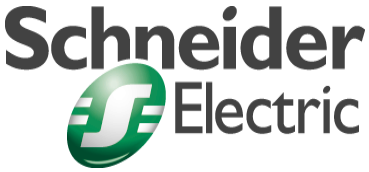
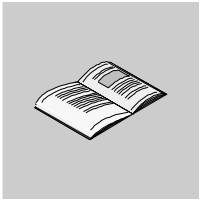
Concept
IEC Block Library
Part: FUZZY

840 USE 504 00 eng Version 2.6

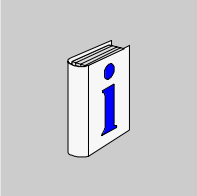


33002215.00

Table of Contents



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

User Comments

We welcome your comments about this document. You can reach us by e-mail at TECHCOMM@modicon.com

General information on the FUZZY Block Library

I

Overview

Introduction

This section contains general information on the FUZZY Block Library.

What's in this Part?

This part contains the following chapters:

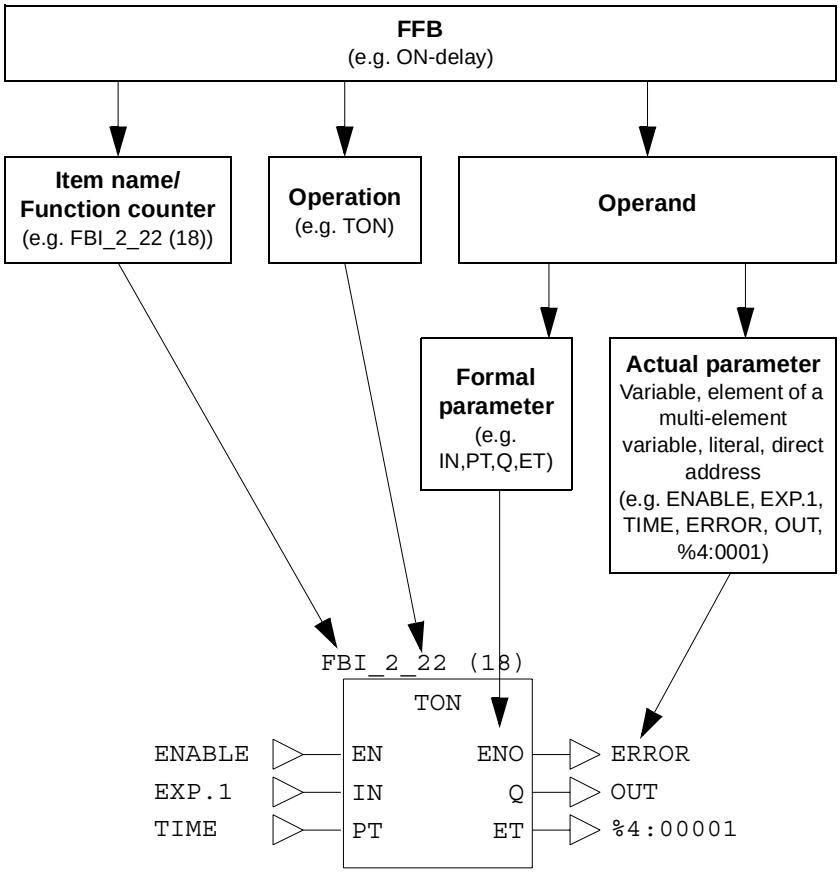
Parameterizing functions and function blocks

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.

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.

Fuzzy Control

2

Overview

Introduction

In this chapter, first of all a more precise explanation of what Fuzzy Control is and how Fuzzy Control can be used for control and regulatory functions will be provided. For the use of Fuzzy Control by means of Concept, elementary functions and function blocks (EFBs) are provided, the interaction of which is also more precisely explained in this chapter.

What's in this Chapter?

This chapter contains the following sections:

2.1 Introduction to Fuzzy Control Theory

Overview

Introduction

This section looks at the fundamental principles of Fuzzy Control

Note: This chapter does not attempt to describe Fuzzy Control in a mathematically exact way, but rather to provide an understanding of Fuzzy Control. More extensive information on the theory can be found in the relevant literature.

What's in this Section?

This section contains the following topics:

Fundamental Principles of Fuzzy Control

Introduction

The expression "Fuzzy Logic" describes a theory of "ambiguous quantities".
In science or in computer programs, it is usually the case that only two values are used, true or false, on or off, 1 or 0. There are no intermediate values.
The main idea of Fuzzy Logic on the other hand is the dealing with ambiguous quantities, the elements of which only gradually belong to a set. Instead of just "membership" and "non-membership", intermediate stages are also permitted. With the help of Fuzzy Logic, ambiguities of the most varying kinds can thus be mathematically described and handled.
This theory finds its most successful application in control engineering as Fuzzy Control.

Example: Temperature

If one considers a physical magnitude, such as temperature for example, this magnitude is generally represented by a number and a physical unit, for example 21 degrees C.
The temperature can also be described just as well verbally through attributes. The temperature can be described with the attributes "cold", "pleasant" and "warm" when we speak of room temperature, for example. These attributes are no longer precisely limited in relation to one another, they are ambiguous descriptions of the temperature.
Room temperature of 15 degrees C is generally characterized as "cold,", while a temperature of 21 degrees C is characterized as "warm". In relation to human feeling, there is not a sharp dividing line between "cold" and "pleasant", as is known in classical logic which only works with the truth terms "TRUE" (1) and "FALSE" (0). It has much more to do with a gradual transition from "cold" to "pleasant".

Fuzzy Control in control engineering

Introduction

In control engineering, one or more control variables are produced depending on one or more input variables. The input variables are logically and numerically associated with one another. The result of the association or the calculation are the output variables. Something similar occurs with Fuzzy Control.
The input variable is converted into a linguistic variable with the help of linguistic terms and finally evaluated with the help of rules. The result, also a "fuzzy" linguistic variable, must now again be converted into a control variable, since a valve cannot be controlled with a "fuzzy" variable.

Fuzzy Control in control engineering

Process in control engineering

Level	Description
1	The input variables are fuzzified
2	The linguistic variables produced are associated with the corresponding operators in accordance with specific rules. Result of the processing of all rules (inference): a "fuzzy" output variable, that is to say a variable which is described with its membership degrees through attributes.
3	Defuzzification of the output variables into a unambiguous number with which an action can then be associated in the control process.

Concepts of Fuzzy Theory

Linguistic Variables and Linguistic Terms

Example
If the temperature is defined as a linguistic variable, then it can be described by the linguistic terms "cold", "pleasant" and "hot".

Definition

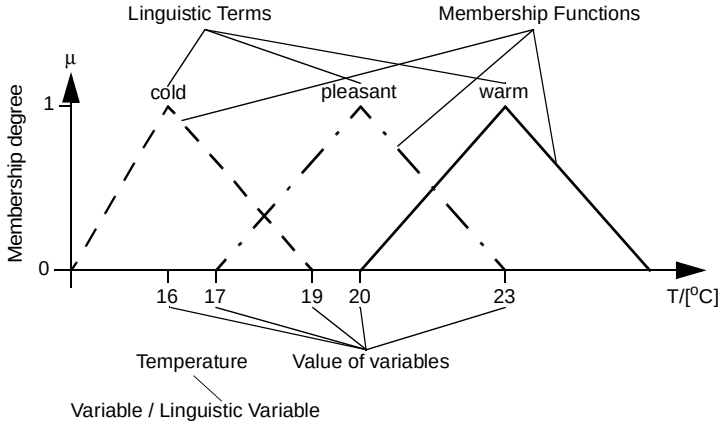
- A linguistic variable is a variable (e.g. temperature) which is described by linguistic terms.
- Linguistic terms describe attributes of a linguistic variable.

Membership Degree

Example
The linguistic terms "cold", "pleasant" and "warm" cannot be clearly distinguished from one another. In order to properly characterize this fuzzy dividing line, every linguistic term is assessed and given a so-called membership degree.
This membership degree now indicates to what extent the temperature is really cold.

- A membership degree of 0 for the linguistic term "cold" means that the temperature is not cold at all.
- A membership degree of 1 for the linguistic term "cold" means that the temperature is 100% cold; it is really cold. The various concepts are depicted once more in the illustration.

Example for the membership degree



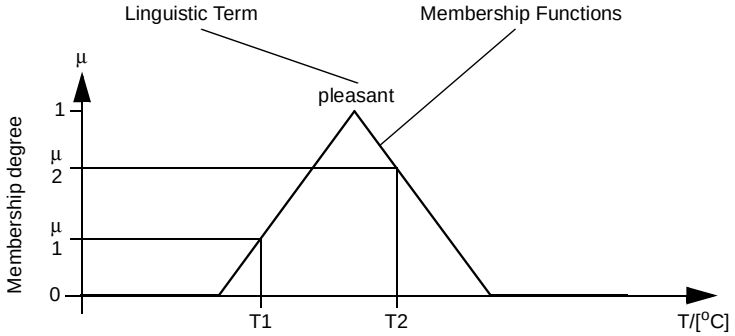
Definition
The membership degree defines to what extent a physical value is assigned to a linguistic term.

- 0 means that the value does not correspond at all to the linguistic term.
- 1 means that the value corresponds completely to the linguistic term.

Membership Function

Example

The gentle transition from "not really cold" to "really cold" is described by a function. These kinds of functions are so-called membership functions, since they describe the membership degree to its linguistic term for each physical value.
In the case of the variable temperature, the membership function describes not only whether a temperature is "cold" or not, the membership function assigns each physical value a confidence value with the expression "the temperature is cold" - this is the membership degree.
In general it is sufficient to characterize the membership via relatively simple courses of the function such as for example triangles, trapezoids or ramps.
In this way a membership degree can be determined for each physical value of the variable temperature.



(T1, μ1), (T2, μ2): Elements of the Fuzzy Set

In this example, the membership degree for two different temperature values is determined. If one has defined a membership function to a linguistic term, then one can determine the membership of the variable temperature to the appropriate linguistic term for every temperature with the help of the membership degree.

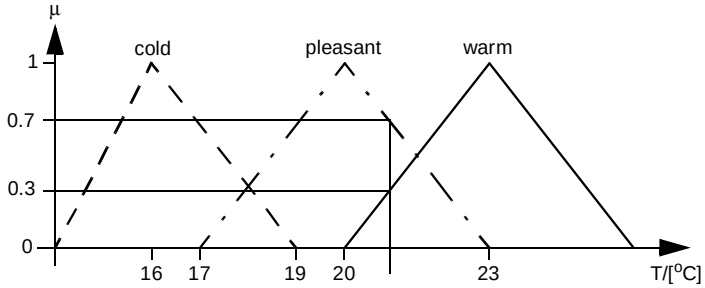
Definition

The membership function defines a membership degree to a linguistic term for each physical value.

Fuzzification

Example

Fuzzification of the input variable temperature



The input variable in the conventional "sharp" depiction in comparison to the fuzzy depiction

- Sharp depiction: The temperature is 21 degrees C.
- Fuzzy depiction:
  the temperature is:
  - cold to a membership degree of 0.0
  - pleasant to a membership degree of 0.7
  - warm to a membership degree of 0.3

Definition

The determination of the membership degrees for all linguistic terms of a variable in relation to an input variable is called fuzzification.

Rules

Example
If we consider ourselves and our daily activities, we can see that our activities are based on various prerequisites. We do something when a prerequisite is met. In order to describe the activities/reaction we use rules.
For example we say:
- If the room temperature is perceived as cold, then open the heating valve a little or we say:
- If the room temperature is perceived as warm, then close the heating valve a little.

Another rule could say:
- If it is summer and the outdoor temperature is not cold, then turn the heating off completely.

Behind these rules stands the knowledge of an expert, who, through more or less extensive learning, has acquired the ability to influence a process in a desired manner. The knowledge of the expert is not just limited to the type of action (open the valve a little), but is at the same time associated with the knowledge of what "a little" means, for example what angle of rotation corresponds to this.

Definition
A rule forms a (fuzzy) statement on the output variable for a specific (fuzzy) input variable.

Operators

Example
Usually conditions (premises) are associated in the rules with one another through the verbal expressions "AND" and "OR". In the example, the premise "If it is summer" (the basic variable is the season and the linguistic term is summer) is associated with the premise "if the outdoor temperature is not cold" in such a way that both statements must be true in order to be able to carry out the conclusion. These AND and OR associations are processed by mathematical operators in fuzzy technology. In Fuzzy Theory there are a series of operators which realize the AND associations on the one hand and the OR associations on the other hand. The simplest operator for the AND association is the "minimum" and the simplest operator for the "OR" association is the "maximum".

Definition
The association of conditions takes place through operators.

Weighting of the rules

In drawing up rules it can happen that they should have different weights. For example, one rule always applies (100%), another rule however does not always apply (80%). In order to be able to express this, there is the possibility of assigning each rule a degree of plausibility. In general this is achieved mathematically by multiplying the association result by the degree of plausibility.

Inference

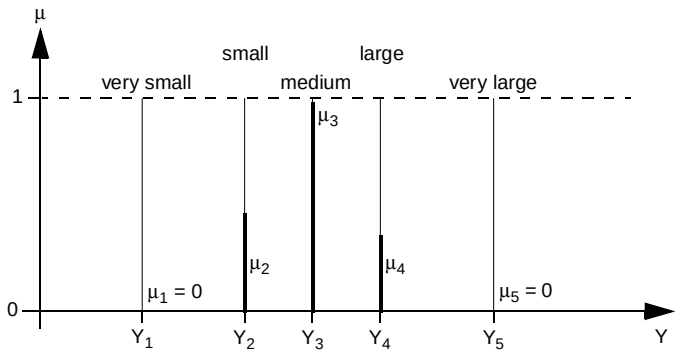
The inference is the result of the processing of all rules

Defuzzification

Example

The simplest possibility to create a fuzzy variable from a sharp variable is for every attribute of the fuzzy variable to supply a suggestion for the sharp variable. Each linguistic term is thus represented by a set numerical value for the variable. One speaks in this case of so-called singletons. The suggestions made for each linguistic term are now weighted with the appropriate membership degrees. In this way the fuzzy variable is arithmetically transformed back into a sharp variable

Singletons



Note: There is also a series of other defuzzification methods in Fuzzy Theory, some of which are characterized by a great deal of calculation. These will not be looked at at this point.

Definition

The transformation of fuzzy variables into sharp variables is called defuzzification.

2.2 Fuzzy Control in Concept

Overview

Introduction

This chapter describes how Fuzzy Logic is projected and which EFBs are provided by Concept for this purpose.

What's in this Section?

This section contains the following topics:

EFBs in the FUZZY Library

EFBs in the Fuzzy Library

EFBs for INT- and REAL-arithmetic

Operation	EFB	Group	Description
Fuzzification	FUZ_STERM_INT (See *FUZ_STERM_***: Fuzzification of one term, p. 67*), FUZ_STERM_REAL (See *FUZ_STERM_***: Fuzzification of one term, p. 67*)	Fuzzify	Fuzzification of a term
Fuzzification	FUZ_ATERM_INT (See *FUZ_ATERM_INT, FUZ_ATERM_REAL: Fuzzification of all terms, p. 49*), FUZ_ATERM_REAL (See *FUZ_ATERM_INT, FUZ_ATERM_REAL: Fuzzification of all terms, p. 49*)	Fuzzify	Fuzzification of up to 9 terms at one time
Fuzzification	FUZ_ATERM_STI (See *FUZ_ATERM_STI, FUZ_ATERM_STR: Fuzzification of all terms (structure), p. 55*), FUZ_ATERM_STR (See *FUZ_ATERM_STI, FUZ_ATERM_STR: Fuzzification of all terms (structure), p. 55*)	Fuzzyfy_Struct	Fuzzification of up to 9 terms at one time. Store result in data structure.
Inference	FUZ_MAX_INT (See *FUZ_MAX_***: Fuzzy Maximum, p. 59*), FUZ_MAX_REAL (See *FUZ_MAX_***: Fuzzy Maximum, p. 59*)	Operators_OR	OR operator: Maximum
Inference	FUZ_MIN_INT (See *FUZ_MIN_***: Fuzzy Minimum, p. 61*), FUZ_MIN_REAL (See *FUZ_MIN_***: Fuzzy Minimum, p. 61*)	Operators_AND	AND operator: Minimum

Operation	EFB	Group	Description
Inference	FUZ_SUM_INT (See *FUZ_SUM_***: Fuzzy Sum, p. 75*), FUZ_SUM_REAL (See *FUZ_SUM_***: Fuzzy Sum, p. 75*)	Operators_OR	OR operator: Sum
Inference	FUZ_PROD_INT (See *FUZ_PROD_***: Fuzzy Product, p. 63*), FUZ_PROD_REAL (See *FUZ_PROD_***: Fuzzy Product, p. 63*)	Operators_AND	AND operator: Product
Defuzzification	DEFUZ_INT (See *DEFUZ_INT, DEFUZ_REAL: Defuzzification with singletons, p. 37*), DEFUZ_REAL (See *DEFUZ_INT, DEFUZ_REAL: Defuzzification with singletons, p. 37*)	Defuzzify	Defuzzification with singletons
Defuzzification	DEFUZ_STI (See *DEFUZ_STI, DEFUZ_STR: Defuzzification with singletons (structure), p. 43*), DEFUZ_STR (See *DEFUZ_STI, DEFUZ_STR: Defuzzification with singletons (structure), p. 43*)	Defuzzify_Struct	Defuzzification with singletons. Fetch inputs from data structure.

The difference between integer arithmetic and real arithmetic is:

- the solution inside the calculation.
  The solution of the membership degree is 0.01 %.
  The range of the membership degrees 0...1 is scaled to 0...10 000.
- the possibility of working with physical values in real arithmetic.
- the execution time, which is longer than in real arithmetic.

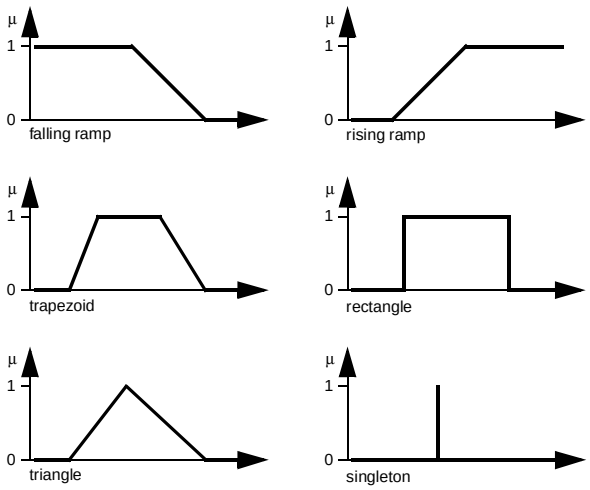
Fuzzification

Introduction

Firstly, all variables that are to be associated with one another through linguistic rules must be fuzzified. For this the number of attributes a variable is to be assigned must first be considered. This depends on whether these attributes are also to be used in the rules. Usually 3 to 5 attributes are sufficient. In Concept you have the possibility of selecting a fuzzification for each term of a variable or a fuzzification of up to 9 terms of one variable.

Fuzzification with FUZ_STERM

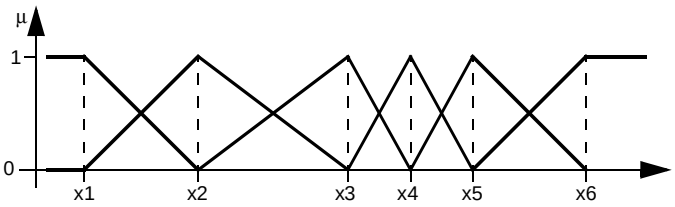
The fuzzification of an individual term takes place with the membership function, which supports up to 4 support points.
The membership functions can for example look like this:



For more information please see the description of EFBs FUZ_STERM (See *FUZ_STERM_***: Fuzzification of one term, p. 67*).

Fuzzification with FUZ_ATERM or FUZ_ATERM_ST

In general the functions that are used the most are ramp and triangle. The fuzzification of an individual term offers the greatest flexibility for experienced fuzzy users, but the easiest way to fuzzify a variable is to use the EFB FUZ_ATERM (See *FUZ_ATERM_INT, FUZ_ATERM_REAL: Fuzzification of all terms, p. 49*) or FUZ_ATERM_ST (See *FUZ_ATERM_STI, FUZ_ATERM_STR: Fuzzification of all terms (structure), p. 55*) since these EFBs fuzzify not just one term but all the terms of a fuzzy variable. These EFBs also simplify the input of membership functions for the different terms.
The idea behind this simplification is that the different membership functions of all terms of a fuzzy variable are not independent of one another. In most cases, the total of the membership degrees of two successive functions is 1. If, for the sake of simplicity, only triangular functions are used instead of trapezoidal functions there results the following example of the definition of a membership function for 6 terms



In this case there are only 6 base points to be defined instead of 2 x 2 + 4 x 3 = 14 base points in the case of a fuzzification with FUZ_STERM (See *FUZ_STERM_***: Fuzzification of one term, p. 67*). The EFB FUZ_ATERM (See *FUZ_ATERM_INT, FUZ_ATERM_REAL: Fuzzification of all terms, p. 49*) / FUZ_ATERM_ST (See *FUZ_ATERM_STI, FUZ_ATERM_STR: Fuzzification of all terms (structure), p. 55*) allows a maximum of 9 attributes.
The fuzzification must be carried out for each input variable. The result of each fuzzification are membership degrees for each term that are assigned to the attributes. In using the EFBs FUZ_ATERM (See *FUZ_ATERM_INT, FUZ_ATERM_REAL: Fuzzification of all terms, p. 49*) the result is the creation of individual variables for the use of graphic connections in the inference. In using the EFBs FUZ_ATERM_ST (See *FUZ_ATERM_STI, FUZ_ATERM_STR: Fuzzification of all terms (structure), p. 55*) the membership degrees are summed in a data structure.

Inference

Realization of the rules

The rules are realized through the concatenation of the membership degrees of the fuzzy variables. The inputs of the EFBs, which represent the fuzzy operators, are combined with the membership grades (created by the fuzzy EFBs). The pairs FUZ_MIN (See *FUZ_MIN_***: Fuzzy Minimum, p. 61*)/FUZ_MAX (See *FUZ_MAX_***: Fuzzy Maximum, p. 59*) and FUZ_PROD (See *FUZ_PROD_***: Fuzzy Product, p. 63*)/FUZ_SUM (See *FUZ_SUM_***: Fuzzy Sum, p. 75*) have proven to be good combinations for the AND/OR concatenation. Use the pair FUZ_MIN (See *FUZ_MIN_***: Fuzzy Minimum, p. 61*)/FUZ_MAX (See *FUZ_MAX_***: Fuzzy Maximum, p. 59*) in a first approach.

Weighting the rules

Rules must be weighted with a multiplication. In real arithmetic, this purpose is served by the EFB FUZ_PROD_REAL (See *FUZ_PROD_***: Fuzzy Product, p. 63*). In integer arithmetic this purpose is served by the EFB FUZ_PROD_INT (See *FUZ_PROD_***: Fuzzy Product, p. 63*), which takes into account the normalized form of the membership degrees from 0 ... 10 000.

Defuzzification

Principle

The results of the application of all rules are in turn membership degrees for attributes of the output variable. In order to achieve a usable result (attributes with membership degrees cannot for example be used directly to control values), the membership degrees of all terms of the variables must be usefully combined. This is achieved with the defuzzification function block DEFUZ (See *DEFUZ_INT, DEFUZ_REAL: Defuzzification with singletons, p. 37*) .

Working method in Concept

The function block DEFUZ (See *DEFUZ_INT, DEFUZ_REAL: Defuzzification with singletons, p. 37*) creates, with the help of singletons from the membership degrees, an unambiguous value for the output variables, which is assigned to the terms of the output variables.

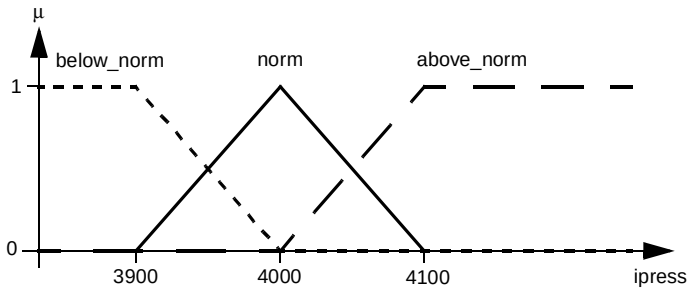
Example for Concept

Defaults

The example is based on integer logic. There are 2 variables, "ipres" and "itemp", which are to be assessed in accordance with four prescribed rules.
Fuzzification of the variable "ipress" (ipress has 3 linguistic terms)

- below_norm
- norm
- above_norm

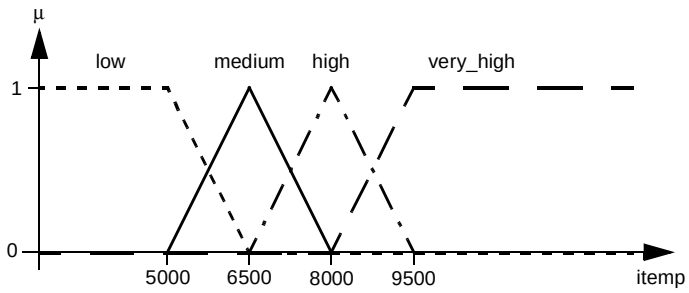
Fuzzification of the variable "ipress"



Fuzzification of the variable "itemp" (itemp has 4 linguistic terms)

- low
- medium
- high
- very_high

Fuzzification of the variable "itemp"



The following four rules are valid:

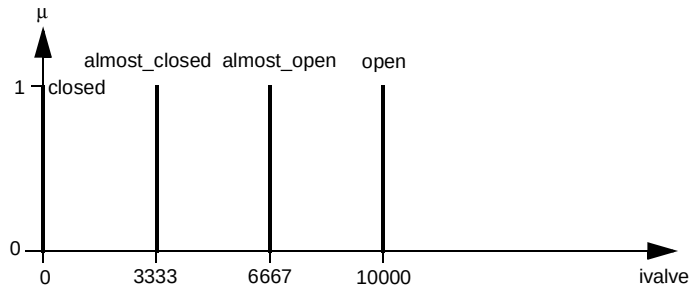
- IF ipress = norm AND itemp = high THEN valve= almost_closed
- IF ipress = above_norm AND itemp = very_high THEN valve = closed
- IF ipress = norm AND itemp = medium THEN valve= almost_open
- IF ipress = below_norm AND itemp = low THEN valve = open

The defuzzification should take place with singletons.

Defuzzification of the variable "ivalve" (ivalve has 4 linguistic terms)

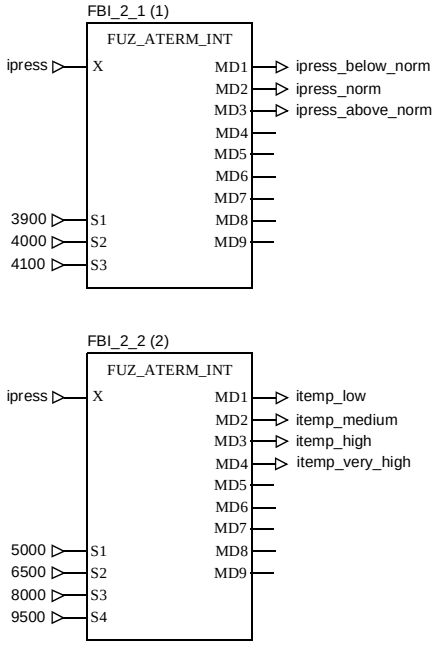
- closed
- almost_closed
- almost_open
- open

Defuzzification of the variable "ivalve"

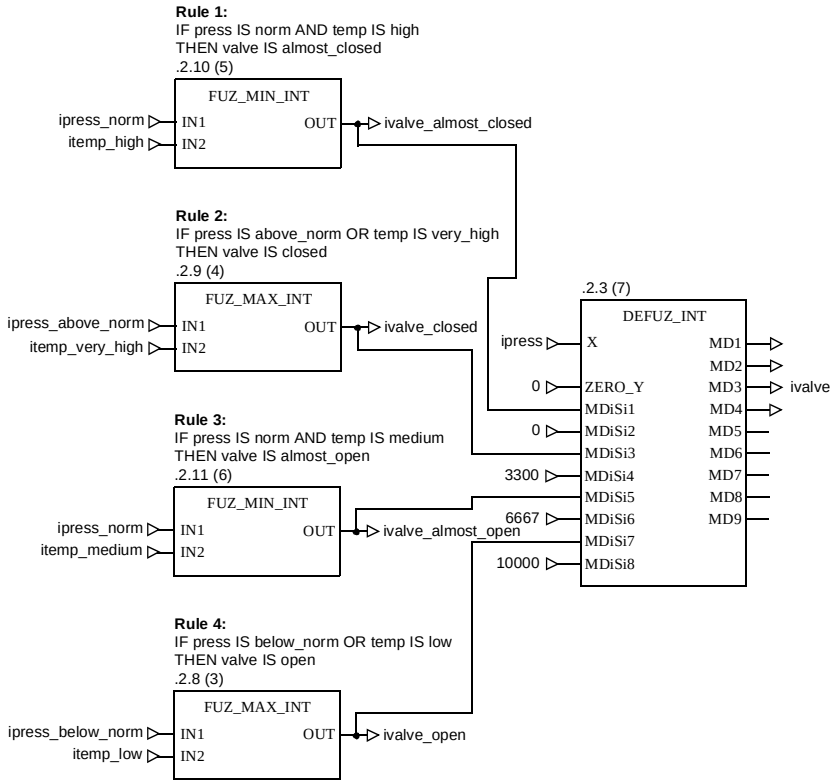


Realization in Concept

Fuzzification:



Defuzzification:



EFB descriptions

II

Overview

Introduction

These EFB descriptions are documented in alphabetical order.

Note: The number of inputs of some EFBs can be increased to a max. of 32 through vertical size alteration of the FFB symbol. See the description of the individual EFBs to determine which EFBs.

What's in this Part?

This part contains the following chapters:

DEFUZ_INT, DEFUZ_REAL: Defuzzification with singletons

3

Overview

Introduction

This chapter describes the blocks:
- DEFUZ_INT
- DEFUZ_REAL

What's in this Chapter?

This chapter contains the following topics:

Brief Description

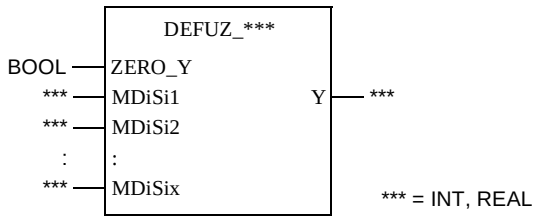
Brief Description

The function defuzzifies linguistic terms that are represented by singletons, in accordance with the maximum mean method. The position of the singletons is defined via support points (S1 ... S9). Each term is weighted with the respective membership degree (MD1 ... MD9). The range of the membership degrees for data type INT is 0 ... 10 000 and for data type REAL is 0 ... 1. The meaning of the inputs (expandable inputs MDiSi) can be inferred from the *Parameter description, p. 39* .
The number of inputs (MDiSi) can be increased to a max. of 18 through vertical size alteration of the block frame. This corresponds to 9 singletons. Additional inputs cannot be projected.
The data types of all input values (MDiSi) and those of the output value must be the same. There is a special function block available for the processing of each of the different data types.
Additional parameters EN and ENO can be projected.

Description

Symbol

Block description:



Formula

Block formula:

$$Y = \frac{\sum_{i=1}^{n} MDi \times Si}{\sum_{i=1}^{n} MDi}$$

Explanation: n = number of singletons
Prerequisite: $2 \leq n \leq 9$

Parameter description

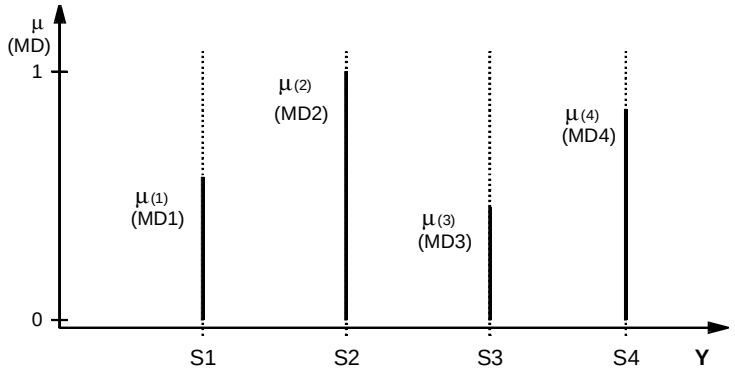
Block parameter description:

Parameter	Data type	Meaning
ZERO_Y	BOOL	0: Output of the last Y value
MDiSi1	INT, REAL	Singleton: 1; MD1 Membership Degree
MDiSi2	INT, REAL	Singleton: 1; S1 Support point
MDiSi3	INT, REAL	Singleton: 2; MD2 Membership Degree
MDiSi4	INT, REAL	Singleton: 2; S2 Support point
:	:	:
MDiSi17	INT, REAL	Singleton: 9; MD9 Membership Degree
MDiSi18	INT, REAL	Singleton: 9; S9 Support point
Y	INT, REAL	Output

Detailed description

Function Description

The function block DEFUZ is available as membership function for the mean of maximum method with singletons for the defuzzification of linguistic variables. The position of the singletons is defined with support points.



In projecting the functional block, attention should be given that the inputs are always used in pairs, since every linguistic term is weighted via the respective membership degree. The meaning of the input indices is described in *Parameter description, p. 39*.
The output Y is set to 0.
If all memberships grades have the value 0, then the output behaviour of the function block can be determined via the input ZERO_Y:

ZERO_Y value	**Result**
ZERO_Y = 0	The output Y remains unchanged.
ZERO_Y = 1	The output Y is set to 0.

Runtime errors

Runtime errors

An error message results when

- more than 9 support points (this corresponds to a max. of 18 MDiSi inputs) were projected (E_EFB_TOO_MANY_INPUTS),
- the functional block is initialized with an uneven number of inputs (E_EFB_WRONG_NUMBER_OF_INPUTS) or
- one of the membership degrees MD1 ... MD9 is outside of the range (E_EFB_INPUT_VALUE_OUT_OF_RANGE). Only the following ranges are possible:
  - INT: 0 ... 10 000
  - REAL: 0 ... 1

DEFUZ_STI, DEFUZ_STR: Defuzzification with singletons (structure)

4

Overview

Introduction

This chapter describes the blocks:
- DEFUZ_STI
- DEFUZ_STR

What's in this Chapter?

This chapter contains the following topics:

Brief Description

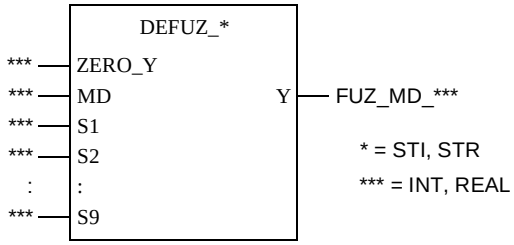
Function description

The function defuzzifies linguistic terms that are represented by singletons, in accordance with the maximum mean method. The position of the singletons is defined via support points (S1 ... S9). Every term is weighted with the respective membership degree (term1 ... term9) from the data structure FUZ_MD_INT (for DEFUZ_STI) or FUZ_MD_REAL (for DEFUZ_STR). The range of the membership degrees for data type INT is 0 ... 10 000 and for data type REAL is 0 ... 1.
The number of inputs (S1 ... Sn) can be increased to a max. of 9 through vertical size alteration of the block frame. Additional inputs cannot be projected.
The data types of all input values (Sn) must be the same. There is a special function block available for the processing of each of the different data types.
Additional parameters EN and ENO can be projected.

Representation

Symbol

Block representation:



Form

Form of block

$$Y = \frac{\sum_{i=1}^{n} MDi \times Si}{\sum_{i=1}^{n} MDi}$$

Definition: n = number of singletons
Prerequisite: $2 \leq n \leq 9$

Parameter description

DEFUZ_STI, DEFUZ_STR

Parameter	Data type	Meaning
ZERO_Y	BOOL	0: Output of the last Y value 1: Output Y is set to "0"
MD	FUZ_MD_INT, FUZ_MD_REAL	Membership Degree (term1 ... term9)
S1	INT, REAL	Singleton: 1
S2	INT, REAL	Singleton: 2
:	:	:
S9	INT, REAL	Singleton: 9
Y	INT, REAL	Output

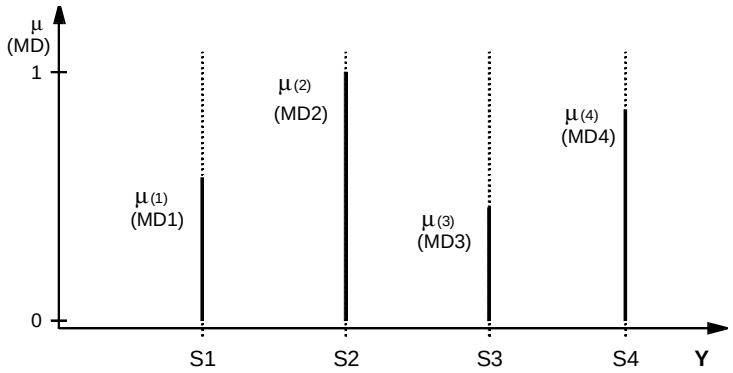
FUZ_MD_INT, FUZ_MD_REAL

Element	Data type	Meaning
n	INT	Number of terms
term1	INT, REAL	Membership Degree (MD1)
:	:	:
term9	INT, REAL	Membership Degree (MD9)

Detailed description

Function description

The function block DEFUZ_ST is available as membership function for the defuzzification of linguistic variables for the mean of maximum method with singletons.
The position of the singletons is defined with the support points (S1 ... S9).



Each linguistic term is weighted via the respective membership degree.
If all memberships grades have the value 0, then the output behaviour of the function block can be determined via the input ZERO_Y:

ZERO_Y value	**Result**
ZERO_Y = 0	The output Y remains unchanged.
ZERO_Y = 1	The output Y is set to 0.

Runtime errors

Runtime errors

An error message (E_EFB_TOO_MANY_INPUTS) results if more than 9 support points were projected.
A warning (E_EFB_WRONG_NUMBER_OF_INPUTS) results, if the number of inputs (=number of support points S1 ... Sn) does not correspond to the number of terms used (MD) in the data structure FUZ_MD_INT.n (for DEFUZ_STI) or FUZ_MD_REAL.n (for DEFUZ_STR). In this case the number of support points included in the calculation of the output value must correspond to the number of terms used (MD) in the data structure FUZ_MD_INT.n (for DEFUZ_STI) or FUZ_MD_REAL.n (for DEFUZ_STR).

FUZ_ATERM_INT, FUZ_ATERM_REAL: Fuzzification of all terms

5

Overview

At a Glance

This chapter describes the blocks:

- FUZ_ATERM_INT
- FUZ_ATERM_REAL

What's in this Chapter?

This chapter contains the following topics:

Brief Description

Function description

The function block fuzzifies up to 9 terms of the linguistic variables (input X) and indicates the individual membership degree (outputs MD1 ... MD9). The range at output for data type INT is 0 ... 10 000 and for data type REAL is 0 ... 1. The membership functions are defined via support points (expandable inputs S1 ... S9). The function block works with a special simplification for the definition of the membership functions:

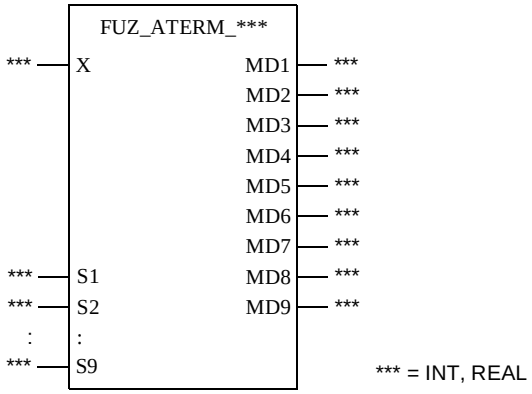
- Ramps for the first and last membership function
- only triangles for the membership functions between the first and the last
- the total of two membership degrees of two successive linguistic terms is always 1 (10 000)
- the sum of all membership degrees of all linguistic terms for each input value X is always 1 (10 000)

The number of support points (S1 ... Sx) can be increased to a max. of 9 through vertical size alteration of the block frame. Additional support points cannot be projected.
The number of the calculated membership degrees corresponds to the number of membership functions. If less than 9 membership functions are projected, then the other outputs are assigned the value 0. (e.g. for 4 membership functions, 4 membership degrees for MD1 ... MD4 are calculated and MD5 ... MD9 are to 0).
The data types of all input values and that of the output value must be the same. There is a special function block available for the processing of each of the different data types.
Additional parameters EN and ENO can be projected.

Representation

Symbol

Block representation:



Parameter description

Block parameter description:

Parameters	Data type	Meaning
X	INT, REAL	linguistic variable
S1	INT, REAL	Point S1
S2	INT, REAL	Point S2
:	:	:
S9	INT, REAL	Point S9
MD1	INT, REAL	Output MD1 Membership Degree
MD2	INT, REAL	Output MD2 Membership Degree
:	:	:
MD9	INT, REAL	Output MD9 Membership Degree

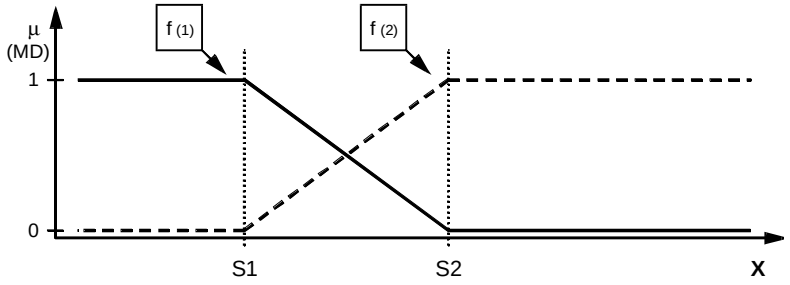
Detailed description

Parameter description

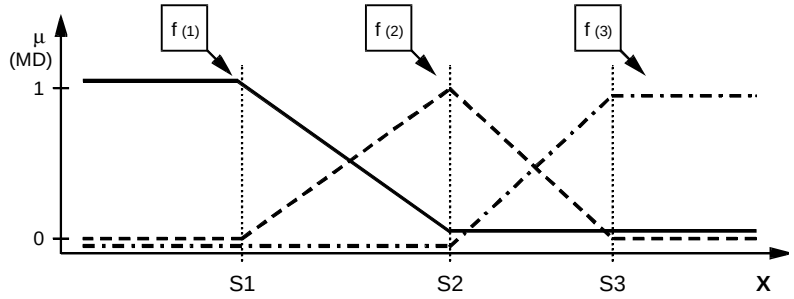
With the function block FUZ_ATERM all terms of a linguistic variable can be fuzzified simultaneously. The membership functions are determined via support points (S1, S2, S3,...). Through the concept of this fuzzification vertices are defined by a number of membership functions with one support point in each case. The resulting forms of the membership functions are ramps and triangles, in which the sum of the individual membership degrees is always 100%. This connection is explained in the following timing diagrams.

Timing diagrams

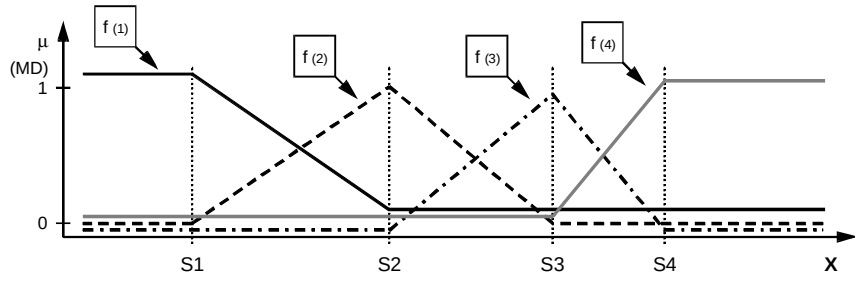
2 membership functions, 2 support points, 2 membership degrees



3 membership functions, 3 support points, 3 membership degrees



4 membership functions, 4 support points, 4 membership degrees



Runtime errors

Runtime errors

An error message results when

- the order of the support points is incorrect (E_EFB_INPUT_VALUE_OUT_OF_RANGE). The support points must be arranged in ascending order (S1 < S2 < S3 < ... < S9) or
- more than 9 support points were projected (E_EFB_TOO_MANY_INPUTS).

FUZ_ATERM_STI, FUZ_ATERM_STR: Fuzzification of all terms (structure)

6

Overview

Introduction

This chapter describes the blocks:

- FUZ_ATERM_STI
- FUZ_ATERM_STR

What's in this Chapter?

This chapter contains the following topics:

Brief Description

Function description

The function fuzzifies up to 9 terms of the linguistic variables (input X) and issues the individual membership degrees in the data structure FUZ_MD_INT (for FUZ_ATERM_STI) or FUZ_MD_REAL (for FUZ_ATERM_STR) in the elements term1 ... term9. The range at output for data type INT is 0 ... 10 000 and for data type REAL is 0 ... 1. The membership functions are defined via support points (expandable inputs S1 ... S9).
The function block works with a special simplification for the definition of the membership functions.

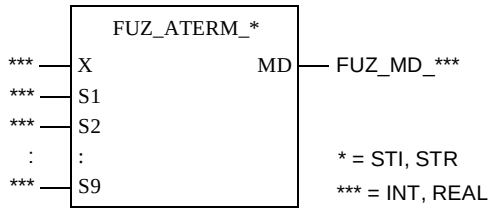
- Ramps for the first and last membership function
- only triangles for the membership functions between the first and the last
- the total of two membership degrees of two successive linguistic terms is always 1 (10 000)
- the sum of all membership degrees of all linguistic terms for every input value X is always 1 (10 000)

The number of support points (S1 ... Sx) can be increased to a max. of 9 through vertical size alteration of the block frame. Additional support points cannot be projected.
The number of the calculated membership degrees corresponds to the number of membership functions and is stored in the data structure (element n). If less than nine membership functions are projected, then the remaining elements of the data structure (termx) are not changed. (e.g. for 4 support points there are 4 membership functions; their 4 membership degrees are on the elements term1 ... term4; term5 .. term9 are not changed.) This has the advantage that data structures which are imaged in the signal memory only take up as much memory location as they actually require.
The data types of all input values must be the same. There is a special function block available for the processing of each of the different data types.
Additional parameters EN and ENO can be projected.

Description

Symbol

Block description:



Parameter description

FUZ_ATERM_STI, FUZ_ATERM_STR

Parameter	Data type	Meaning
X	INT, REAL	linguistic variable
S1	INT, REAL	Support point S1
S2	INT, REAL	Support point S2
:	:	:
S9	INT, REAL	Support point S9
MD	FUZ_MD_INT, FUZ_MD_REAL	Output Membership Degree (term1 ... term9)

FUZ_MD_INT, FUZ_MD_REAL

Element	Data type	Meaning
n	INT	Number of terms
term1	INT, REAL	Membership Degree (MD1)
:	:	:
term9	INT, REAL	Membership Degree (MD9)

Detailed description

Parameter description and timing diagram

The parameter description and timing diagram for this block can be found in *Detailed description, p. 52*

Runtime errors

Runtime errors

An error message results if

- the order of the support points is incorrect (E_EFB_INPUT_VALUE_OUT_OF_RANGE). The support points must be arranged in ascending order (S1 < S2 < S3 < ... < S9) or
- more than 9 support points were projected (E_EFB_TOO_MANY_INPUTS).

FUZ_MAX_***: Fuzzy Maximum

7

Overview

Introduction

This chapter describes the block FUZ_MAX_***.

What's in this Chapter?

This chapter contains the following topics:

Brief Description

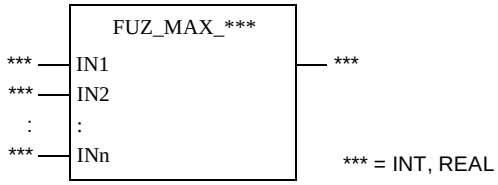
Function description

The function recognizes the largest input value and issues it at the output.
The data types INT and REAL can be processed. The range of the inputs and the output is for data type INT 0 ... 10 000 and for data type REAL 0 ... 1.
The data types of all input values and that of the output value must be the same.
There is a special function available for each of the different data types.
The number of inputs can be increased.
As additional parameter EN and ENO can be projected.

Description

Symbol

Block description:



Formula

Block Formula:

$OUT = MAX\{0,IN1,IN2,\ldots,INn\}$

Parameter description

Block parameter description:

Parameter	Data type	Meaning
IN1	INT, REAL	1st Input value
IN2	INT, REAL	2nd Input value
:	:	:
INn	INT, REAL	n. Input value
OUT	INT, REAL	Maximum value

FUZ_MIN_***: Fuzzy Minimum

8

Overview

Introduction

This chapter describes the block FUZ_MIN_***.

What's in this Chapter?

This chapter contains the following topics:

Brief Description

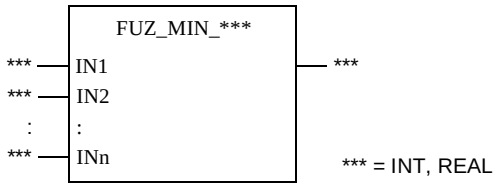
Function description

The function recognizes the smallest input value and issues it at the output.
The data types INT and REAL can be processed. The range of the inputs and the output is for data type INT 0 ... 10 000 and for data type REAL 0 ... 1.
The data types of all input values and that of the output value must be the same.
There is a special function available for each of the different data types.
The number of inputs can be increased.
Additional parameters EN and ENO can be projected.

Description

Symbol

Block description:



Formula

Block Formula:

REAL: OUT = MIN{1,IN1,IN2,…,INn}
INT: OUT = MIN{10000,IN1,IN2,…,INn}

Parameter description

Block parameter description:

Parameter	Data type	Meaning
IN1	INT, REAL	1st Input value
IN2	INT, REAL	2nd Input value
:	:	:
INn	INT, REAL	n. Input value
OUT	INT, REAL	Maximum value

FUZ_PROD_***: Fuzzy Product

9

Overview

Introduction

This chapter describes the block FUZ_PROD_***.

What's in this Chapter?

This chapter contains the following topics:

Brief Description

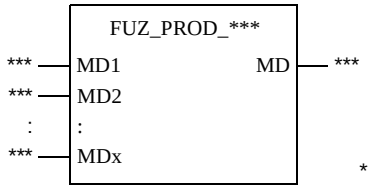
Function description

The function forms the product (output MD) of the membership degrees (expandable inputs MD1 ... MDx). In addition, the function (in the case of integer calculation) carries out a multiplication, taking into consideration the range of the membership degrees (0 ... 10 000). The range at the inputs and the output for the data type INT is 0 ... 10 000 and for data type REAL is 0 ... 1.
The number of inputs can be increased.
Additional parameters EN and ENO can be projected.

Description

Symbol

Block description:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
MD1	INT, REAL	1st Membership Degree
MD2	INT, REAL	2nd Membership Degree
:	:	:
MDx	INT, REAL	x. Membership Degree
MD	INT, REAL	Output product (fuzzy)

Detailed description

Function description

In real arithmetic, the formation of the product is achieved through simple multiplication.

Rule	Example
(0 ... 1) * (0 ... 1) = (0 ... 1)	0.3 * 0.6 = 0.18

In integer arithmetic, a correction must be carried out through the rescaling of the range:

Rule	Example
(0 ... 10 000) * (0 ... 10 000) = (0 ... 10 000)	3 000 * 6 000 = 1 800 (!)

FUZ_STERM_***: Fuzzification of one term

10

Overview

Introduction

This chapter describes the block FUZ_STERM_***.

What's in this Chapter?

This chapter contains the following topics:

Brief Description

Function description

The function fuzzifies an individual term of the linguistic variables (input X) and issues its membership degree to the output MD. The range of the output for data type INT is 0 ... 10 000 and for data type REAL is 0 ... 1. The membership function is determined by means of a maximum of 4 support points (S1 ... S4). Only 2 to 4 inputs (support points) are possible. For 2 support points the function realizes a ramp function. For 3 support points a triangular function and for 4 support points a trapezoidal function. Various function behaviors are possible through different initialization of the support points.

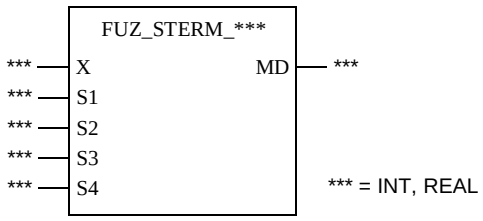
Note: For initialization with 4 support points, the processing of the membership function corresponds to SFB 361 FUZZYFY of AKF125.

The data types of all input values and that of the output value must be the same. There is a special function block available for the processing of each of the different data types.
The number of the inputs can be increased to a max. of 4 through vertical size alteration of the block frame.
Additional parameters EN and ENO can be projected.

Description

Symbol

Block description:



Parameter description

Block parameter description:

Parameter	Data type	Meaning
X	INT, REAL	linguistic variable
S1	INT, REAL	Support point S1
S2	INT, REAL	Support point S2
:	:	:
S4	INT, REAL	Support point S4
MD	INT, REAL	Output Membership Degree

Detailed description

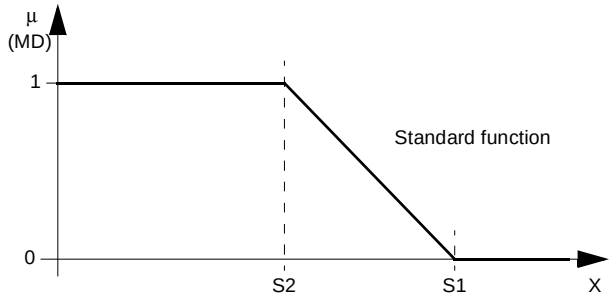
Parameter description

With the function block FUZ_STERM one term of a linguistic variable is fuzzified. The membership function can be defined with up to 4 support points (S1 ... S4). The support points must be given in ascending order. The standard functions can be found in the following table, for exceptions see timing diagram. Forms that deviate from this are possible through a corresponding inversion of the support points order. The possible sequences of functions are described in the following overview.
Standard functions

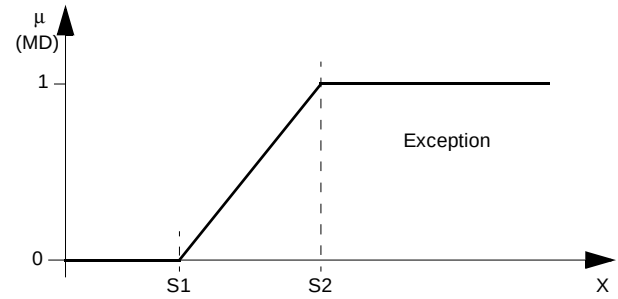
Membership function	Number of inputs	Condition
Falling ramp	2	S2<S1
Rising ramp	2	S1<S2
Triangle	3	S1<S2<S3
Trapezoid	4	S1<S2<S3<S4

Timing diagram

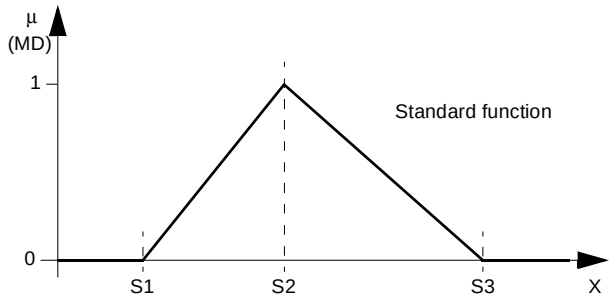
Function block with 2 inputs (S1 .... S2)
Falling ramp: S2 < S1



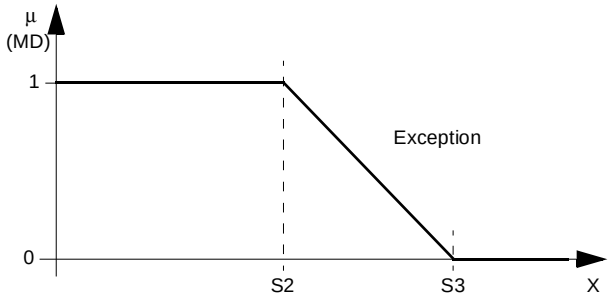
Rising ramp: S1 < S2



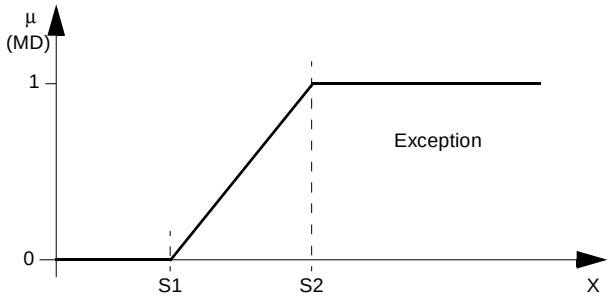
Function block with 3 inputs (S1 .... S3)
Triangle: S1 < S2 < S3



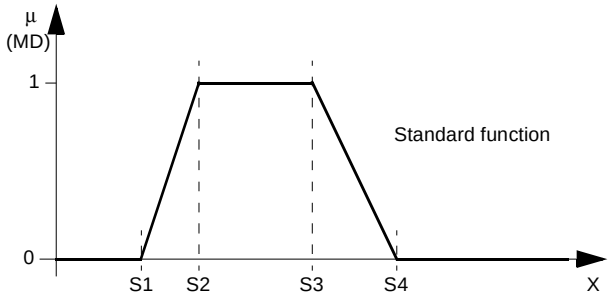
Falling ramp: S2 < S3 and S1 > S2



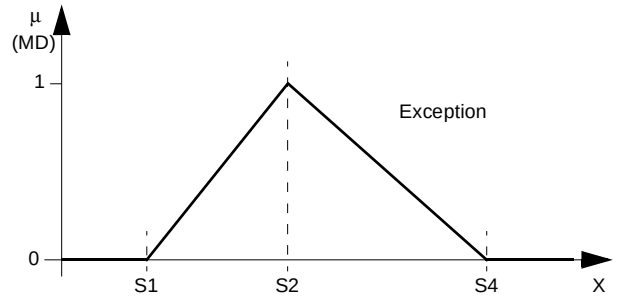
Rising ramp: S1 < S2 and S3 < S2



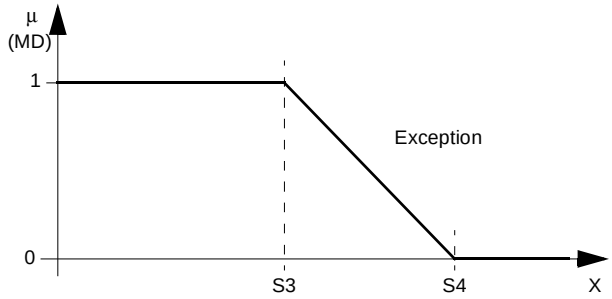
Function block with 4 inputs (S1 .... S4)
Trapezoid: S1 < S2 < S3 < S4



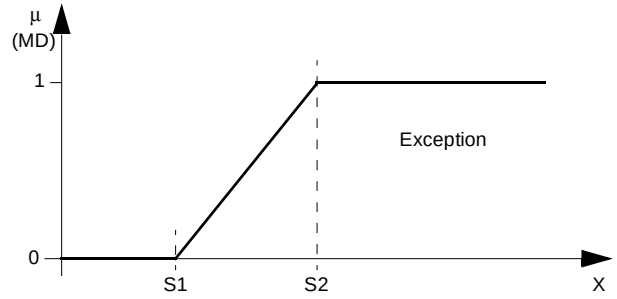
Triangle: S1 < S2 < S4 and S3 ≤ S2



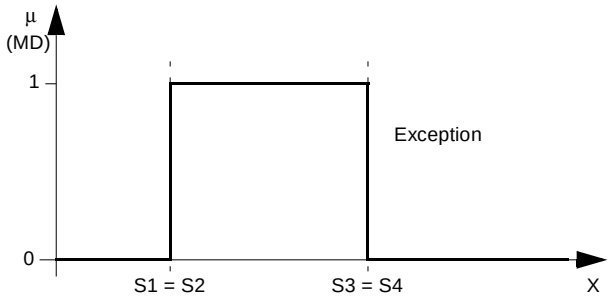
Falling ramp: S2 ≤ S3 < S4 and S1 > S2



Rising ramp: S1 < S2 ≤ S3 and S4 < S3



Rectangle: S1 = S2 < S3 = S4



Singleton: S1 = S2 = S3 = S4

Runtime errors

Runtime errors

An error message results when

- the number of support points > is 4 (E_EFB_TO MANY_INPUTS).

FUZ_SUM_***: Fuzzy Sum

11

Overview

Introduction

This chapter describes the block FUZ_SUM_***.

What's in this Chapter?

This chapter contains the following topics:

Brief Description

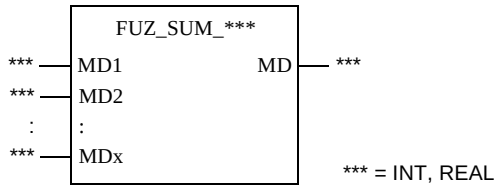
Function description

The function forms the limited sum (output MD) of the membership degrees (inputs MD1 ... MDx). The range of the inputs and the output for data type INT is 0 ... 10 000 and for data type REAL is 0 ... 1.
The number of inputs can be increased.
The data types of all input values and that of the output value must be the same.
There is a special function block available for the processing of each of the different data types.
As additional parameter EN and ENO can be projected.

Description

Symbol

Block description:



Formula

Block Formula:

$$\text{REAL: } MD = \text{Min}\left\{1, \sum_{i=1}^{n} MDi\right\}$$

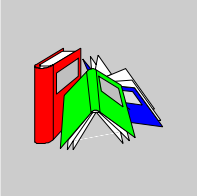
$$\text{INT: } MD = \text{Min}\left\{10000, \sum_{i=1}^{n} MDi\right\}$$

Parameter description

Description of the block parameter:

Parameter	Data type	Meaning
MD1	INT, REAL	1st Membership Degree
MD2	INT, REAL	2nd Membership Degree
:	:	:
MDx	INT, REAL	x. Membership Degree
MD	INT, REAL	Output limited sum

Glossary



A

active window The window, which is currently selected. Only one window can be active at any one given time. When a window is active, the heading changes color, in order to distinguish it from other windows. Unselected windows are inactive.

Actual parameter Currently connected Input/Output parameters.

Addresses (Direct) addresses are memory areas on the PLC. These are found in the State RAM and can be assigned input/output modules.
The display/input 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 data type "Analog Input" and is used for processing analog values. The 3x References of the configured analog input module, which is specified in the I/O component list is automatically assigned the data type and should therefore only be occupied by Unlocated variables.

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

ANY In the existing version "ANY" covers the elementary data types BOOL, BYTE, DINT, INT, REAL, UDINT, UINT, TIME and WORD and therefore derived data types.

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

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

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

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

ANY_REAL In the existing version "ANY_REAL" covers the data type REAL.

Application window The window, which contains the working area, the menu bar and the tool bar for the application. The name of the application appears in the heading. An application window can contain several document windows. In Concept the application window corresponds to a Project.

Argument Synonymous with Actual parameters.

ASCII mode American Standard Code for Information Interchange. The ASCII mode is used for communication 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 occupies a motherboard (requires SA85 driver) with two slots for PC104 daughter boards. From this, a PC104 daughter board is used as a CPU and the others for INTERBUS control.

B

Back up data file (Concept EFB) The back up file is a copy of the last Source files. The name of this back up file is "backup??.c" (it is accepted that there are no more than 100 copies of the source files. The first back up file is called "backup00.c". If changes have been made on the Definition file, which do not create any changes to the interface in the EFB, there is no need to create a back up file by editing the source files (**Objects** → **Source**). If a back up file can be assigned, the name of the source file can be given.

Base 16 literals

Base 16 literals function as the input of whole number values in the hexadecimal system. The base must be denoted by the prefix 16#. The values may not be preceded by signs (+/-). Single underline signs (_) between figures are not significant.

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

Base 8 literal

Base 8 literals function as the input of whole number values in the octal system. The base must be denoted by the prefix 3.63kg. The values may not be preceded by signs (+/-). Single underline signs (_) between figures are not significant.

Example
8#3_1111 or 8#377 (decimal 255)
8#34_1111 or 8#340 (decimal 224)

Basis 2 literals

Base 2 literals function as the input of whole number values in the dual system. The base must be denoted by the prefix 0.91kg. The values may not be preceded by signs (+/-). Single underline signs (_) between figures are not significant.

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

Binary connections

Connections between outputs and inputs of FFBs of data type BOOL.

Bit sequence

A data element, which is made up from one or more bits.

BOOL

BOOL stands for the data type "Boolean". The length of the data elements is 1 bit (in the memory contained in 1 byte). The range of values for variables of this type is 0 (FALSE) and 1 (TRUE).

Bridge

A bridge serves to connect networks. It enables communication between nodes on the two networks. Each network has its own token rotation sequence – the token is not deployed via bridges.

BYTE

BYTE stands for the data type "Bit sequence 8". The input appears as Base 2 literal, Base 8 literal or Base 1 16 literal. The length of the data element is 8 bit. A numerical range of values cannot be assigned to this data type.

C

Cache The cache is a temporary memory for cut or copied objects. These objects can be inserted into sections. The old content in the cache is overwritten for each new Cut or Copy.

Call up The operation, by which the execution of an operation is initiated.

Coil A coil is a LD element, which transfers (without alteration) the status of the horizontal link on the left side to the horizontal link on the right side. In this way, the status is saved in the associated Variable/ direct address.

Compact format (4:1) The first figure (the Reference) is separated from the following address with a colon (:), where the leading zero are not entered in the address.

Connection A check or flow of data connection between graphic objects (e.g. steps in the SFC editor, Function blocks in the FBD editor) within a section, is graphically shown as a line.

Constants Constants are Unlocated variables, which are assigned a value that cannot be altered from the program logic (write protected).

Contact A contact is a LD element, which transfers a horizontal connection status onto the right side. This status is from the Boolean AND- operation of the horizontal connection status on the left side with the status of the associated Variables/direct Address. A contact does not alter the value of the associated variables/direct address.

D

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

Data types

The overview shows the hierarchy of data types, as they are used with inputs and outputs of Functions and Function blocks. Generic data types are denoted by 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 station

With a Distributed Control Processor (D908) a remote network can be set up with a parent PLC. When using a D908 with remote PLC, the parent PLC views the remote PLC as a remote I/O station. The D908 and the remote PLC communicate via the system bus, which results in high performance, with minimum effect on the cycle time. The data exchange between the D908 and the parent PLC takes place at 1.5 Megabits per second via the remote I/O bus. A parent PLC can support up to 31 (Address 2-32) D908 processors.

DDE (Dynamic Data Exchange)

The DDE interface enables a dynamic data exchange between two programs under Windows. The DDE interface can be used in the extended monitor to call up its 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 onto the PLC via the server. Data can therefore be altered directly in the PLC, while it monitors and analyzes the results. When using this interface, the user is able to make their own "Graphic-Tool", "Face Plate" or "Tuning Tool", and integrate this into the system. The tools can be written in any DDE supporting language, e.g. Visual Basic and Visual-C++. The tools are called up, when the one of the buttons in the dialog box extended monitor uses Concept Graphic Tool: Signals of a projection can be displayed as timing diagrams via the DDE connection between Concept and Concept Graphic Tool.

Decentral Network (DIO)

A remote programming in Modbus Plus network enables maximum data transfer performance and no specific requests on the links. The programming of a remote net is easy. To set up the net, no additional ladder diagram logic is needed. Via corresponding entries into the Peer Cop processor all data transfer requests are met.

Declaration

Mechanism for determining the definition of a Language element. A declaration normally covers the connection of an Identifier with a language element and the assignment of attributes such as Data types and algorithms.

Definition data file (Concept EFB)	The definition file contains general descriptive information about the selected FFB and its formal parameters.
Derived data type	Derived data types are types of data, which are derived from the Elementary data types and/or other derived data types. The definition of the derived data types appears in the data type editor in Concept. Distinctions are made between global data types and local data types.
Derived Function Block (DFB)	A derived function block represents the Call up of a derived function block type. Details of the graphic form of call up can be found in the definition " Function block (Item)". Contrary to calling up EFB types, calling up DFB types is denoted by double vertical lines on the left and right side of the rectangular block symbol. The body of a derived function block type is designed using FBD language, but only in the current version of the programming system. Other IEC languages cannot yet be used for defining DFB types, nor can derived functions be defined in the current version. Distinctions are made between local and global DFBs.
DINT	DINT stands for the data type "double integer". The input appears as Integer literal, Base 2 literal, Base 8 literal or Base 16 literal. The length of the data element is 32 bit. The range of values for variables of this data type is from –2 exp (31) to 2 exp (31) –1.
Direct display	A method of displaying variables in the PLC program, from which the assignment of configured memory can be directly and indirectly derived from the physical memory.
Document window	A window within an Application window. Several document windows can be opened at the same time in an application window. However, only one document window can be active. Document windows in Concept are, for example, sections, the message window, the reference data editor and the PLC configuration.
Dummy	An empty data file, which consists of a text header with general file information, i.e. author, date of creation, EFB identifier etc. The user must complete this dummy file with additional entries.
DX Zoom	This property enables connection to a programming object to observe and, if necessary, change its data value.

E

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 bodies, for example, cannot be modified with the DFB Editor (Concept-DFB). EFB types are programmed in "C" and mounted via Libraries in precompiled form.

EN / ENO (Enable / Error display)
If the value of EN is "0" when the FFB is called up, the algorithms defined by the FFB are not executed and all outputs contain the previous value. The value of ENO is automatically set to "0" in this case. If the value of EN is "1" when the FFB is called up, the algorithms defined by the FFB are executed. After the error free execution of the algorithms, the ENO value is automatically set to "1". If an error occurs during the execution of the algorithm, ENO is automatically set to "0". The output behavior of the FFB depends whether the FFBs are called up without EN/ENO or with EN=1. If the EN/ENO display is enabled, the EN input must be active. Otherwise, the FFB is not executed. The projection of EN and ENO is enabled/disabled in the block properties dialog box. The dialog box is called up via the menu commands **Objects** → **Properties...** or via a double click on the FFB.

Error
When processing a FFB or a Step an error is detected (e.g. unauthorized input value or a time error), an error message appears, which can be viewed with the menu command **Online** → **Event display...** . With FFBs the ENO output is set to "0".

Evaluation
The process, by which a value for a Function or for the outputs of a Function block during the Program execution is transmitted.

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
Variables, one of which is assigned, with the assistance of the key word ARRAY (field), a defined Derived data type. A field is a collection of data elements of the same Data type.

FIR filter
Finite Impulse Response Filter

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 exactly supplies a data element when executing. A function has no internal status information. Multiple call ups of the same function with the same input parameter values always supply the same output values. Details of the graphic form of function call up can be found in the definition " Function block (Item)". In contrast to the call up of function blocks, the function call ups only have one unnamed output, whose name is the name of the function itself. In FBD each call up is denoted by a unique number over the graphic block; this number is automatically generated and cannot be altered.

Function block (item) (FB)

A function block is a Program organization unit, which correspondingly calculates the functionality values, defined in the function block type description, for the output and internal variables, when it is called up as a certain item. All output values and internal variables of a certain function block item remain as a call up of the function block until the next. Multiple call up of the same function block item with the same arguments (Input parameter values) supply generally supply the same output value(s).
Each function block item is displayed graphically by a rectangular block symbol. The name of the function block type is located on the top center within the rectangle. The name of the function block item is located also at the top, but on the outside of the rectangle. An instance is automatically generated when creating, which can however be altered manually, if required. Inputs are displayed on the left side and outputs on the right of the block. The names of the formal input/output parameters are displayed within the rectangle in the corresponding places.
The above description of the graphic presentation is principally applicable to Function call ups and to DFB call ups. Differences are described 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, subdivided into input, output and internal variables, 2. A set of operations, which is used with the elements of the data structure, when a function block type instance is called up. This set of operations can be formulated either in one of the IEC languages (DFB type) or in "C" (EFB type). A function block type can be instanced (called up) several times.

Function counter The function counter serves as a unique identifier for the function in a Program or DFB. The function counter cannot be edited and is automatically assigned. The function counter always has the structure: .n.m

n = Section number (number running)
m = Number of the FFB object in the section (number running)

G

Generic data type A Data type, which stands in for several other data types.

Generic literal If the Data type of a literal is not relevant, simply enter the value for the literal. In this case Concept automatically assigns the literal to a suitable data type.

Global derived data types Global Derived data types are available in every Concept project and are contained in the DFB directory directly under the Concept directory.

Global DFBs Global DFBs are available in every Concept project and are contained in the DFB directory directly under the Concept directory.

Global macros Global Macros are available in every Concept project and are contained in the DFB directory directly under the Concept directory.

Groups (EFBs) Some EFB libraries (e.g. the IEC library) are subdivided into groups. This facilitates the search for FFBs, especially in extensive libraries.

I

I/O component list The I/O and expert assemblies of the various CPUs are configured in the I/O component list.

IEC 61131-3 International norm: Programmable controllers – part 3: Programming languages.

IEC format (QW1)

In the place of the address stands an IEC identifier, followed by a five figure address:

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

IEC name conventions (identifier)

An identifier is a sequence of letters, figures, and underscores, which must start with a letter or underscores (e.g. name of a function block type, of an item or section). Letters from national sets of characters (e.g. ö,ü, é, õ) can be used, taken from project and DFB names.
Underscores are significant in identifiers; e.g. "A_BCD" and "AB_CD" are interpreted as different identifiers. Several leading and multiple underscores are not authorized consecutively.
Identifiers are not permitted to contain space characters. Upper and/or lower case is not significant; e.g. "ABCD" and "abcd" are interpreted as the same identifier.
Identifiers are not permitted to be Key words.

IIR filter

Infinite Impulse Response Filter

Initial step (starting step)

The first step in a chain. In each chain, an initial step must be defined. The chain is started with the initial step when first called up.

Initial value

The allocated value of one of the variables when starting the program. The value assignment appears in the form of a Literal.

Input bits (1x references)

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

Note: The x, which comes after the first figure of the reference type, represents a five figure storage location in the application data store, i.e. if the reference 100201 signifies an input bit in the address 201 of the State RAM.

Input parameters (Input)

When calling up a FFB the associated Argument is transferred.

Input words (3x references)

An input word contains information, which come from an external source and are represented by a 16 bit figure. 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 comes after the first figure of the reference type, represents a five figure storage location in the user data store, i.e. if the reference 300201 signifies a 16 bit input word in the address 201 of the State RAM.

Instantiation

The generation of an Item.

Instruction (IL)

Instructions are "commands" of the IL programming language. Each operation begins on a new line and is succeeded by an operator (with modifier if needed) and, if necessary for each relevant operation, by one or more operands. If several operands are used, they are separated by commas. A tag can stand before the instruction, which is followed by a colon. The commentary must, if available, be the last element in the line.

Instruction (LL984)

When programming electric controllers, the task of implementing operational coded instructions in the form of picture objects, which are divided into recognizable contact forms, must be executed. The designed program objects are, on the user level, converted to computer useable OP codes during the loading process. The OP codes are deciphered in the CPU and processed by the controller's firmware functions so that the desired controller is implemented.

Instruction list (IL)

IL is a text language according to IEC 1131, in which operations, e.g. conditional/ unconditional call up of Function blocks and Functions, conditional/unconditional jumps etc. are displayed through instructions.

INT

INT stands for the data type "whole number". The input appears as Integer literal, Base 2 literal, Base 8 literal or Base 16 literal. The length of the data element is 16 bit. The range of values for variables of this data type is from –2 exp (15) to 2 exp (15) –1.

Integer literals

Integer literals function as the input of whole number values in the decimal system. The values may be preceded by the signs (+/-). Single underline signs (_) between figures are not significant.

Example
-12, 0, 123_456, +986

INTERBUS (PCP)

To use the INTERBUS PCP channel and the INTERBUS process data preprocessing (PDP), the new I/O station type INTERBUS (PCP) is led into the Concept configurator. This I/O station type is assigned fixed to the INTERBUS connection module 180-CRP-660-01.
The 180-CRP-660-01 differs from the 180-CRP-660-00 only by a clearly larger I/O area in the state RAM of the controller.

Item name

An Identifier, which belongs to a certain Function block item. The item name serves as a unique identifier for the function block in a program organization unit. The item name is automatically generated, but can be edited. The item name must be unique throughout the Program organization unit, and no distinction is made between upper/lower case. If the given name already exists, a warning is given and another name must be selected. The item name must conform to the IEC name conventions, otherwise an error message appears. The automatically generated instance name always has the structure: FBI_n_m

FBI = Function block item
n = Section number (number running)
m = Number of the FFB object in the section (number running)

J

Jump

Element of the SFC language. Jumps are used to jump over areas of the chain.

K

Key words

Key words are unique combinations of figures, which are used as special syntactic elements, as is defined in appendix B of the IEC 1131-3. All key words, which are used in the IEC 1131-3 and in Concept, are listed in appendix C of the IEC 1131-3. These listed keywords cannot be used for any other purpose, i.e. not as variable names, section names, item names etc.

L

Ladder Diagram (LD)

Ladder Diagram is a graphic programming language according to IEC1131, which optically orientates itself to the "rung" of a relay ladder diagram.

Ladder Logic 984 (LL)

In the terms Ladder Logic and Ladder Diagram, the word Ladder refers to execution. In contrast to a diagram, a ladder logic is used by engineers to draw up a circuit (with assistance from electrical symbols),which should chart the cycle of events and not the existing wires, which connect the parts together. A usual user interface for controlling the action by automated devices permits ladder logic interfaces, so that when implementing a control system, engineers do not have to learn any new programming languages, with which they are not conversant.
The structure of the actual ladder logic enables electrical elements to be linked in a way that generates a control output, which is dependant upon a configured flow of power through the electrical objects used, which displays the previously demanded condition of a physical electric appliance.
In simple form, the user interface is one of the video displays used by the PLC programming application, which establishes a vertical and horizontal grid, in which the programming objects are arranged. The logic is powered from the left side of the grid, and by connecting activated objects the electricity flows from left to right.

Landscape format

Landscape format means that the page is wider than it is long when looking at the printed text.

Language element

Each basic element in one of the IEC programming languages, e.g. a Step in SFC, a Function block item in FBD or the Start value of a variable.

Library

Collection of software objects, which are provided for reuse when programming new projects, or even when building new libraries. Examples are the Elementary function block types libraries.
EFB libraries can be subdivided into Groups.

Literals

Literals serve to directly supply values to inputs of FFBs, transition conditions etc. These values cannot be overwritten by the program logic (write protected). In this way, generic and standardized literals are differentiated.
Furthermore literals serve to assign a Constant a value or a Variable an Initial value. The input appears 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 its local DFBs and are contained in the DFB directory under the project directory.

Local DFBs

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

Local link

The local network link is the network, which links the local nodes with other nodes either directly or via a bus amplifier.

Local macros

Local Macros are only available in a single Concept project and are contained in the DFB directory under the project directory.

Local network nodes The local node is the one, which is projected evenly.

Located variable Located variables are assigned a state RAM address (reference addresses 0x,1x, 3x, 4x). The value of these variables is saved in the state RAM and can be altered online with the reference data editor. These variables can be addressed by symbolic names or the reference addresses.

Collective PLC inputs and outputs are connected to the state RAM. The program access to the peripheral signals, which are connected to the PLC, appears only via located variables. PLC access from external sides via Modbus or Modbus plus interfaces, i.e. from visualizing systems, are likewise possible via located variables.

M

Macro Macros are created with help from the software Concept DFB.
Macros function to duplicate frequently used sections and networks (including the logic, variables, and variable declaration).
Distinctions are made between local and global macros.

Macros have the following properties:
- Macros can only be created in the programming languages FBD and LD.
- Macros only contain one single section.
- Macros can contain any complex section.
- From a program technical point of view, there is no differentiation between an instanced macro, i.e. a macro inserted into a section, and a conventionally created macro.
- Calling up DFBs in a macro
- Variable declaration
- Use of macro-own data structures
- Automatic acceptance of the variables declared in the macro
- Initial value for variables
- Multiple instancing of a macro in the whole program with different variables
- The section name, the variable name and the data structure name can contain up to 10 different exchange markings (@0 to @9).

MMI Man Machine Interface

Multi element variables Variables, one of which is assigned a Derived data type defined with STRUCT or ARRAY.
Distinctions are made between Field variables and structured variables.

N

Network A network is the connection of devices to a common data path, which communicate with each other via a common protocol.

Network node A node is a device with an address (164) on the Modbus Plus network.

Node address The node address serves a unique identifier for the network in the routing path. The address is set directly on the node, e.g. with a rotary switch on the back of the module.

O

Operand An operand is a Literal, a Variable, a Function call up or an Expression.

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

Output parameters (Output) A parameter, with which the result(s) of the Evaluation of a FFB are returned.

Output/discretes (0x references) An output/marker bit can be used to control real output data via an output unit of the control system, or to define one or more outputs in the state RAM. Note: The x, which comes after the first figure of the reference type, represents a five figure storage location in the application data store, i.e. if the reference 000201 signifies an output or marker bit in the address 201 of 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 also to send data from the CPU to an output unit in the control system. Note: The x, which comes after the first figure of the reference type, represents a five figure storage location in the application data store, i.e. if the reference 400201 signifies a 16 bit output or marker word in the address 201 of the State RAM.

P

Peer processor The peer processor processes the token run and the flow of data between the Modbus Plus network and the PLC application logic.

PLC Programmable controller

Program The uppermost Program organization unit. A program is closed and loaded onto a single PLC.

Program cycle A program cycle consists of reading in the inputs, processing the program logic and the output of the outputs.

Program organization unit A Function, a Function block, or a Program. This term can refer to either a Type or an Item.

Programming device Hardware and software, which supports programming, configuring, testing, implementing and error searching in PLC applications as well as in remote system applications, to enable source documentation and archiving. The programming device could also be used for process visualization.

Programming redundancy system (Hot Standby) A redundancy system consists of two identically configured PLC devices, which communicate with each other via redundancy processors. In the case of the primary PLC failing, the secondary PLC takes over the control checks. Under normal conditions the secondary PLC does not take over any controlling functions, but instead checks the status information, to detect mistakes.

Project General identification of the uppermost level of a software tree structure, which specifies the parent project name of a PLC application. After specifying the project name, the system configuration and control program can be saved under this name. All data, which results during the creation of the configuration and the program, belongs to this parent project for this special automation.
General identification for the complete set of programming and configuring information in the Project data bank, which displays the source code that describes the automation of a system.

Project data bank The data bank in the Programming device, which contains the projection information for a Project.

Prototype data file (Concept EFB)

The prototype data file contains all prototypes of the assigned functions. Further, if available, a type definition of the internal

R

REAL

REAL stands for the data type "real". The input appears as Real literal or as Real literal with exponent. The length of the data element is 32 bit. The value range for variables of this data type reaches from 8.43E-37 to 3.36E+38.

Note: Depending on the mathematic processor type of the CPU, various areas within this valid value range cannot be represented. This is valid for values nearing ZERO and for values nearing INFINITY. In these cases, a number value is not shown in animation, instead NAN (**N**ot **A N**umber) oder INF (**INF**inite).

Real literal

Real literals function as the input of real values in the decimal system. Real literals are denoted by the input of the decimal point. The values may be preceded by the signs (+/-). Single underline signs (_) between figures are not significant.

Example
-12.0, 0.0, +0.456, 3.14159_26

Real literal with exponent

Real literals with exponent function as the input of real values in the decimal system. Real literals with exponent are denoted by the input of the decimal point. The exponent sets the key potency, by which the preceding number is multiplied to get to the value to be displayed. The basis may be preceded by a negative sign (-). The exponent may be preceded by a positive or negative sign (+/-). Single underline signs (_) between figures are not significant. (Only between numbers, not before or after the decimal poiont 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

Each direct address is a reference, which starts with an ID, specifying whether it concerns an input or an output and whether it concerns a bit or a word. References, which start with the code 6, display the register in the extended memory of the state RAM.
0x area = Discrete outputs
1x area = Input bits
3x area = Input words
4x area = Output bits/Marker words
6x area = Register in the extended memory

Note: The x, which comes after the first figure of each reference type, represents a five figure storage location in the application data store, i.e. if the reference 400201 signifies a 16 bit output or marker word in the address 201 of the State RAM.

Register in the extended memory (6x reference)

6x references are marker words in the extended memory of the PLC. Only LL984 user programs and CPU 213 04 or CPU 424 02 can be used.

RIO (Remote I/O)

Remote I/O provides a physical location of the I/O coordinate setting device in relation to the processor to be controlled. Remote inputs/outputs are connected to the consumer control via a wired communication cable.

RP (PROFIBUS)

RP = Remote Peripheral

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.

Rum-time error

Error, which occurs during program processing on the PLC, with SFC objects (i.e. steps) or FFBs. These are, for example, over-runs of value ranges with figures, or time errors with steps.

S

SA85 module

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

Section

A section can be used, for example, to describe the functioning method of a technological unit, such as a motor.
A Program or DFB consist of one or more sections. Sections can be programmed with the IEC programming languages FBD and SFC. Only one of the named programming languages can be used within a section.
Each section has its own Document window in Concept. For reasons of clarity, it is recommended to subdivide a very large section into several small ones. The scroll bar serves to assist scrolling in a section.

Separator format (4:00001)

The first figure (the Reference) is separated from the ensuing five figure address by a colon (:).

Sequence language (SFC)

The SFC Language elements enable the subdivision of a PLC program organizational unit in a number of Steps and Transitions, which are connected horizontally by aligned Connections. A number of actions belong to each step, and a transition condition is linked to a transition.

Serial ports

With serial ports (COM) the information is transferred bit by bit.

Source code data file (Concept EFB)

The source code data file is a usual C++ source file. After execution of the menu command **Library** → **Generate data files** this file contains an EFB code framework, in which a specific code must be entered for the selected EFB. To do this, click on the menu command **Objects** → **Source**.

Standard format (400001)

The five figure address is located directly after the first figure (the reference).

Standardized literals

If the data type for the literal is to be automatically determined, use the following construction: 'Data type name'#'Literal value'.

Example
INT#15 (Data type: Integer, value: 15),
BYTE#00001111 (data type: Byte, value: 00001111)
REAL#23.0 (Data type: Real, value: 23.0)

For the assignment of REAL data types, there is also the possibility to enter the value in the following way: 23.0.
Entering a comma will automatically assign the data type REAL.

State RAM

The state RAM is the storage for all sizes, which are addressed in the user program via References (Direct display). For example, input bits, discretes, input words, and discrete words are located in the state RAM.

Statement (ST) Instructions are "commands" of the ST programming language. Instructions must be terminated with semicolons. Several instructions (separated by semi-colons) can occupy the same line.

Status bits There is a status bit for every node with a global input or specific input/output of Peer Cop data. If a defined group of data was successfully transferred within the set time out, the corresponding status bit is set to 1. Alternatively, this bit is set to 0 and all data belonging to this group (of 0) is deleted.

Step SFC Language element: Situations, in which the Program behavior follows in relation to the inputs and outputs of the same operations, which are defined by the associated actions of the step.

Step name The step name functions as the unique flag of a step in a Program organization unit. The step name is automatically generated, but can be edited. The step name must be unique throughout the whole program organization unit, otherwise an Error message appears.
The automatically generated step name always has the structure: S_n_m

S = Step
n = Section number (number running)
m = Number of steps in the section (number running)

Structured text (ST) ST is a text language according to IEC 1131, in which operations, e.g. call up of Function blocks and Functions, conditional execution of instructions, repetition of instructions etc. are displayed through instructions.

Structured variables Variables, one of which is assigned a Derived data type defined with STRUCT (structure).
A structure is a collection of data elements with generally differing data types (Elementary data types and/or derived data types).

SY/MAX In Quantum control devices, Concept closes the mounting on the I/O population SY/MAX I/O modules for RIO control via the Quantum PLC with on. The SY/MAX remote subrack 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 performed when highlighting and including in the I/O population of the Concept configuration.

Symbol (Icon) Graphic display of various objects in Windows, e.g. drives, user programs and Document windows.

T

Template data file (Concept EFB)
The template data file is an ASCII data file with a layout information for the Concept FBD editor, and the parameters for code generation.

TIME
TIME stands for the data type "Time span". The input appears as Time span literal. The length of the data element is 32 bit. The value range for variables of this type stretches from 0 to 2exp(32)-1. The unit for the data type TIME is 1 ms.

Time span literals
Permitted units for time spans (TIME) are days (D), hours (H), minutes (M), seconds (S) and milliseconds (MS) or a combination thereof. The time span must be denoted by the prefix t#, T#, time# or TIME#. An "overrun" of the highest ranking unit is permitted, i.e. the input T#25H15M is permitted.

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 property of the transfer rights via a single node. The token runs through the node in a circulating (rising) address sequence. All nodes track the Token run through and can contain all possible data sent with it.

Traffic Cop
The Traffic Cop is a component list, which is compiled from the user component list. The Traffic Cop is managed in the PLC and in addition contains the user component list e.g. Status information of the I/O stations and modules.

Transition
The condition with which the control of one or more Previous steps transfers to one or more ensuing steps along a directional Link.

U

UDEFB
User defined elementary functions/function blocks
Functions or Function blocks, which were created in the programming language C, and are available in Concept Libraries.

UDINT UDINT stands for the data type "unsigned double integer". The input appears as Integer literal, Base 2 literal, Base 8 literal or Base 16 literal. The length of the data element is 32 bit. The value range for variables of this type stretches from 0 to 2exp(32)-1.

UINT UINT stands for the data type "unsigned integer". The input appears as Integer literal, Base 2 literal, Base 8 literal or Base 16 literal. The length of the data element is 16 bit. The value range for variables of this type stretches from 0 to (2exp16)-1.

Unlocated variable Unlocated variables are not assigned any state RAM addresses. They therefore do not occupy any state RAM addresses. The value of these variables is saved in the system and can be altered with the reference data editor. These variables are only addressed by symbolic names.

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

V

Variables Variables function as a data exchange within sections between several sections and between the Program and the PLC.
Variables consist of at least a variable name and a Data type.
Should a variable be assigned a direct Address (Reference), it is referred to as a Located variable. Should a variable not be assigned a direct address, it is referred to as an unlocated variable. If the variable is assigned a Derived data type, it is referred to as a Multi-element variable.
Otherwise there are Constants and Literals.

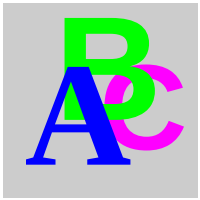
Vertical format Vertical format means that the page is higher than it is wide when looking at the printed text.

W

Warning When processing a FFB or a Step a critical status is detected (e.g. critical input value or a time out), a warning appears, which can be viewed with the menu command **Online** → **Event display...** . With FFBs the ENO output remains at "1".

WORD	WORD stands for the data type "Bit sequence 16". The input appears as Base 2 literal, Base 8 literal or Base 1 16 literal. The length of the data element is 16 bit. A numerical range of values cannot be assigned to this data type.

Index



I

L

M

O

P

R

S