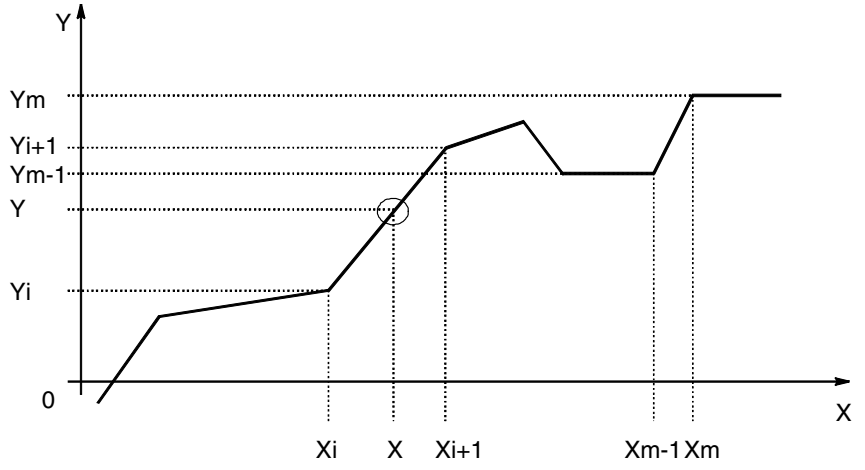
For $X > X_1$ is Y = Y_1

If the value at input X is higher than the value of the last support point Xm, the output QXHI becomes "1".

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

Principle of interpolation

Traverse progression with 1st degree interpolation



Interpolation

The following algorithm applies to a Point y:

$$Y = Y_i + \frac{Y_{(j+1)} - Y_j}{X_{(j+1)} - X_j} \times (X - X_i)$$

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

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

The X values must be in ascending order.

Two consecutive X values can be identical. This could cause a discrete curve progression.

In this instance, the special case applies:

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

for

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

NEG_***: Negation

25

Overview

Introduction

This chapter describes the NEG_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function negates the input value and delivers the result at the OUT output.

Data types of the ANY_NUM group can be processed.

The negation causes a sign reversal, e.g.

6 -> -6

-4 -> 4

Note: When the data types are processedDINT and INT it is not possible to convert very long negative values into positive ones. However, the ENO output is not set to 0 when this error occurs.

Note: When the data types are processedUDINT and UINT there is always an Error message.

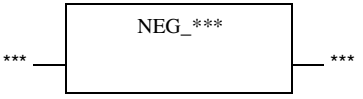
The data types of the input and output values must be identical. A specific function is available to process each of the different data types.

It is possible to project EN and ENO as additional parameters.

Representation

Symbol

Block representation:



*** = INT, DINT, UINT, UDINT, REAL

Parameter description

Block parameter description:

Parameter	Data type	Meaning
IN	INT, DINT, UINT, UDINT, REAL	Input
OUT	INT, DINT, UINT, UDINT, REAL	Negated output

Runtime error

Runtime error

A violation of the value range at the input during the execution of the function will cause an error message to appear.

An Error message appears, when

- the value range of the input is exceeded or
- an input value of the data type UDINT or UINT is to be converted.

REAL_AS_WORD: Type conversion

26

Overview

Introduction

This chapter describes the REAL_AS_WORD block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

This function block converts one input word from Data type REAL to 2 output values of data type WORD.

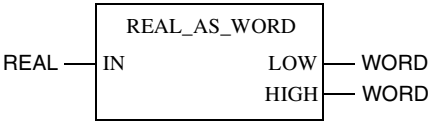
The individual words of the REAL input are assigned to the outputs according to the output names.

It is possible to project EN and ENO as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
IN	REAL	Input
LOW	WORD	less significant word
HIGH	WORD	more significant word

SAH: Detecting and holding with rising edge

27

Overview

Introduction

This chapter describes the SAH block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block transfers the input value PV to the OUT output when first called up. With a rising edge (0 to 1) at input CLK, the input value IN is transferred to the OUT output. This value remains at the output until the next rising edge causes a new value to be loaded from IN to OUT.

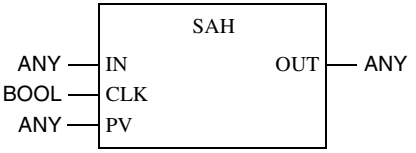
The data types of the input values IN, PV and of the output value OUT must be identical.

The parameters EN and ENO can additionally be projected.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
IN	ANY	Input value
CLK	BOOL	Clock input
PV	ANY	Preset value
OUT	ANY	Output value

SIGN_***: Sign evaluation

28

Overview

Introduction

This chapter describes the SIGN_*** block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function is used to detect negative signs.

Data types of the ANY_NUM group can be processed.

With a value ≥ 0 at the input, the output becomes "0". With a value < 0 at the input, the output becomes "1".

Note: Different processing of REAL and INT values results in the following behavior for signed 0 (+/-0):
-0.0 -> SIGN_REAL -> 1
-0 -> SIGN_INT/DINT -> 0
+0.0 -> SIGN_REAL -> 0
+0 -> SIGN_INT/DINT -> 0

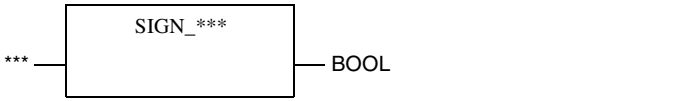
A specific function is available to process each of the different data types.

EN and ENO can be configured as additional parameters.

Representation

Symbol

Block representation:



*** = INT, DINT, UINT, UDINT, REAL

Formula

Block formula:

OUT = 1, if IN < 0

OUT = 0, if IN ≥ 0

Note: Different processing of REAL and INT values results in the following behavior for signed 0 (+/-0):
-0.0 -> SIGN_REAL -> 1
-0 -> SIGN_INT/DINT -> 0
+0.0 -> SIGN_REAL -> 0
+0 -> SIGN_INT/DINT -> 0

Parameter description

Block parameter description:

Parameter	Data type	Meaning
IN	INT, DINT, UINT, UDINT, REAL	Signed input
OUT	BOOL	Sign evaluation

TIME_AS_WORD: Type conversion

29

Overview

Introduction

This chapter describes the TIME_AS_WORD block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block converts one input word from Data type TIME to 2 output values of data type WORD.

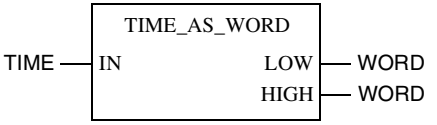
The individual words of the TIME input are assigned to the outputs according to the output names.

It is possible to project EN and ENO as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
IN	TIME	Input
LOW	WORD	less significant word
HIGH	WORD	more significant word

TOF_P: Off Delay with Pause

30

Overview

Introduction

This chapter describes the TOF_P block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block is used as 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".

With a "1" signal on input PAUSE, the timer is stopped and all values are frozen. If the input PAUSE becomes "0" again, the timer continues.

PAUSE has the highest priority. With a 1 -> 0 edge on input PAUSE, the block is no longer processed in this cycle. The block continues processing in the next cycle.

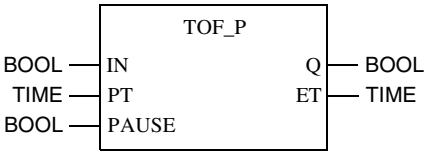
EN and ENO can be configured as additional parameters.

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.
However, if PAUSE is "1" when EN becomes "0", the pause is continued after EN becomes "1" until PAUSE becomes "0". In this case, there is no jump on output ET.

Representation

Symbol

Block representation:



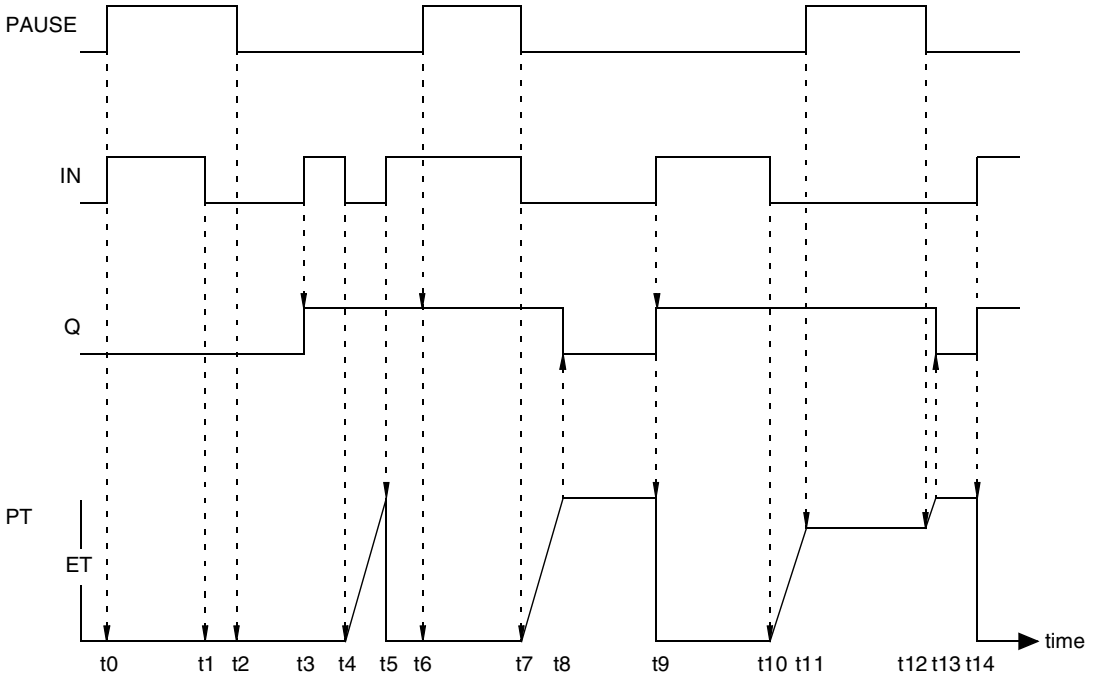
Parameter description

Block parameter description:

Parameter	Data type	Meaning
IN	BOOL	0 -> 1: Reset 1 -> 0: Start timer
PT	TIME	Preset delay time
PAUSE	BOOL	1: The timer values are frozen.
Q	BOOL	Output
ET	TIME	Elapsed time

Detailed description

Timing diagram Representation of the Off delay TOF_P:



t0 If IN becomes "1" and PAUSE is "1", Q remains "0" and the internal time is not started (PAUSE has priority above IN).
t1 If IN becomes "0" while PAUSE is "1", the block remains inactive.
t2 If PAUSE becomes "0" while IN is "0", the block remains inactive.
t3 If IN becomes "1", Q becomes "1".
t4 If IN becomes "0", the internal time (ET) is started.
t5 If IN becomes "1" before the internal time has reached the value of PT, the internal time is reset without Q becoming "0".
t6 If PAUSE becomes "1", Q remains "1" (the block is inactive).
t7 If IN and PAUSE become "0", the internal time is started in the next cycle.
t8 If the internal time reaches the value of PT, Q becomes "0".
t9 If IN becomes "1", Q becomes "1" and the internal time is reset.
t10 If IN becomes "0", the internal time is started.
t11 If PAUSE becomes "1" before the internal time has reached the value of PT, the internal time is stopped without Q becoming "0".
t12 If PAUSE becomes "0", the internal time (ET) continues.
t13 If the internal time reaches the value of PT, Q becomes "0".
t14 If IN becomes "1", Q becomes "1" and the internal time is reset.

TON_P: On Delay with Pause

31

Overview

Introduction

This chapter describes the TON_P block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block is used as 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".

With a "1" signal on input PAUSE, the timer is stopped and all values are frozen. If the input PAUSE becomes "0" again, the timer continues.

PAUSE has the highest priority. With a 1 -> 0 edge on input PAUSE, the block is no longer processed in this cycle. The block continues processing in the next cycle.

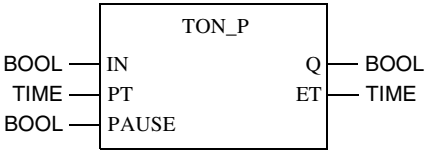
EN and ENO can be configured as additional parameters.

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.
However, if PAUSE is "1" when EN becomes "0", the pause is continued after EN becomes "1" until PAUSE becomes "0". In this case, there is no jump on output ET.

Representation

Symbol

Block representation:



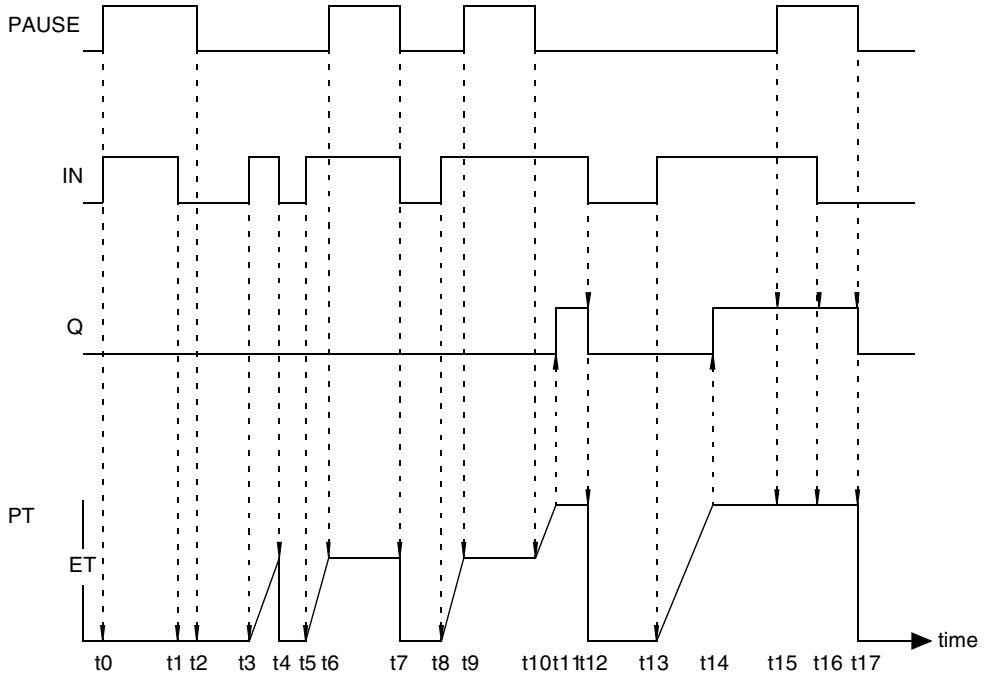
Parameter description

Block parameter description:

Parameter	Data type	Meaning
IN	BOOL	0 -> 1: Start timer 1 -> 0: Reset
PT	TIME	Preset delay time
PAUSE	BOOL	1: The timer values are frozen.
Q	BOOL	Output
ET	TIME	Elapsed time

Detailed description

Timing diagram Representation of the On delay TON_P:



t0 If IN becomes "1" and PAUSE is "1", Q remains "0" and the internal time is not started (PAUSE has priority above IN).
t1 If IN becomes "0" while PAUSE is "1", the block remains inactive.
t2 If PAUSE becomes "0" while IN is "0", the block remains inactive.
t3 If IN becomes "1", the internal time (ET) is started.
t4 If IN becomes "0" before the internal time has reached the value of PT, the internal time is reset without Q becoming "1".
t5 If IN becomes "1", the internal time (ET) is started.
t6 If PAUSE becomes "1" before the internal time has reached the value of PT, the internal time is stopped without Q becoming "1".
t7 If IN and PAUSE become "0", the internal time is reset in the next cycle without Q becoming "1".
t8 If IN becomes "1", the internal time (ET) is started.
t9 If PAUSE becomes "1" before the internal time has reached the value of PT, the internal time is stopped without Q becoming "1".
t10 If PAUSE becomes "0", the internal time (ET) continues.
t11 If the internal time reaches the value of PT, Q becomes "1".
t12 If IN becomes "0", Q becomes "0" and the internal time is reset.

t13 If IN becomes "1", the internal time (ET) is started.
t14 If the internal time reaches the value of PT, Q becomes "1".
t15 If PAUSE becomes "1", Q remains "1".
t16 If IN becomes "0" while PAUSE is "1", Q remains "1" and ET is not reset (the block is inactive).
t17 If PAUSE becomes "0" while IN is "0", Q becomes "0" in the next cycle and the internal time is reset.

TRIGGER: Detection of all types of edges

32

Overview

Introduction

This chapter describes the TRIGGER block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function block recognizes all types of edges (1 -> 0 and 0 -> 1) at input CLK.

Output EDGE becomes "1" if there is a transition from "0" to "1" or from "1" to "0" at CLK; otherwise it remains on "0".

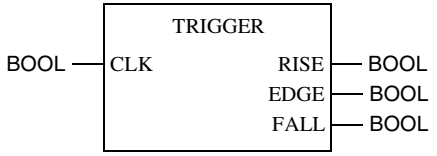
If there is a transition from "0" to "1" at input CLK, output RISE becomes "1"; if there is a transition from "1" to "0" at input CLK, output FALL becomes "1"; if neither of these two instances occurs, both outputs become "0".

The parameters EN and ENO can additionally be projected.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
CLK	BOOL	Clock input
RISE	BOOL	Indicator signal of rising edge
EDGE	BOOL	Indicator of all types of edges
FALL	BOOL	Indicator signal of falling edge

UDINT_AS_WORD: Type conversion

33

Overview

Introduction

This chapter describes the UDINT_AS_WORD block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block converts one input word from Data type UDINT to 2 output values of data type WORD.

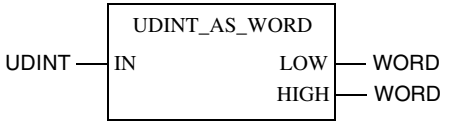
The individual words of the UDINT input are assigned to the outputs according to the output names.

It is possible to project EN and ENO as additional parameters.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
IN	UDINT	Input
LOW	WORD	less significant word
HIGH	WORD	more significant word

WORD_AS_BYTE: Type conversion

34

Overview

Introduction

This chapter describes the WORD_AS_BYTE block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The function block converts one input word from Data type WORD to two output values of data type BYTE.

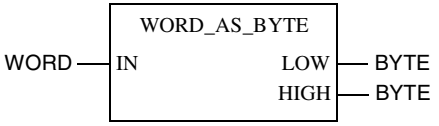
The individual bytes of the word at the input are assigned to the outputs according to the output names.

The parameters EN and ENO can additionally be projected.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
IN	WORD	Input
LOW	BYTE	less significant byte
HIGH	BYTE	more significant byte

WORD_AS_DINT: Type conversion

35

Overview

Introduction

This chapter describes the WORD_AS_DINT block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function converts 2 input words from Data type WORD to an output value of data type DINT.

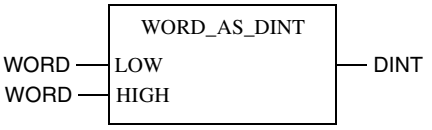
The input values are assigned to the word at the output according to the input names.

The parameters EN and ENO can additionally be projected.

Representation

Symbol

Block representation:



Formula

Block formula:

$$OUT = \{HIGH,LOW\}$$

Parameter description

Block parameter description:

Parameter	Data type	Meaning
LOW	WORD	less significant word
HIGH	WORD	more significant word
OUT	DINT	Output value

WORD_AS_REAL: Type conversion

36

Overview

Introduction

This chapter describes the WORD_AS_REAL block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function converts 2 input words from Data type WORD to an output value of data type REAL.

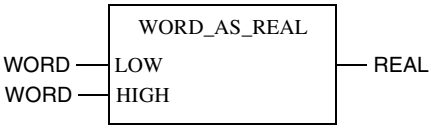
The input values are assigned to the word at the output according to the input names.

It is possible to project EN and ENO as additional parameters.

Representation

Symbol

Block representation:



Formula

Block formula:

$$OUT = \{HIGH,LOW\}$$

Parameter description

Block parameter description:

Parameter	Data type	Meaning
LOW	WORD	less significant word
HIGH	WORD	more significant word
OUT	REAL	Output value

WORD_AS_TIME: Type conversion

37

Overview

Introduction

This chapter describes the WORD_AS_TIME block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function converts 2 input words from Data type WORD to an output value of data type TIME.

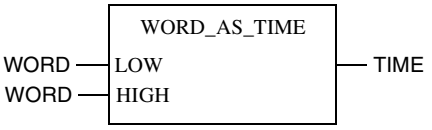
The input values are assigned to the word at the output according to the input names.

The parameters EN and ENO can additionally be projected.

Representation

Symbol

Block representation:



Formula

Block formula:

$$OUT = \{HIGH,LOW\}$$

Parameter description

Block parameter description:

Parameter	Data type	Meaning
LOW	WORD	less significant word
HIGH	WORD	more significant word
OUT	TIME	Output value

WORD_AS_UDINT: Type conversion

38

Overview

Introduction

This chapter describes the WORD_AS_UDINT block.

What's in this Chapter?

This chapter contains the following topics:

Brief description

Function description

The Function converts 2 input words from Data type WORD to an output value of data type UDINT.

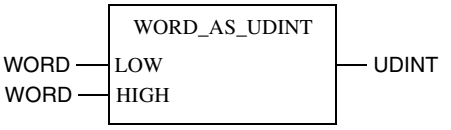
The input values are assigned to the word at the output according to the input names.

It is possible to project EN and ENO as additional parameters.

Representation

Symbol

Block representation:



Formula

Block formula:

$$OUT = \{HIGH,LOW\}$$

Parameter description

Block parameter description:

Parameter	Data type	Meaning
LOW	WORD	less significant word
HIGH	WORD	more significant word
OUT	UDINT	Output value

WORD_TO_BIT: Type conversion

39

Overview

Introduction

This chapter describes the WORD_TO_BIT block.

What's in this Chapter?

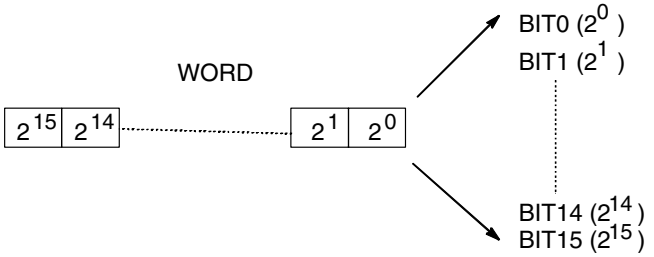
This chapter contains the following topics:

Brief description

Function description

The function block converts one input word from Data type WORD to 16 output values of data type BOOL.

The individual bits of the word at the input are assigned to the outputs according to the output names.

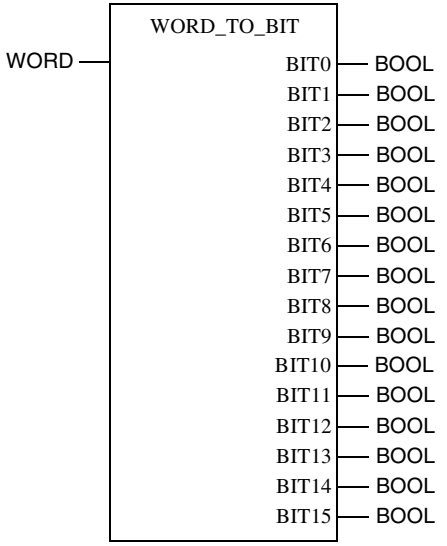


The parameters EN and ENO can additionally be projected.

Representation

Symbol

Block representation:

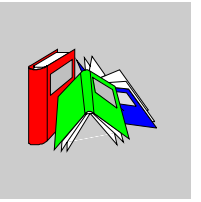


Parameter description

Block parameter description:

Parameter	Data type	Meaning
IN	WORD	Input
BIT0	BOOL	Output BIT0
BIT1	BOOL	Output BIT1
:	:	:
BIT15	BOOL	Output BIT15

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)

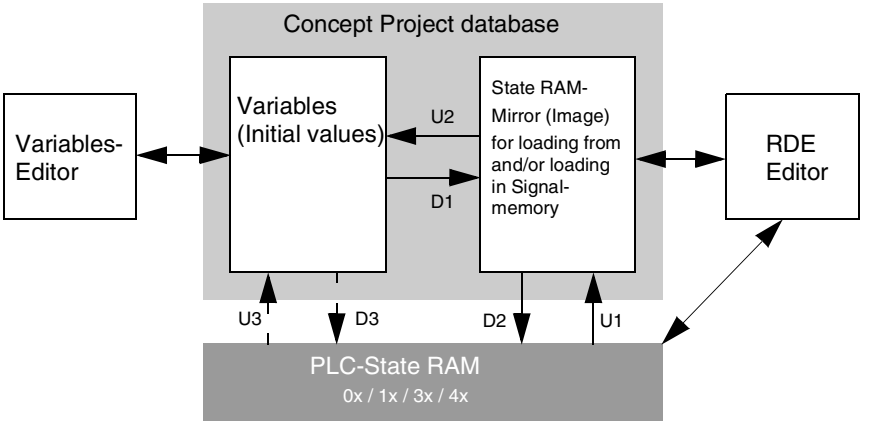
To assign the data type REAL, the value may also be specified in the following manner: 23.0.
Entering a comma will automatically assign the data type REAL.

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

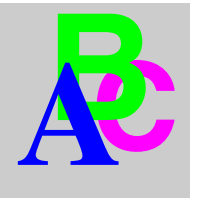
Warning

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



E

F

H

I

L

M

N

O

P

R

S

T

U

W