Unity Pro
Standard
Block Library

September 2004

33002519.01

a brand of Schneider Electric

Telemecanique

Table of Contents

About the Book

At a Glance

Document Scope

This document describes the functions and function blocks of the Standard library. This document is valid for Unity Pro Version 2.0.

Validity Note

The data and illustrations found in this document are not binding. We reserve the right to modify our products in line with our policy of continuous product development. The information in this document is subject to change without notice and should not be construed as a commitment by Schneider Electric.

Product Related Warnings

Schneider Electric assumes no responsibility for any errors that may appear in this document. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

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

When controllers are used for applications with technical safety requirements, please follow the relevant instructions.

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

Failure to observe this product related warning can result in injury or equipment damage.

User Comments

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

General information

I

Introduction

Overview

This section contains general information about the Standard library.

What's in this Part?

This part contains the following chapters:

Block types and their applications

1

Introduction

Overview

This chapter describes the different block types and their applications.

What's in this Chapter?

This chapter contains the following topics:

Block types

Block types

Different block types are used in Unity Pro. The general term for all block types is FFB.
There are the following types of block:
- Elementary Function (EF)
- Elementary Function Block (EFB)
- Derived Function Block (DFB)
- Procedure

Elementary Function

Elementary functions (EF) have no internal status.. If the input values are the same, the value at the output is the same for all executions of the function, e.g. the addition of two values gives the same result at every execution.
An elementary function is represented in the graphical languages (FDB and LD) as a block frame with inputs and an output. The inputs are always represented on the left and the outputs always on the right of the frame The name of the function, i.e. the function type, is shown in the center of the frame.
The number of inputs can be increased with some elementary functions.

Elementary function block

Elementary function blocks (EFB) have an internal status. If the inputs have the same values, the value on the output can have another value during the individual executions. For example, with a counter, the value on the output is incremented.
An elementary function block is represented in the graphical languages (FDB and LD) as a block frame with inputs and outputs. The inputs are always represented on the left and the outputs always on the right of the frame The name of the function block, i.e. the function block type, is shown in the center of the frame. The instance name is displayed above the frame.

Derived function block

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

Procedure

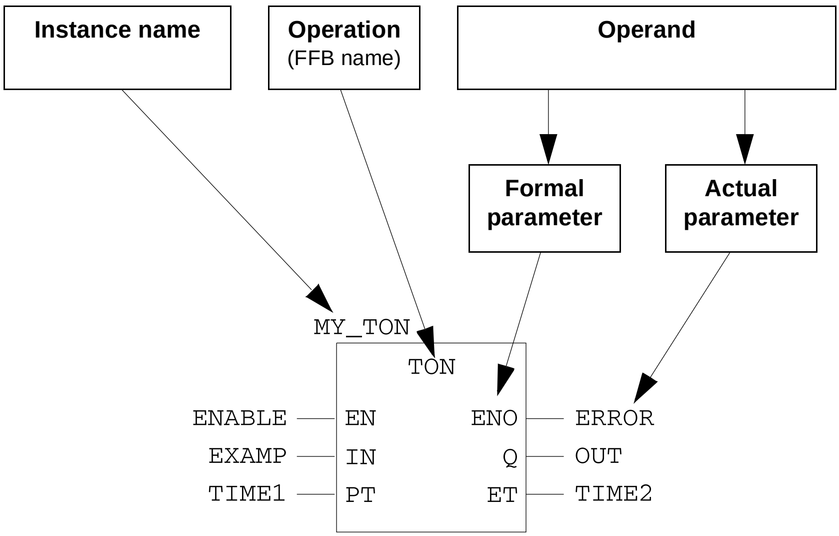
Procedures are technical functions.
The only difference from elementary functions is that procedures can have more than one output and they support variables of the `VAR_IN_OUT` data type.
Procedures do not return a value.
Procedures are a supplement to IEC 61131-3 and must be enabled explicitly.
There is no visual difference between procedures and elementary functions.

FFB Structure

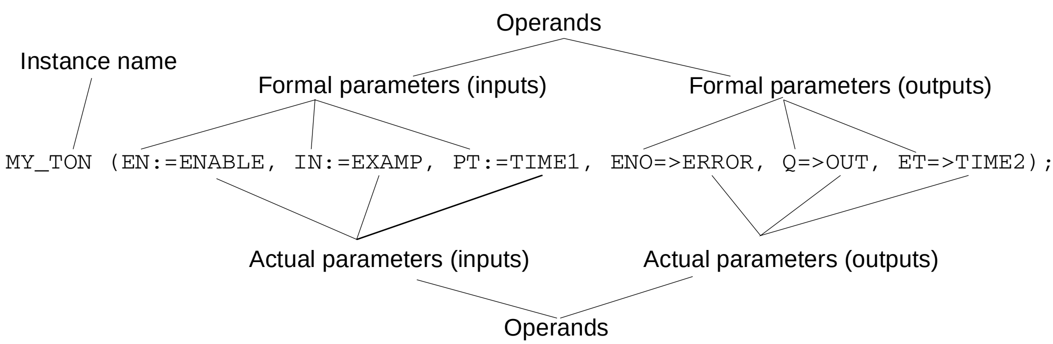
Structure

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

Call of a function block in the FBD programming language:



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



Operation

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

Operand

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

Formal/actual parameters

Inputs and outputs are required to give values to the FFB or to take values from the FFB. These are called formal parameters.
Objects are connected to the formal parameters which contain the current process states. These are called actual parameters.
During program runtime, the actual parameters are used to pass the process values to the FFB and output them after processing.
The data type of the actual parameters must match the data type of the input/output (formal parameters). The only exceptions are generic inputs/outputs, for which the data types are determined by the actual parameters. If all actual parameters are literals, the correct data type for the function block will be selected.

FFB Call in IL/ST

In text languages IL and ST, FFBs can be called in formal and in informal form. Details can be found in the *Reference manual*.
Example of a formal function call:
`out:=LIMIT (MN:=0, IN:=var1, MX:=5) ;`
Example of an informal function call:
`out:=LIMIT (0, var1, 5) ;`

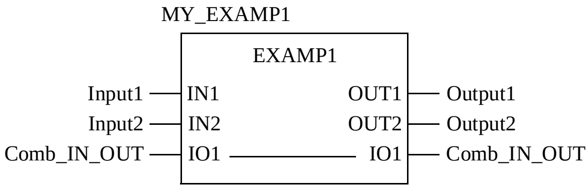
Note: Take note that the use of `EN` and `ENO` is only possible for formal calls.

`VAR_IN_OUT` variable

FFBs are often used to read a variable on an input (input variables), to process them and output the changed value of the **same** variable again (output variables). This is special case for an input/output variable and is also called `VAR_IN_OUT` variable.

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

Function block with `VAR_IN_OUT` variable in FBD:



Function block with `VAR_IN_OUT` variable in ST:

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

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

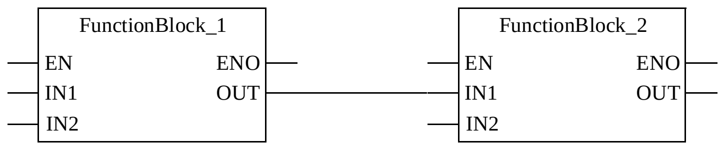
- `VAR_IN_OUT` variables absolutely must be assigned to as variable.
- The same variable/variable components must be assigned to the `VAR_IN_OUT` input and the `VAR_IN_OUT` output.
- In the graphic languages (FBD and LD), graphic connections cannot be made on `VAR_IN_OUT` inputs/outputs.
- Literals or constants cannot be assigned to `VAR_IN_OUT` inputs/outputs.
- In the graphic languages (FBD and LD), negations cannot be used on `VAR_IN_OUT` inputs/outputs.

EN and ENO

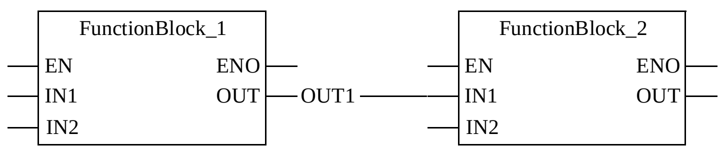
Description

An `EN` input and an `ENO` output can be configured for all FFBs.
If the value of `EN` is "0" when the FFB is called up, the algorithms defined by the FFB are not executed and `ENO` is set to "0".
If the value of `EN` is "1" when the FFB is called up, the algorithms defined by the FFB are executed. After the algorithms have been executed successfully, the value of `ENO` is set to "1". If an error occurs when executing these algorithms, `ENO` is set to "0".
If `ENO` is set to "0" (caused by `EN`=0 or an error during execution):

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

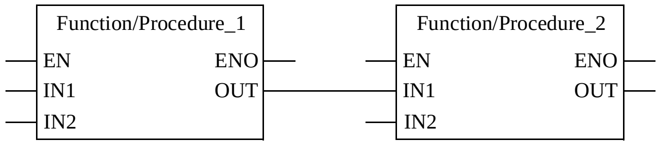
    

    If `EN` from `FunctionBlock_1` is set to "0", the output connection `OUT` from `FunctionBlock_1` retains the status it had in the last correctly executed cycle.
  - EN/ENO-handling with function blocks that have one variable and one connection as output parameters:

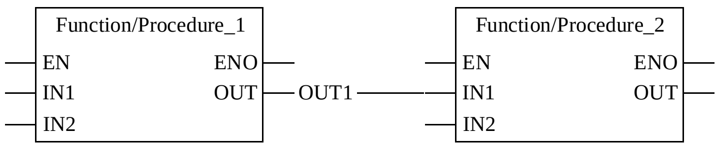
    

    If `EN` from `FunctionBlock_1` is set to "0", the output connection `OUT` from `FunctionBlock_1` retains the status it had in the last correctly executed cycle. The variable `OUT1` on the same pin, either retains its previous status or can be changed externally without influencing the connection. The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

- Functions/Procedures
  As defined in IEC61131-3, the outputs from deactivated functions (EN-input set to "0") is undefined. (The same applies for procedures.)
  Nevertheless here is an explanation of the output statuses for this case:
  - EN/ENO-handling with function/procedure blocks that (only) have one connection as output parameter:



    If EN from Function/Procedure_1 is set to "0", the output connection OUT from Function/Procedure_1 is also set to "0".
  - EN/ENO-handling with function/procedure blocks that have one variable and one connection as output parameters:



    If EN from Function/Procedure_1 is set to "0", the output connection OUT from Function/Procedure_1 is also set to "0", however the variable OUT1 on the same pin retains its previous value. In this way it is possible that the variable and the connection have different values.

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

Conditional/ Unconditional FFB Call

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

- EN connected
  conditional calls (the FFB is only processed if EN = 1)
- EN not used or set to TRUE
  unconditional calls (FFB is always processed)

Note for FBD

If the EN input is used, it must be connected to logic (conditional call) or permanently set to TRUE (unconditional call) because otherwise the FFB will never be processed.

Note for LD

In LD, each FFB must be connected with the left power rail using a Boolean input. Normally, the `EN` input is used for this purpose.
If the `EN` input is not connected to the left power rail, it cannot be used or it must be permanently set to TRUE because otherwise the FFB will never be processed.

Note for IL and ST

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

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

Assigning the variables to `ENO` must be done with the operator `=>`.
With an informal call, `EN` and `ENO` cannot be used.

Availability of the blocks on different hardware platforms

2

Availability of the block on the various hardware platforms

Introduction

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

Arrays

Availability of the blocks:

Block name	Block type	defined in IEC 61131-3	Premium	Quantum
ADD_***_***	EF	-	+	+
AND_***_***	EF	-	+	+
COPY_***_***	EF	-	+	+
DIV_***_***	EF	-	+	+
EQUAL_***	EF	-	+	+
FIND_EQ_***	EF	-	+	+
FIND_EQP_***	EF	-	+	+
FIND_GT_***	EF	-	+	+
FIND_LT_***	EF	-	+	+
LENGHT_***	EF	-	+	+
MAX_***	EF	-	+	+
MIN_***	EF	-	+	+
MOD_***_***	EF	-	+	+
MOVE_***_*** (direct assignment)	Procedure	-	+	+
MOVE_***_*** (conversion)	Procedure	-	+	+
MUL_***_***	EF	-	+	+
NOT_***	EF	-	+	+
OCCUR_***	EF	-	+	+
OR_***_***	EF	-	+	+
ROL_***	Procedure	-	+	+
ROR_***	Procedure	-	+	+
SORT_***	Procedure	-	+	+
SUB_***_***	EF	-	+	+
SUM_***	EF	-	+	+
SWAP_***	Procedure	-	+	+
XOR_***_***	EF	-	+	+
Legend:				
+	Yes			
-	No			

CLC_INT

Availability of the blocks:

Block name	Block type	defined in IEC 61131-3	Premium	Quantum
PID_INT	Procedure	-	+	+
PWM_INT	Procedure	-	+	+
SERVO_INT	Procedure	-	+	+
Legend:				
+	Yes			
-	No			

Comparison

Availability of the blocks:

Block name	Block type	defined in IEC 61131-3	Premium	Quantum
EQ	EF	+	+*	+
GE	EF	+	+*	+
GT	EF	+	+*	+
LE	EF	+	+*	+
LT	EF	+	+*	+
NE	EF	+	+*	+
Legend:				
+	Yes			
Premium: +*	The data types UINT and UDINT are only available on Premium TSX P 57 5••.			
-	No			

Date & Time

Availability of the blocks:

Block name	Block type	defined in IEC 61131-3	Premium	Quantum
ADD_***_TIME	EF	+	+	+
DIVTIME	EF	+	+*	+
MULTIME	EF	+	+*	+
SUB_***_***	EF	+	+	+
SUB_***_TIME	EF	+	+	+
Legend:				
+	Yes			
Premium: +*	The data types UINT and UDINT are only available on Premium TSX P 57 5••.			
-	No			

Logic

Availability of the blocks:

Block name	Block type	defined in IEC 61131-3	Premium	Quantum
AND	EF	+	+	+
F_TRIG	EFB	+	+	+
FE	EF	-	+	+
NOT	EF	+	+	+
OR	EF	+	+	+
R_TRIG	EFB	+	+	+
RE	EF	-	+	+
RESET	Procedure	-	+	+
ROL	EF	+	+	+
ROR	EF	+	+	+
RS	EFB	+	+	+
SET	Procedure	-	+	+
SHL	EF	+	+	+
SHR	EF	+	+	+
SR	EFB	+	+	+
TRIGGER	EFB	-	+	+
XOR	EF	+	+	+
Legend:				
+	Yes			
-	No			

Mathematics

Availability of the blocks:

Block name	Block type	defined in IEC 61131-3	Premium	Quantum
ABS	EF	+	+*	+
ACOS	EF	+	+	+
ADD	EF	+	+*	+
ADD_TIME	EF	+	+*	+
ASIN	EF	+	+	+
ATAN	EF	+	+	+
COS	EF	+	+	+
DEC	Procedure	-	+*	+
DIV	EF	+	+*	+
DIVMOD	Procedure	-	+*	+
EXP	EF	+	+	+
EXPT_REAL	EF	+	+*	+
INC	Procedure	-	+*	+
LN	EF	+	+	+
LOG	EF	+	+	+
MOD	EF	+	+*	+
MOVE	EF	+	+	+
MUL	EF	+	+*	+
NEG	EF	-	+*	+
SIGN	EF	-	+*	+
SIN	EF	+	+	+
SQRT	EF	DINT: - INT: - REAL: +	+	+
SUB	EF	+	+*	+
SUB_TIME	EF	+	+*	+
TAN	EF	+	+	+
Legend:				
+	Yes			

Block name	Block type	defined in IEC 61131-3	Premium	Quantum
Premium: +*	The data types UINT and UDINT are only available on Premium TSX P 57 5••.			
-	No			

Statistical

Availability of the blocks:

Block name	Block type	defined in IEC 61131-3	Premium	Quantum
AVE	EF	-	+*	+
LIMIT	EF	+	+*	+
LIMIT_IND	Procedure	-	+*	+
MAX	EF	+	+*	+
MIN	EF	+	+*	+
MUX	EF	+	+*	+
SEL	EF	+	+	+
Legend:				
+	Yes			
Premium: +*	The data types UINT and UDINT are only available on Premium TSX P 57 5••.			
-	No			

Strings

Availability of the blocks:

Block name	Block type	defined in IEC 61131-3	Premium	Quantum
CONCAT_STR	EF	-	+	+
DELETE_INT	EF	+	+	+
EQUAL_STR	EF	+	+	+
FIND_INT	EF	+	+	+
INSERT_INT	EF	+	+	+
LEFT_INT	EF	+	+	+
LEN_INT	EF	+	+	+
MID_INT	EF	+	+	+
REPLACE_INT	EF	+	+	+
RIGHT_INT	EF	+	+	+
Legend:				
+	Yes			
-	No			

Timers & Counter

Availability of the blocks:

Block name	**Block type**	**defined in IEC 61131-3**	**Premium**	**Quantum**
CTD	EFB	+	+	+
CTD_***	EFB	-	+*	+
CTU	EFB	+	+	+
CTU_***	EFB	-	+*	+
CTUD	EFB	+	+	+
CTUD_***	EFB	-	+*	+
TOF	EFB	+	+	+
TON	EFB	+	+	+
TP	EFB	+	+	+
Legend:				
+	Yes			
Premium: +*	The data types UINT and UDINT are only available on Premium TSX P 57 5••.			
-	No			

Type to type

Availability of the blocks:

Block name	Block type	defined in IEC 61131-3	Premium	Quantum
BCD_TO_INT	EF	+	+	+
BIT_TO_BYTE	EF	-	+	+
BIT_TO_WORD	EF	-	+	+
BOOL_TO_***	EF	+	+*	+
BYTE_AS_WORD	EF	-	+	+
BYTE_TO_BIT	Procedure	-	+	+
BYTE_TO_***	EF	+	+*	+
DATE_TO_STRING	EF	+	+	+
DBCD_TO_***	EF	-	+	+
DEG_TO_RAD	EF	-	+	+
DINT_AS_WORD	Procedure	-	+	+
DINT_TO_***	EF	+	+*	+
DINT_TO_DBCD	EF	-	+	+
DINT_TO_STRING	EF	+	+	+
DT_TO_STRING	EF	+	+	+
DWORD_TO_***	EF	+	+*	+
GRAY_TO_INT	EF	-	+	+
INT_AS_DINT	EF	-	+	+
INT_TO_***	EF	+	+*	+
INT_TO_BCD	EF	-	+	+
INT_TO_DBCD	EF	-	+	+
INT_TO_STRING	EF	+	+	+
RAD_TO_DEG	EF	-	+	+
REAL_AS_WORD	Procedure	-	+	+
REAL_TO_***	EF	+	+*	+
REAL_TO_STRING	EF	+	+	+
REAL_TRUNC_***	EF	+	+*	+
STRING_TO_***	EF	+	+	+
TIME_AS_WORD	Procedure	-	+	+
TIME_TO_***	EF	+	+*	+
TIME_TO_STRING	EF	+	+	+

Block name	Block type	defined in IEC 61131-3	Premium	Quantum
TOD_TO_STRING	EF	+	+	+
UDINT_AS_WORD	Procedure	-	+*	+
UDINT_TO_***	EF	+	+*	+
UINT_TO_***	EF	+	+*	+
WORD_AS_BYTE	Procedure	-	+	+
WORD_AS_DINT	EF	-	+	+
WORD_AS_REAL	Procedure	-	+	+
WORD_AS_TIME	EF	-	+	+
WORD_AS_UDINT	EF	-	+*	+
WORD_TO_BIT	Procedure	-	+	+
WORD_TO_***	EF	+	+	+
Legend:				
+	Yes			
-	No			
Premium: +*	The data types UINT and UDINT are only available on Premium TSX P 57 5••.			

Arrays

II

Introduction

Overview

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

What's in this Part?

This part contains the following chapters:

ADD_***_***: Addition of a number to elements of a table or addition of two tables

3

Description

Function description

The ADD_***_*** function adds a number to the elements of a table or adds two tables together.

The additional parameters EN and ENO can be configured.

Available functions

The available functions for adding a number to the elements of a table are as follows:

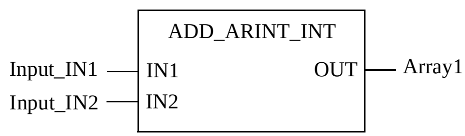
- ADD_ARINT_INT (addition of each element of a table of INTs to an INT).
- ADD_ARDINT_DINT (addition of each element of a table of DINTs to a DINT).

The available functions for adding the elements of one table to the elements of another table:

- ADD_ARINT (Sum of the respective elements of both INT tables).
- ADD_ARDINT (Sum of the respective elements of both DINT tables).

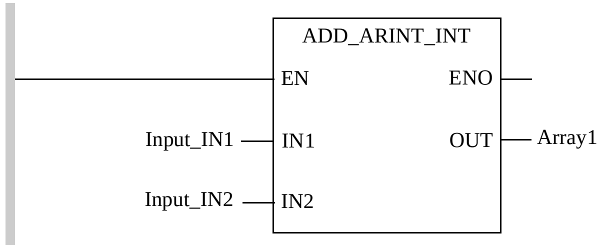
Representation in FBD

Representation applied to the sum of an integer and an integer table:



Representation in LD

Representation applied to the sum of an integer and an integer table:



Representation in IL

Representation applied to the sum of an integer and an integer table:

```
LD Input_IN1
ADD_ARINT_INT Input_IN2
ST Array1
```

Representation in ST

Representation applied to the sum of an integer and an integer table:

```
Array1 := ADD_ARINT_INT(Input_IN1,Input_IN2);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Input_IN1	ARRAY [n..m] OF INT, ARRAY [n..m] OF DINT	Input_IN1 is either a single or double integer, or a table of single or double integers, n and m are positive or negative integers or nil.
Input_IN2	INT, DINT, ARRAY [n..m] OF INT, ARRAY [n..m] OF DINT	Input_IN2 is either a single or double integer, or a table of single or double integers, n and m are positive or negative integers or nil.

The following table describes the output parameters:

Parameter	Type	Comment
Array1	ARRAY [n..m] OF INT ARRAY [n..m] OF DINT	According to the type of Input_IN1 and Input_IN2, each element of Array1 is the sum: ● of a single or double integer and the corresponding element of a table, ● the corresponding elements of two tables.

Runtime errors

The management of the **%S18** (See *Description of system bits %S15 to %S21, p. 448*) system bit is identical to that for operations on words or double words.

If an operation between two elements sets the **%S18** (See *Description of system bits %S15 to %S21, p. 448*) bit (overflow or division by zero), the result for this operation is incorrect, but the operation on the following elements is carried out correctly.

AND_***_***: Logical AND between tables and variables

4

Description

Function description

The AND_***_*** function carries out a logical AND (bit to bit) between:
- the elements of two tables,
- between a single type variable and the elements of a table,
- between the elements of a table and a single type variable.

Note: The result is always a table.

The additional parameters EN and ENO can be configured.

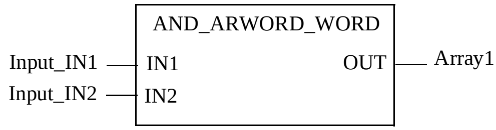
Available functions

The functions available in the general library are the following:
- AND_AREBOOL (logical AND of two EBOOL tables).
- AND_ARWORD (logical AND of two WORD tables).
- AND_ARWORD_WORD (logical AND of each element of a WORD table with a WORD).
- AND_ARDWORD_DWORD (logical AND of each element of a DWORD table with a DWORD).
- AND_ARDWORD (logical AND of two DWORD tables).

The functions available in the **Obsolete** library are the following:
- AND_ARINT_INT (logical AND of each element of an INT table with an INT).
- AND_ARDINT_DINT (logical AND of each element of a DINT table with a DINT).
- AND_ARINT (logical AND of each element of an INT table with each element corresponding to another INT table).
- AND_ARDINT (logical AND of each element of an INT table with each element corresponding to another DINT table).

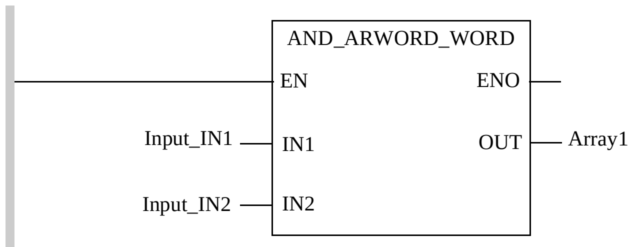
Representation in FBD

Representation applied to a 16-bit string and a 16-bit string table:



Representation in LD

Representation applied to a 16-bit string and a 16-bit string table:



Representation in IL

Representation applied to a 16-bit string and a 16-bit string table:

```
LD Input_IN1
AND_ARWORD_WORD Input_IN2
ST Array1
```

Representation in ST

Representation applied to a 16-bit string and a 16-bit string table:

```
Array1:= AND_ARWORD_WORD(Input_IN1,Input_IN2);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Input_IN1	ARRAY [n..m] OF EBOOL, WORD, ARRAY [n..m] OF WORD, DWORD, ARRAY [n..m] OF DWORD, INT, ARRAY [n..m] OF INT, DINT, ARRAY [n..m] OF DINT	n and m maximum and minimum limits.
Input_IN2	ARRAY [n..m] OF EBOOL, WORD, ARRAY [n..m] OF WORD, DWORD, ARRAY [n..m] OF DWORD, INT, ARRAY [n..m] OF INT, DINT, ARRAY [n..m] OF DINT	n and m are positive or negative integers or nil.

The following table describes the output parameters:

Parameter	Type	Comment
Array1	ARRAY [n..m] OF EBOOL, ARRAY [n..m] OF WORD, ARRAY [n..m] OF DWORD, ARRAY [n..m] OF INT, ARRAY [n..m] OF DINT	The elements of Array1 are the result of the logical AND (bit to bit) between Input_IN1 and Input_IN2, which can be respectively: • a table and a single variable, • a table and a table.

COPY_***_***: Copy on tables

5

Description

Function description

The `COPY_***_***` function copies a series of contiguous elements from one table into another table. The tables are of different or identical types and the target zone is fixed by the parameters of the function.

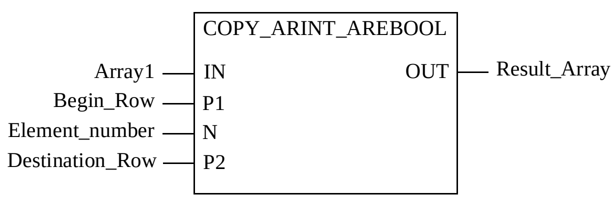
The additional parameters `EN` and `ENO` can be configured.

Available functions

The available functions are as follows:
- `COPY_AREBOOL_ARINT`,
- `COPY_AREBOOL_AREBOOL`,
- `COPY_AREBOOL_ARDINT`,
- `COPY_ARINT_AREBOOL`,
- `COPY_ARDINT_AREBOOL.`

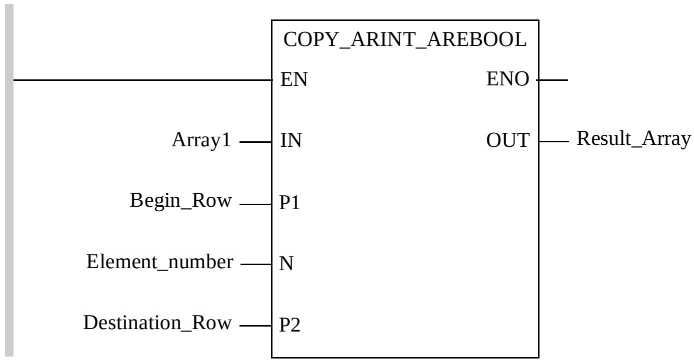
Representation in FBD

Representation applied to the assignment of a zone of an integer table to a zone of a bit table:



Representation in LD

Representation applied to the assignment of a zone of an integer table to a zone of a bit table:



Representation in IL

Representation applied to the assignment of a zone of an integer table to a zone of a bit table:

```
LD Array1
COPY_ARINT_AREBOOL Begin_Row, Element_Number, Destination_Row
ST Result_Array
```

Representation in ST

Representation applied to the assignment of a zone of an integer table to a zone of a bit table:

```
Result_Array := COPY_ARINT_AREBOOL(Array1, Begin_Row,
Element_Number, Destination_Row);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`Array1`	ARRAY [n..m] OF `EBOOL`, ARRAY [n..m] OF `INT`, ARRAY [n..m] OF `DINT`	n and m are positive or negative integers or nil.
`Begin_Row`	`INT`	Rank of first element to be copied from the table `Array1`. **Note**: The first element of the table has the rank 0.
`Element_Number`	`INT`	Number of elements to be copied from the table `Array1`.
`Destination_Row`	`INT`	Target rank in the table `Result_Array`.

The following table describes the output parameters:

Parameter	Type	Comment
`Result_Array`	ARRAY [n..m] OF `EBOOL`, ARRAY [n..m] OF `INT`, ARRAY [n..m] OF `DINT`	n and m are positive or negative integers or nil. Target table which contains the elements selected from `Array1`.

Note: if the number of elements to be extracted is greater than the remaining size, starting from the rank`Begin_Row`, the function extracts all the elements from`Begin_Row` to the last element of the table.
If the number of elements to be extracted is greater than the space available starting from the rank`Destination_Row`, the copy function runs to the last element of the table.
A negative value of `Begin_Row`, `Element_Number` and `Destination_Row` is interpreted as null.

DIV_***_***: Division of tables

6

Description

Function description

The DIV_***_*** function carries out the division:
- of a number by the elements of a table,
- of the elements of a table by a number,
- of the elements of a table by the respective elements of another table.

The additional parameters EN and ENO can be configured.

Available functions

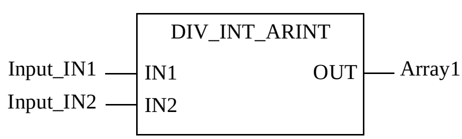
The available functions for division of a number by the elements of a table are as follows:
- DIV_INT_ARINT,
- DIV_DINT_ARDINT.

The available functions for division of the elements of a table by a number are as follows:
- DIV_ARINT_INT,
- DIV_ARDINT_DINT.

The available functions for division of the elements of a table by the respective elements of another table are as follows:
- DIV_ARINT,
- DIV_ARDINT.

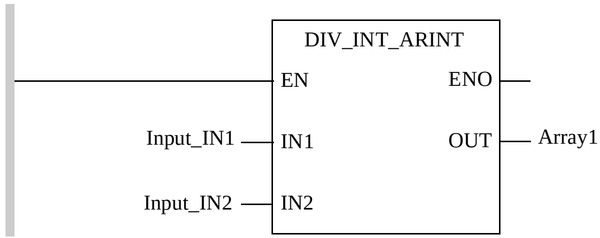
Representation in FBD

Representation applied to the division of an integer by the elements of an integer table:



Representation in LD

Representation applied to the division of an integer by the elements of an integer table:



Representation in IL

Representation applied to the division of an integer by the elements of an integer table:

```
LD Input_IN1
DIV_INT_ARINT Input_IN2
ST Array1
```

Representation in ST

Representation applied to the division of an integer by the elements of an integer table:

```
Array1:= DIV_INT_ARINT(Input_IN1,Input_IN2);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Input_IN1	INT, DINT, ARRAY [n..m] OF INT, ARRAY [n..m] OF DINT	Input_IN1 is either a single or double integer, or a table of single or double integers, n and m are positive or negative integers or nil.
Input_IN2	INT, DINT, ARRAY [n..m] OF INT, ARRAY [n..m] OF DINT	Input_IN2 is either a single or double integer, or a table of single or double integers, n and m are positive or negative integers or nil.

The following table describes the output parameters:

Parameter	Type	Comment
Array1	ARRAY [n..m] OF INT ARRAY [n..m] OF DINT	According to the type of Input_IN1 and Input_IN2, each element of Array1 is the division: ● of a single or double integer Input_IN1 by the corresponding element of the table Input_IN2 or else, ● of the elements of the table Input_IN1 by single or double integers Input_IN2 or else, ● of the elements of the table Input_IN1 by the respective elements of the table Input_IN2.

Runtime errors

The management of the **%S18** (See *Description of system bits %S15 to %S21, p. 448*) system bit is identical to that for operations on words or double words. In the case of division by zero, the value of the result is equal to the value of the numerator.

If an operation between two elements sets the **%S18** (See *Description of system bits %S15 to %S21, p. 448*) bit (overflow or division by zero), the result for this operation is incorrect, but the operation on the following elements is carried out correctly.

EQUAL_***: Comparison of two tables

7

Description

Function description

The EQUAL_*** function compares two tables element by element.

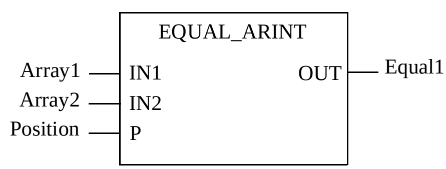
The additional parameters EN and ENO can be configured.

Available functions

The available functions are as follows:
- EQUAL_ARWORD,
- EQUAL_ARDWORD,
- EQUAL_ARINT,
- EQUAL_ARDINT,
- EQUAL_ARREAL.

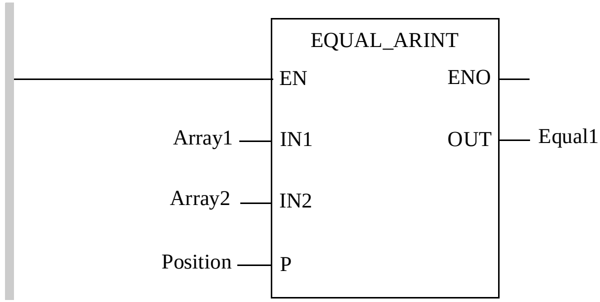
Representation in FBD

Representation applied to integer tables:



Representation in LD

Representation applied to integer tables:



Representation in IL

Representation applied to integer tables:

```
LD Array1
EQUAL_ARINT Array2, Position
ST Equal1
```

Representation in ST

Representation applied to integer tables:

```
Equal1:= EQUAL_ARINT(Array1, Array2, Position);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Array1	ARRAY [n..m] OF WORD ARRAY [n..m] OF DWORD ARRAY [n..m] OF INT ARRAY [n..m] OF DINT ARRAY [n..m] OF REAL	n and m are positive or negative integers or nil.
Array2	ARRAY [n..m] OF WORD ARRAY [n..m] OF DWORD ARRAY [n..m] OF INT ARRAY [n..m] OF DINT ARRAY [n..m] OF REAL	n and m are positive or negative integers or nil.
Position	INT	Rank of first element from which the search is launched.

The following table describes the output parameters:

Parameter	Type	Comment
Equal1	INT	Rank of first different elements. If the two tables are equivalent, Equal1 = -1.

Runtime errors

When the table contains an invalid value, the result of the function contains -2 and the bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) = 1.

8 FIND_EQ_***: First element of a table equal to a given value

Description

Function description

The FIND_EQ_*** function searches for the first element of a table equal to a given value.

The additional parameters EN and ENO can be configured.

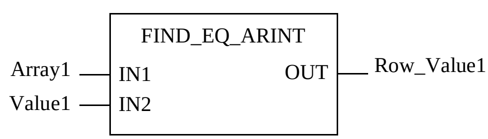
Available functions

The available functions are as follows:

- FIND_EQ_ARWORD,
- FIND_EQ_ARDWORD,
- FIND_EQ_ARINT,
- FIND_EQ_ARINT,
- FIND_EQ_ARREAL.

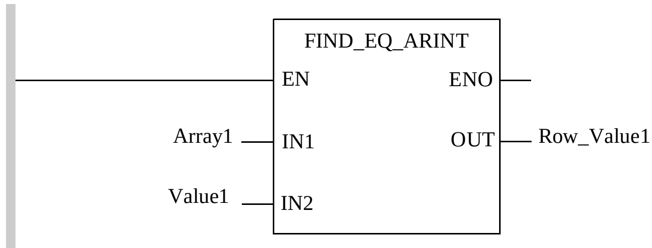
Representation in FBD

Representation applied to an integer table:



Representation in LD

Representation applied to an integer table:



Representation in IL

Representation applied to an integer table:

```
LD Array1
FIND_EQ_ARINT Value1
ST Row_Value1
```

Representation in ST

Representation applied to an integer table:

```
Row_Value1:= FIND_EQ_ARINT(Array1, Value1);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`Array1`	ARRAY [n..m] OF `INT` ARRAY [n..m] OF `DINT` ARRAY [n..m] OF `WORD` ARRAY [n..m] OF `DWORD` ARRAY [n..m] OF `REAL`	n and m are positive or negative integers or nil.
`Value1`	`INT`, `DINT`, `WORD`, `DWORD`, `REAL`.	Value whose rank is searched for in `Array1`. Of the same type as the elements of the table `Array 1`.

The following table describes the output parameters:

Parameter	Type	Comment
`Row_Value1`	`INT`	Rank of first element of `Array1` equal to `Value1`. If none of the elements of the table is equal to `Value1`, `Row_Value1` = -1

Runtime errors

When the table contains an invalid value or if `Value1` is an invalid value, the result of the function contains-2 and the bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) = 1.

FIND_EQP_***: First element of a table equal to a value starting from a given rank

9

Description

Function description

The FIND_EQP_*** function searches for the first element of a table equal to a value starting from a given rank.

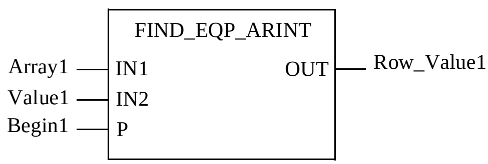
The additional parameters EN and ENO can be configured.

Available functions

The available functions are as follows:
- FIND_EQP_ARWORD,
- FIND_EQP_ARDWORD,
- FIND_EQP_ARINT,
- FIND_EQP_ARDINT,
- FIND_EQP_ARREAL.

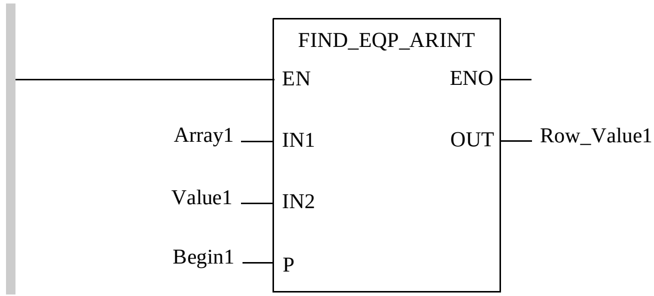
Representation in FBD

Representation applied to an integer table:



Representation in LD

Representation applied to an integer table:



Representation in IL

Representation applied to an integer table:

```
LD Array1
FIND_EQP_ARINT Value1, Begin1
ST Row_Value1
```

Representation in ST

Representation applied to an integer table:

```
Row_Value1:= FIND_EQP_ARINT(Array1, Value1, Begin1);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`Array1`	ARRAY [n..m] OF `WORD` ARRAY [n..m] OF `DWORD` ARRAY [n..m] OF `INT` ARRAY [n..m] OF `DINT` ARRAY [n..m] OF `REAL`	n and m are positive or negative integers or nil.
`Value1`	`WORD`, `DWORD`, `INT`, `DINT`, `REAL`.	Value whose rank is searched for in `Array1`. Of the same type as the elements of the table `Array 1`.
`Begin1`	`INT`	Rank the search starts from

The following table describes the output parameters:

Parameter	Type	Comment
`Row_Value1`	`INT`	Rank of first element of `Array1` equal to `Value1`. If none of the elements of the table is equal to `Value1`, `Row_Value1` = -1 **Note**: `Row_Value1` indicates the rank in relation to the start of the table.

Runtime errors

When the table contains an invalid value or if `Value1` is an invalid value, the result of the function contains-2 and the bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) = 1.

FIND_GT_***: First element of a table greater than a given value

10

Description

Function description

The FIND_GT_*** function searches for the first element of a table greater than a given value.

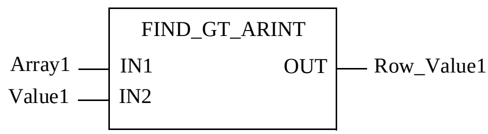
The additional parameters EN and ENO can be configured.

Available functions

The available functions are as follows:
- FIND_GT_ARWORD,
- FIND_GT_ARDWORD,
- FIND_GT_ARINT,
- FIND_GT_ARDINT,
- FIND_GT_ARREAL.

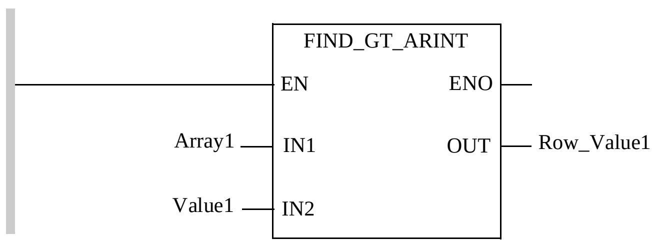
Representation in FBD

Representation applied to an integer table:



Representation in LD

Representation applied to an integer table:



Representation in IL

Representation applied to an integer table:

```
LD Array1
FIND_GT_ARINT Value1
ST Row_Value1
```

Representation in ST

Representation applied to an integer table:

```
Row_Value1:= FIND_GT_ARINT(Array1, Value1);
```

Description of parameters

The following table describes the input parameters:

Input	Type	Comment
Array1	ARRAY [n..m] OF WORD ARRAY [n..m] OF DWORD ARRAY [n..m] OF INT ARRAY [n..m] OF DINT ARRAY [n..m] OF REAL	n and m are positive or negative integers or nil.
Value1	WORD, DWORD, INT, DINT, REAL	Value for which the rank of the first greater value is searched for in Array1. Of the same type as the elements of the table Array 1.

The following table describes the output parameters:

Output	Type	Comment
Row_Value1	INT	Rank of the first element of Array1 > than Value1. If none of the elements of the table is greater than Value1, Row_Value1 = -1

Runtime errors

When the table contains an invalid value or if Value1 is an invalid value, the result of the function contains-2 and the bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) = 1.

FIND_LT_***: First element of a table less than a given value

11

Description

Function description

The FIND_LT_*** function searches for the first element of a table less than a given value.

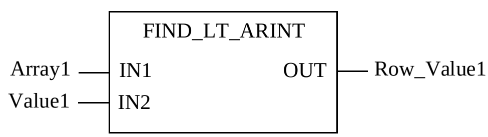
The additional parameters EN and ENO can be configured.

Available functions

The available functions are as follows:
- FIND_LT_ARWORD,
- FIND_LT_ARDWORD,
- FIND_LT_ARINT,
- FIND_LT_ARDINT,
- FIND_LT_ARREAL.

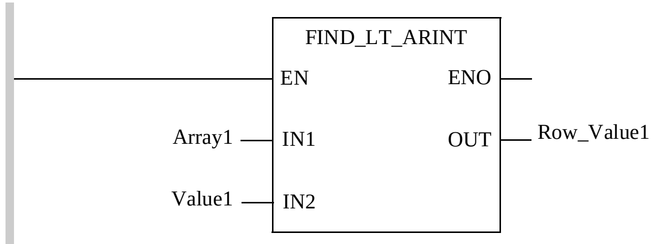
Representation in FBD

Representation applied to an integer table:



Representation in LD

Representation applied to an integer table:



Representation in IL

Representation applied to an integer table:

```
LD Array1
FIND_LT_ARINT Value1
ST Row_Value1
```

Representation in ST

Representation applied to an integer table:

```
Row_Value1:= FIND_LT_ARINT(Array1, Value1);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Array1	ARRAY [n..m] OF WORD ARRAY [n..m] OF DWORD ARRAY [n..m] OF INT ARRAY [n..m] OF DINT ARRAY [n..m] OF REAL	n and m are positive or negative integers or nil.
Value1	WORD, DWORD, INT, DINT, REAL	Value for which a smaller value is searched for in Array1. Of the same type as the elements of the table Array 1.

The following table describes the output parameters:

Parameter	Type	Comment
Row_Value1	INT	Rank of the first element of Array1 < Value1. If none of the elements of the table is less than Value1, Row_Value1 = -1

Runtime errors

When the table contains an invalid value or if `Value1` is an invalid value, the result of the function contains-2 and the bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) = 1.

LENGTH_***: Length of a table

12

Description

Function description

The LENGTH_*** function calculates the length of a table. It is used mainly with DFBs when the tables are not explicitly declared.

The additional parameters EN and ENO can be configured.

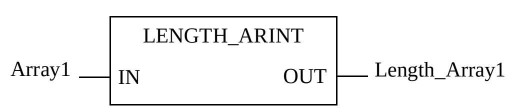
Available functions

The available functions are as follows:

- LENGTH_AREBOOL,
- LENGTH_ARWORD,
- LENGTH_ARDWORD,
- LENGTH_ARINT,
- LENGTH_ARDINT,
- LENGTH_ARREAL,
- LENGTH_ARBOOL,
- LENGTH_ARBYTE,
- LENGTH_ARDATE,
- LENGTH_ARDT,
- LENGTH_ARSTRING,
- LENGTH_ARTIME,
- LENGTH_ARTOD,
- LENGTH_ARUINT,
- LENGTH_ARUDINT.

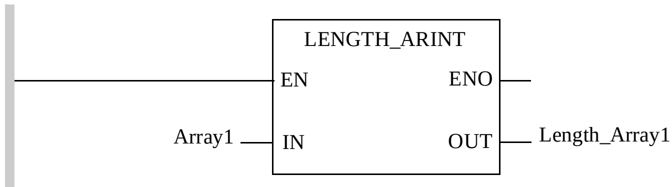
Representation in FBD

Representation applied to an integer table:



Representation in LD

Representation applied to an integer table:



Representation in IL

Representation applied to an integer table:

```
LD Array1
LENGTH_ARINT
ST Length_Array1
```

Representation in ST

Representation:

```
Length_Array1:= LENGTH_ARINT(Array1);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Array1	ARRAY [n..m] OF EBOOL ARRAY [n..m] OF WORD ARRAY [n..m] OF DWORD ARRAY [n..m] OF INT ARRAY [n..m] OF DINT ARRAY [n..m] OF REAL ARRAY [n..m] OF BOOL ARRAY [n..m] OF BYTE ARRAY [n..m] OF DATE ARRAY [n..m] OF DT ARRAY [n..m] OF STRING ARRAY [n..m] OF TIME ARRAY [n..m] OF TOD ARRAY [n..m] OF UINT ARRAY [n..m] OF UDINT	n and m are positive or negative integers or nil.

The following table describes the output parameters:

Parameter	Type	Comment
Length_Array1	INT	Table length (number of table elements).

MAX_***: Maximum value of table elements

13

Description

Function description

The `MAX_***` function searches for the maximum value of the elements of a table.

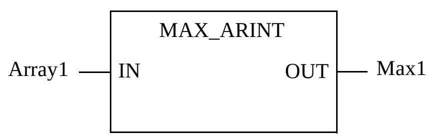
The additional parameters `EN` and `ENO` can be configured.

Available functions

The available functions are as follows:
- `MAX_ARWORD`,
- `MAX_ARDWORD`,
- `MAX_ARINT`,
- `MAX_ARDINT`,
- `MAX_ARREAL`.

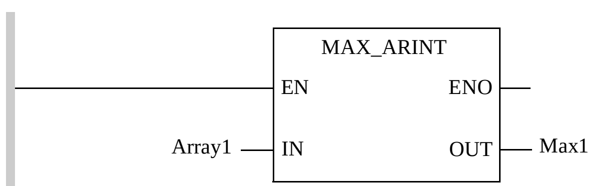
Representation in FBD

Representation applied to an integer table:



Representation in LD

Representation applied to an integer table:



Representation in IL

Representation applied to an integer table:

```
LD Array1
MAX_ARINT
ST Max1
```

Representation in ST

Representation applied to an integer table:

```
Max1:= MAX_ARINT(Array1);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Array1	ARRAY [n..m] OF WORD ARRAY [n..m] OF DWORD ARRAY [n..m] OF INT ARRAY [n..m] OF DINT ARRAY [n..m] OF REAL	n and m are positive or negative integers or nil.

The following table describes the output parameters:

Parameter	Type	Comment
Max1	WORD, DWORD, INT, DINT, REAL	Maximum value contained in the table. This result is of the same type as the table elements.

Runtime errors

When the table contains an invalid value, the result of the function contains -1.#INF and the bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) = 1.

MIN_***: Minimum value of table elements

14

Description

Function description

The MIN_*** function searches for the minimum value of the elements of a table.

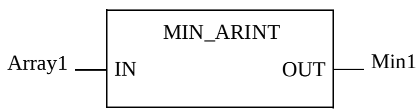
The additional parameters EN and ENO can be configured.

Available functions

The available functions are as follows:
- MIN_ARWORD,
- MIN_ARDWORD,
- MIN_ARINT,
- MIN_ARDINT,
- MIN_ARREAL.

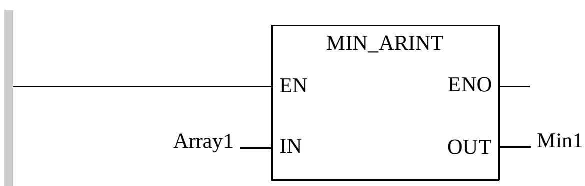
Representation in FBD

Representation applied to an integer table:



Representation in LD

Representation applied to an integer table:



Representation in IL

Representation applied to an integer table:

```
LD Array1
MIN_ARINT
ST Min1
```

Representation in ST

Representation applied to an integer table:

```
Min1:= MIN_ARINT(Array1);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Array1	ARRAY [n..m] OF WORD ARRAY [n..m] OF DWORD ARRAY [n..m] OF INT ARRAY [n..m] OF DINT ARRAY [n..m] OF REAL	n and m are positive or negative integers or nil.

The following table describes the output parameters:

Parameter	Type	Comment
Min1	WORD, DWORD, DINT, INT, REAL	Minimum value contained in the table. This result is of the same type as the table elements.

Runtime errors

When the table contains an invalid value, the result of the function contains 1.#INF and the bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) = 1.

MOD_***_***: Remainder of division of tables

15

Description

Function description

The function MOD_***_*** calculates the remainder of the division:
- of a number by the elements of a table,
- of the elements of a table by a number,
- of the elements of a table by the respective elements of another table.

The additional parameters EN and ENO can be configured.

Available functions

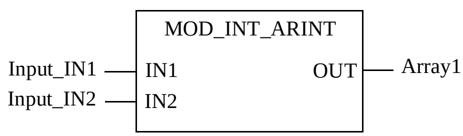
The available functions for calculation of the remainder of the division of a number by the elements of a table are as follows:
- MOD_INT_ARINT,
- MOD_DINT_ARDINT.

The available functions for the calculation of the remainder of the division of the elements of a table by a number are as follows:
- MOD_ARINT_INT,
- MOD_ARDINT_DINT.

The available functions for the calculation of the remainder of the division of the elements of a table by the respective elements of another table are as follows:
- MOD_ARINT,
- MOD_ARDINT.

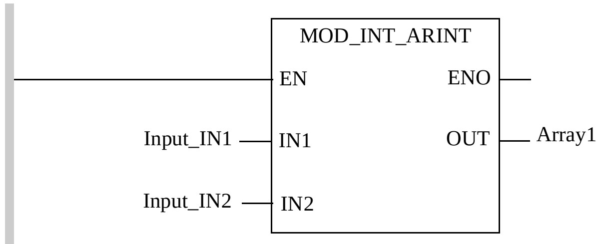
Representation in FBD

Representation applied to the division of an integer by the elements of an integer table:



Representation in LD

Representation applied to the division of an integer by the elements of an integer table:



Representation in IL

Representation applied to the division of an integer by the elements of an integer table:

```
LD Input_IN1
MOD_INT_ARINT Input_IN2
ST Array1
```

Representation in ST

Representation applied to the division of an integer by the elements of an integer table:

```
Array1:= MOD_INT_ARINT(Input_IN1,Input_IN2);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Input_IN1	INT, DINT, ARRAY [n..m] OF INT, ARRAY [n..m] OF DINT	Input_IN1 is either a single or double integer, or a table of single or double integers, n and m are positive or negative integers or nil.
Input_IN2	INT, DINT, ARRAY [n..m] OF INT, ARRAY [n..m] OF DINT	Input_IN2 is either a single or double integer, or a table of single or double integers, n and m are positive or negative integers or nil.

The following table describes the output parameters:

Parameter	Type	Comment
Array1	ARRAY [n..m] OF INT ARRAY [n..m] OF DINT	According to the type of Input_IN1 and Input_IN2, each element of Array1 is the remainder of the division: ● of a single or double integer Input_IN1 by the corresponding element of the table Input_IN2 or else, ● of the elements of the table Input_IN1 by the single or double integer Input_IN2 or else, ● of the elements of the table Input_IN1 by the respective elements of the table Input_IN2.

Runtime errors

The management of the **%S18** (See *Description of system bits %S15 to %S21, p. 448*) system bit is identical to that for operations on words or double words. The remainder of a division by zero is zero and the system bit is set to 1.

If an operation between two elements sets the **%S18** (See *Description of system bits %S15 to %S21, p. 448*) bit (overflow or division by zero), the result for this operation is incorrect, but the operation on the following elements is carried out correctly.

MOVE_***_***: Assignment to tables

16

Description

Function description

One of the actions of the MOVE_***_*** function is the assignment of an identical value to each element of a table.

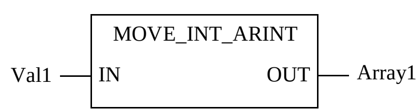
The additional parameters EN and ENO can be configured.

Available functions

The available functions are as follows:
- MOVE_BOOL_AREBOOL,
- MOVE_WORD_ARWORD,
- MOVE_DWORD_ARDWORD,
- MOVE_INT_ARINT,
- MOVE_DINT_ARDINT,
- MOVE_REAL_ARREAL.

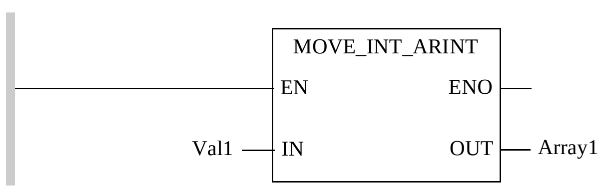
Representation in FBD

Representation applied to the assignment of an integer to an integer table:



Representation in LD

Representation applied to the assignment of an integer to an integer table:



Representation in IL

Representation applied to the assignment of an integer to an integer table:

```
LD Val1
MOVE_INT_ARINT Array1
```

Representation in ST

Representation applied to the assignment of an integer to an integer table:

```
MOVE_INT_ARINT(Val1, Array1);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Val1	BOOL, WORD, DWORD, INT, DINT, REAL.	Val1 contains the value to be assigned to each element of the table Array1. Of the same type as the elements of the table Array1.

The following table describes the output parameters:

Parameter	Type	Comment
Array1	ARRAY [n..m] OF EBOOL, ARRAY [n..m] OF WORD, ARRAY [n..m] OF DWORD, ARRAY [n..m] OF INT, ARRAY [n..m] OF DINT ARRAY [n..m] OF REAL	n and m are positive or negative integers or nil. Array1 is a table each element of which is of the value Val1.

MOVE_***_***: Table conversion

17

Description

Function description

One of the actions of the MOVE_***_*** function is to convert a table into a value or a value into a table.

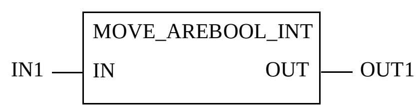
The additional parameters EN and ENO can be configured.

Available functions

The available functions are as follows:
- MOVE_AREBOOL_INT (conversion of an EBOOL table into an INT).
- MOVE_AREBOOL_DINT (conversion of an EBOOL table into a DINT).
- MOVE_INT_AREBOOL (conversion of an INT into an EBOOL table).
- MOVE_DINT_AREBOOL (conversion of a DINT into an EBOOL table).

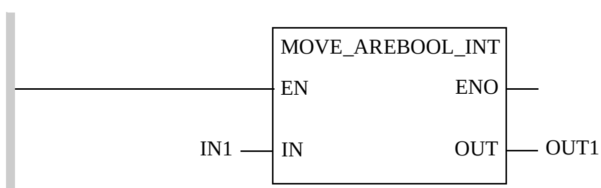
Representation in FBD

Representation applied to the conversion of an EBOOL table into an integer:



Representation in LD

Representation applied to the conversion of an EBOOL table into an integer:



Representation in IL

Representation applied to the conversion of an EBOOL table into an integer:

```
LD IN1
MOVE_AREBOOL_INT OUT1
```

Representation in ST

Representation applied to the conversion of an EBOOL table into an integer:

```
MOVE_AREBOOL_INT(IN1, OUT1);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
IN1	INT, DINT, ARRAY [n..m] OF EBOOL.	n and m are positive or negative integers or nil.

The following table describes the output parameters:

Parameter	Type	Comment
OUT1	INT, DINT, ARRAY [n..m] OF EBOOL.	When IN1 is an EBOOL table, OUT1 is an INT or DINT containing the elements of IN1. When IN1 is not a table, OUT1 is a single or double integer, converted from a Boolean table.

MUL_***_***: Multiplication of tables

18

Description

Function description

The `MUL_***_***` function carries out the multiplication:
- of the elements of a table by a number,
- of the elements of a table by the respective elements of another table.

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

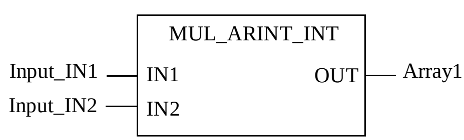
Available functions

The available functions for the multiplication of the elements of a table by a number are as follows:
- `MUL_ARINT_INT`,
- `MUL_ARDINT_DINT`.

The available functions for the multiplication of the elements of a table by the respective elements of another table are as follows:
- `MUL_ARINT`,
- `MUL_ARDINT`.

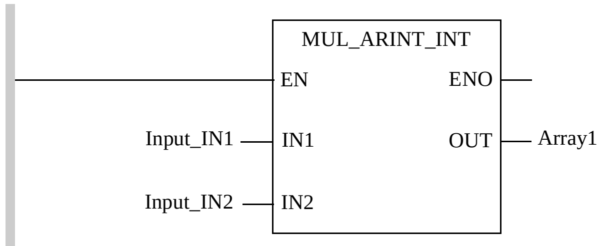
Representation in FBD

Representation applied to the multiplication of the elements of an integer table by an integer:



Representation in LD

Representation applied to the multiplication of the elements of an integer table by an integer:



Representation in IL

Representation applied to the multiplication of the elements of an integer table by an integer:

```
LD Input_IN1
MUL_ARINT_INT Input_IN2
ST Array1
```

Representation in ST

Representation applied to the multiplication of an integer by the elements of an integer table:

```
Array1:= MUL_ARINT_INT(Input_IN1,Input_IN2);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`Input_IN1`	`INT`, `DINT`, ARRAY [n..m] OF `INT`, ARRAY [n..m] OF `DINT`	`Input_IN1` is either a single or double integer, or a table of single or double integers, n and m are positive or negative integers or nil.
`Input_IN2`	`INT`, `DINT`, ARRAY [n..m] OF `INT`, ARRAY [n..m] OF `DINT`	`Input_IN2` is either a single or double integer, or a table of single or double integers, n and m are positive or negative integers or nil.

The following table describes the output parameters:

Parameter	Type	Comment
`Array1`	ARRAY [n..m] OF `INT` ARRAY [n..m] OF `DINT`	According to the type of `Input_IN1` and `Input_IN2`, each element of `Array1` is the multiplication: ● of a single or double integer `Input_IN1` by the corresponding element of the table `Input_IN2` or else, ● of the elements of the table `Input_IN1` by single or double integers `Input_IN2` or else, ● of the elements of the table `Input_IN1` by the respective elements of the table `Input_IN2`.

Runtime errors

The management of the **%S18** (See *Description of system bits %S15 to %S21, p. 448*) system bit is identical to that for operations on words or double words.

If an operation between two elements sets the **%S18** (See *Description of system bits %S15 to %S21, p. 448*) bit (overflow or division by zero), the result for this operation is incorrect, but the operation on the following elements is carried out correctly.

NOT_***: Logical negation of tables

19

Description

Function description

The NOT_*** function carries out a logical negation (bit to bit) between the elements of two tables.

Note: The result is always a table.

The additional parameters EN and ENO can be configured.

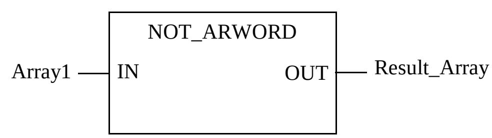
Available functions

The available functions are as follows:
- NOT_AREBOOL,
- NOT_ARWORD,
- NOT_ARDWORD.

The functions available in the **Obsolete** library are the following:
- NOT_ARINT (logical negation of each element of an INT table).
- NOT_ARDINT (logical negation of each element of a DINT table).

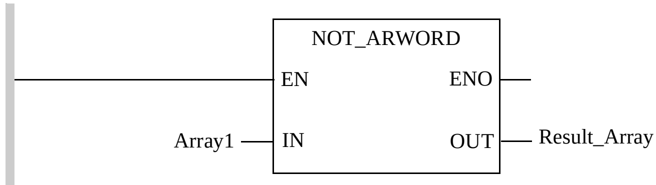
Representation in FBD

Representation applied to a 16-bit string:



Representation in LD

Representation applied to a 16-bit string:



Representation in IL

Representation applied to a 16-bit string:

```
LD Array1
NOT_ARWORD
ST Result_Array
```

Representation in ST

Representation applied to a 16-bit string:

```
Result_Array:= NOT_ARWORD(Array1);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`Array1`	ARRAY [n..m] OF `EBOOL`, ARRAY [n..m] OF `WORD`, ARRAY [n..m] OF `DWORD`	n and m are positive or negative integers or nil.

The following table describes the output parameters:

Parameter	Type	Comment
`Result_Array`	ARRAY [n..m] OF `EBOOL`, ARRAY [n..m] OF `WORD`, ARRAY [n..m] OF `DWORD`	The elements of `Result_Array` are the result of the logical NOT (bit to bit) on `Array1`. Of the same type as the elements of the table `Array 1`.

OCCUR_***: Occurrence of a value in a table

20

Description

Function description

The OCCUR_*** function gives the number of elements of a table equal to a given value.

The additional parameters EN and ENO can be configured.

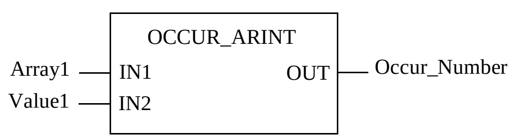
Available functions

The available functions are as follows:

- OCCUR_ARWORD,
- OCCUR_ARDWORD,
- OCCUR_ARINT,
- OCCUR_ARDINT,
- OCCUR_ARREAL.

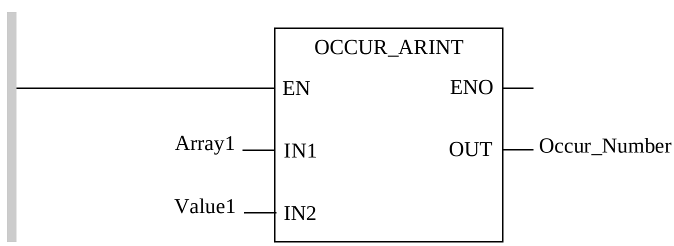
Representation in FBD

Representation applied to an integer table:



Representation in LD

Representation applied to an integer table:



Representation in IL

Representation applied to an integer table:

```
LD Array1
OCCUR_ARINT Value1
ST Occur_Number
```

Representation in ST

Representation applied to an integer table:

```
Occur_Number:= OCCUR_ARINT(Array1,Value1);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`Array1`	ARRAY [n..m] OF `WORD`, ARRAY [n..m] OF `DWORD`, ARRAY [n..m] OF `INT`, ARRAY [n..m] OF `DINT` ARRAY [n..m] OF `REAL`	n and m are positive or negative integers or nil.
`Value1`	`WORD`, `DWORD`, `INT`, `DINT`, `REAL`	Value of which we wish to know the number of occurrences in the table`Array1`. Of the same type as the elements of the table `Array 1`.

The following table describes the output parameters:

Parameter	Type	Comment
`Occur_Number`	`INT`	Number of occurrences of `Value1` in the table `Array1`.

OR_***_***: Logical OR between tables and variables

21

Description

Function description

The OR_***_*** function carries out a logical OR (bit to bit) between:
- the elements of two tables,
- between a single type variable and the elements of a table,
- between the elements of a table and a single type variable.

Note: The result is always a table.

The additional parameters EN and ENO can be configured.

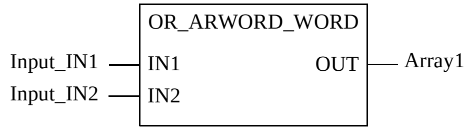
Available functions

The functions available in the general library are the following:
- OR_AREBOOL (logical OR of two BOOL tables).
- OR_ARWORD (logical OR of two WORD tables).
- OR_ARWORD_WORD (logical OR of each element of a WORD table with a WORD).
- OR_ARDWORD_DWORD (logical OR of each element of a DWORD table with a DWORD).
- OR_ARDWORD (logical OR of two DWORD tables).

The functions available in the **Obsolete** library are the following:
- OR_ARINT_INT (logical OR of each element of an INT table with an INT).
- OR_ARDINT_DINT (logical OR of each element of a DINT table with a DINT).
- OR_ARINT (logical OR of each element of an INT table with each element corresponding to another INT table).
- OR_ARDINT (logical OR of each element of a DINT table with each element corresponding to another DINT table).

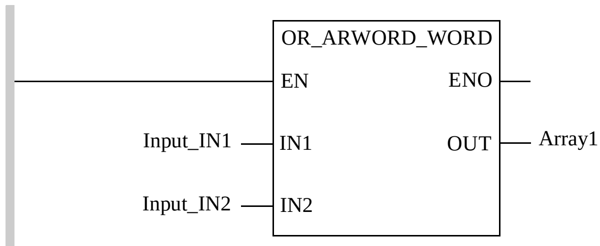
Representation in FBD

Representation applied to a 16-bit string table and a 16-bit string:



Representation in LD

Representation applied to a 16-bit string table and a 16-bit string:



Representation in IL

Representation applied to a 16-bit string table and a 16-bit string:

```
LD IN1
OR_ARWORD_WORD Input_IN2
ST Array1
```

Representation in ST

Representation applied to a 16-bit string and a 16-bit string table:

```
Array1:= OR_ARWORD_WORD(Input_IN1,Input_IN2);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Input_IN1	ARRAY [n..m] OF BOOL, ARRAY [n..m] OF WORD, ARRAY [n..m] OF DWORD, ARRAY [n..m] OF INT, ARRAY [n..m] OF DINT	n and m are positive or negative integers or nil.
Input_IN2	WORD, DWORD, INT, DINT, ARRAY [n..m] OF BOOL, ARRAY [n..m] OF WORD, ARRAY [n..m] OF DWORD, ARRAY [n..m] OF INT, ARRAY [n..m] OF DINT	n and m are positive or negative integers or nil.

The following table describes the output parameters:

Parameter	Type	Comment
Array1	ARRAY [n..m] OF BOOL, ARRAY [n..m] OF WORD, ARRAY [n..m] OF DWORD, ARRAY [n..m] OF INT, ARRAY [n..m] OF DINT	The elements of Array1 are the result of the logical OR (bit to bit) between Input_IN1 and Input_IN2, which can be respectively: ● a table and a single variable, ● a table and a table.

ROL_***: Rotate shift to left

22

Description

Function description

The ROL_*** function carries out a rotate shift of the elements of a table in the ascending direction of the indices.

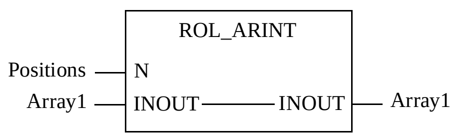
The additional parameters EN and ENO can be configured.

Available functions

The functions available in the general library are the following:
- ROL_ARWORD,
- ROL_ARDWORD,
- ROL_ARINT,
- ROL_ARDINT,
- ROL_ARREAL.

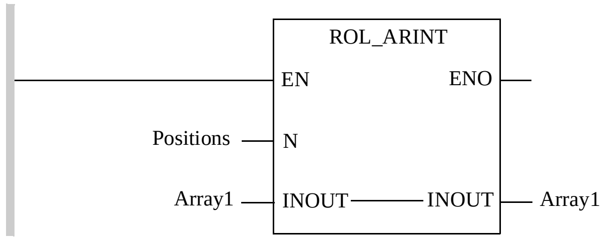
Representation in FBD

Representation applied to an integer table:



Representation in LD

Representation applied to an integer table:



Representation in IL

Representation applied to an integer table:

```
LD Positions
ROL_ARINT Array1
```

Representation in ST

Representation applied to an integer table:

```
ROL_ARINT(Positions, Array1);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`Positions`	`INT`	Shift value according to the ascending indices of the table. **Example**: `Positions` = 2. Note: if the value of `Positions` is negative or nil, no shift is carried out.

The following table describes the input/output parameters:

Parameter	Type	Comment
`Array1`	ARRAY [n..m] OF `WORD`, ARRAY [n..m] OF `DWORD`, ARRAY [n..m] OF `INT`, ARRAY [n..m] OF `DINT` ARRAY [n..m] OF `REAL`	The elements of `Array1` are moved a number of positions equal to `Positions`. The shift is carried out according to the ascending indices. **Example**: With a shift register of 2, the element previously situated in first position goes to third (1+2), the second goes to fourth (2+2), ..., the second last goes to first position and the last goes to second position.

ROR_***: Rotate shift to right

23

Description

Function description

The ROR_*** function carries out a rotate shift of the elements of a table in the descending direction of the indices.

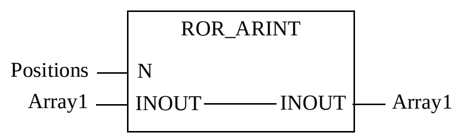
The additional parameters EN and ENO can be configured.

Available functions

The functions available in the general library are the following:
- ROR_ARWORD,
- ROR_ARDWORD,
- ROR_ARINT,
- ROR_ARDINT.
- ROR_ARREAL.

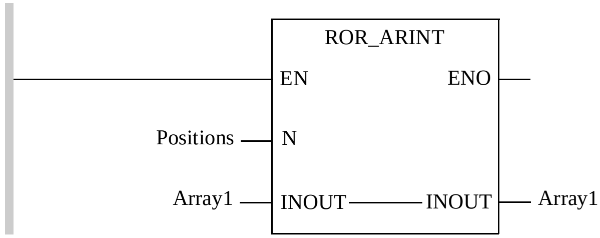
Representation in FBD

Representation applied to an integer table:



Representation in LD

Representation applied to an integer table:



Representation in IL

Representation applied to an integer table:

```
LD Positions
ROR_ARINT Array1
```

Representation in ST

Representation applied to an integer table:

```
ROR_ARINT(Positions, Array1);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`Positions`	`INT`	Shift value according to the descending indices of the table. **Example**: `Positions` = 2. Note: if the value of `Positions` is negative or nil, no shift is carried out.

The following table describes the input/output parameters:

Parameter	Type	Comment
`Array1`	ARRAY [n..m] OF `WORD`, ARRAY [n..m] OF `DWORD`, ARRAY [n..m] OF `INT`, ARRAY [n..m] OF `DINT` ARRAY [n..m] OF `REAL`	The elements of `Array1` are moved a number of positions equal to `Positions`. The shift is carried out according to the descending indicators. **Example**: With a shift register of 2, the element previously situated in first position goes to second last, the second goes to last, the third goes to first (3-2), the fourth to second (4 -2), etc.

SORT_***: Ascending or descending sort

24

Description

Function description

The SORT_*** function sorts a table in ascending or descending order and arranges the sorted elements in this same table.

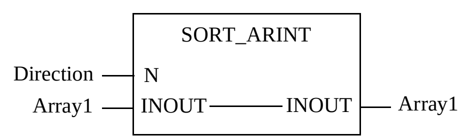
The additional parameters EN and ENO can be configured.

Available functions

The available functions are as follows:
- SORT_ARWORD,
- SORT_ARDWORD,
- SORT_ARINT,
- SORT_ARDINT,
- SORT_ARREAL.

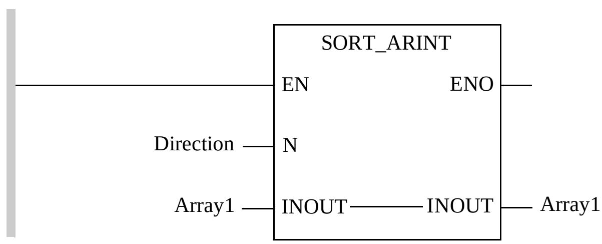
Representation in FBD

Representation applied to an integer table:



Representation in LD

Representation applied to an integer table:



Representation in IL

Representation applied to an integer table:

```
LD Direction
SORT_ARINT Array1
```

Representation in ST

Representation applied to an integer table:

```
SORT_ARINT(Direction, Array1);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`Direction`	`INT`	Direction of sort to be carried out: • `Direction` ≥ 0: ascending sort, • `Direction` < 0: descending sort.

The following table describes the input/output parameters:

Parameter	Type	Comment
`Array1`	ARRAY [n..m] OF `WORD`, ARRAY [n..m] OF `DWORD`, ARRAY [n..m] OF `INT`, ARRAY [n..m] OF `DINT` ARRAY [n..m] OF `REAL`	Table sorted in the direction specified in `Direction`, n and m are positive or negative integers, or nil.

SUB_***_***: Subtraction from tables

25

Description

Function description

The SUB_***_*** function carries out the subtraction:
- of the elements of a table from a number,
- of a number from the elements of a table,
- of the elements of a table from the respective elements of another table.

The additional parameters EN and ENO can be configured.

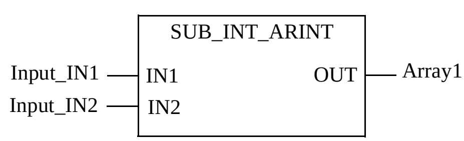
Available functions

The available functions for the subtraction of the elements of a table from a number or of a number from the elements of a table are as follows:
- SUB_INT_ARINT (Subtraction of each element of an INT table from an INT).
- SUB_DINT_ARDINT (Subtraction of each element of a table of DINTs from a DINT).
- SUB_ARINT_INT (Subtraction of an INT from the elements of a table of INTs).
- SUB_ARDINT_DINT (Subtraction of a DINT from the elements of a table of DINTs).

The available functions for the subtraction of the elements of a table from the respective elements of another table are as follows:
- SUB_ARINT (Subtraction of the respective elements of both tables of INTs).
- SUB_ARDINT (Subtraction of the respective elements of both tables of DINTs).

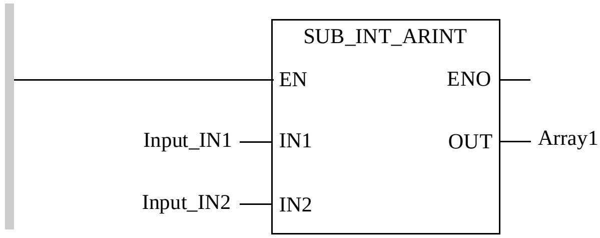
Representation in FBD

Representation applied to the subtraction of the elements of a table of integers from an integer:



Representation in LD

Representation applied to the subtraction of the elements of a table of integers from an integer:



Representation in IL

Representation applied to the subtraction of the elements of a table of integers from an integer:

```
LD Input_IN1
SUB_INT_ARINT Input_IN2
ST Array1
```

Representation in ST

Representation applied to the subtraction of the elements of a table of integers from an integer:

```
Array1:= SUB_INT_ARINT(Input_IN1,Input_IN2);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`Input_IN1`	`INT`, `DINT`, ARRAY [n..m] OF `INT`, ARRAY [n..m] OF `DINT`	`Input_IN1` is either a single or double integer, or a table of single or double integers, n and m are positive or negative integers or nil.
`Input_IN2`	`INT`, `DINT`, ARRAY [n..m] OF `INT`, ARRAY [n..m] OF `DINT`	`Input_IN2` is either a single or double integer, or a table of single or double integers, n and m are positive or negative integers or nil.

The following table describes the output parameters:

Parameter	Type	Comment
`Array1`	ARRAY [n..m] OF `INT` ARRAY [n..m] OF `DINT`	According to the type of `Input_IN1` and `Input_IN2`, each element of `Array1` is the subtraction: ● from a single or double integer `Input_IN1` of the corresponding element of the table `Input_IN2` or else, ● from the elements of the table `Input_IN1` of the single or double integer `Input_IN2` or else, ● from the elements of the table `Input_IN1` of the respective elements of the table `Input_IN2`.

Runtime errors

The management of the **%S18** (See *Description of system bits %S15 to %S21, p. 448*) system bit is identical to that for operations on words or double words.

If an operation between two elements sets the **%S18** (See *Description of system bits %S15 to %S21, p. 448*) bit (overflow or division by zero), the result for this operation is incorrect, but the operation on the following elements is carried out correctly.

SUM_***: Sum of table elements

26

Description

Function description

The SUM_*** function calculates the sum of the elements of a table.

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

Available functions

The available functions are as follows:
- `SUM_ARINT`,
- `SUM_ARDINT`,
- `SUM_ARREAL`.

Formula

The formula is as follows:

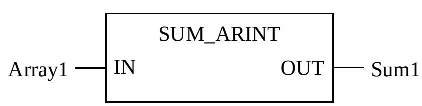
$$\text{Sum1} = \sum_{i=n}^{j=m} \text{Array1}[j]$$

Description:

Element	Signification
Array1	Table declared in the following way: `ARRAY [n..m] OF ...`

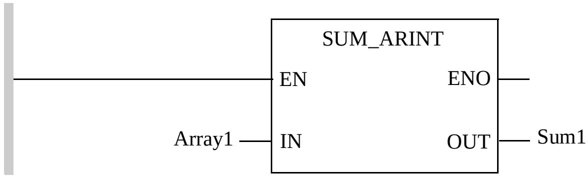
Representation in FBD

Representation applied to an integer table:



Representation in LD

Representation applied to an integer table:



Representation in IL

Representation applied to an integer table:

```
LD Array1
SUM_ARINT
ST Sum1
```

Representation in ST

Representation applied to an integer table:

```
Sum1:= SUM_ARINT(Array1);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`Array1`	ARRAY [n..m] OF `INT` ARRAY [n..m] OF `DINT` ARRAY [n..m] OF `REAL`	Double or single integer tables or tables of reals, n and m are positive or negative integers, or nil.

The following table describes the output parameters:

Parameter	Type	Comment
`Sum1`	`INT`, `DINT`, `REAL`	Sum of table elements assigned to input. The sum is of the same type as the table elements.

Runtime errors

When the table contains an invalid value, the sum of its elements contains 0.0 and the bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) = 1.
When the sum of elements is greater than the maximum authorized value, its value becomes 1.#INF and the bit %S18 = 1

SWAP_***: Permutation of the bytes of a table

27

Description

Function description

The `SWAP_***` function carries out a permutation of the least significant bytes and the most significant bytes of the elements of a table.

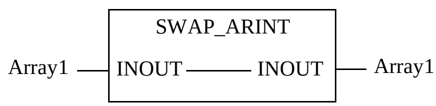
The additional parameters `EN` and `ENO` can be configured.

Available functions

The available functions are as follows:
- `SWAP_ARINT`,
- `SWAP_ARWORD`.

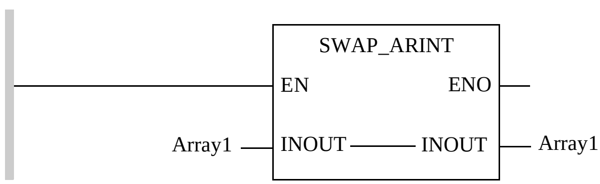
Representation in FBD

Representation applied to an integer table:



Representation in LD

Representation applied to an integer table:



Representation in IL

Representation applied to an integer table:

```
LD Array1
SWAP_ARINT
```

Representation in ST

Representation applied to an integer table:

```
SWAP_ARINT(Array1);
```

Description of parameters

The following table describes the input/output parameters:

Parameter	Type	Comment
`Array1`	ARRAY [n..m] OF `INT` ARRAY [n..m] OF `WORD`	n and m are positive or negative integers or nil. On output, the bytes of `Array1` have been permutated.

XOR_***_***: Exclusive OR between tables

28

Description

Function description

The XOR_***_*** function carries out an exclusive logical OR (bit to bit) between:
- the elements of two tables,
- between a single type variable and the elements of a table,
- between the elements of a table and a single type variable.

Note: The result is always a table.

The additional parameters EN and ENO can be configured.

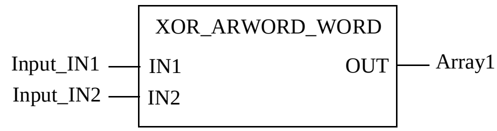
Available functions

The functions available in the general library are the following:
- XOR_AREBOOL (exclusive logical OR of two BOOL tables).
- XOR_ARWORD (exclusive logical OR of two WORD tables).
- XOR_ARWORD_WORD (exclusive logical OR of each element of a WORD table with a WORD).
- XOR_ARDWORD_WORD (exclusive logical OR of each element of a DWORD table with a DWORD).
- XOR_ARDWORD (exclusive logical OR of two DWORD tables).

The functions available in the **Obsolete** library are the following:
- XOR_ARINT_INT (exclusive logical OR of each element of an INT table with an INT).
- XOR_ARDINT_DINT (exclusive logical OR of each element of an DINT table with an DINT).
- XOR_ARINT (exclusive logical OR of each element of an INT table with each element corresponding to another INT table).
- XOR_ARDINT (exclusive logical OR of each element of a DINT table with each element corresponding to another DINT table).

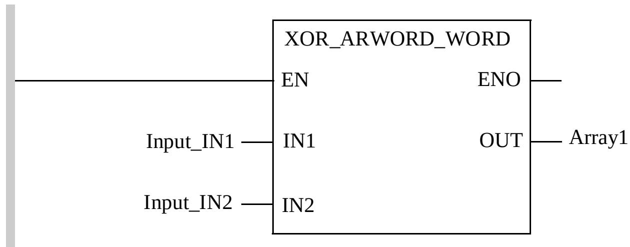
Representation in FBD

Representation applied to a 16-bit string table and a 16-bit string:



Representation in LD

Representation applied to a 16-bit string table and a 16-bit string:



Representation in IL

Representation applied to a 16-bit string table and a 16-bit string:

```
LD Input_IN1
XOR_ARWORD_WORD Input_IN2
ST Array1
```

Representation in ST

Representation applied to a 16-bit string table and a 16-bit string:

```
Array1:= XOR_ARWORD_WORD(Input_IN1,Input_IN2);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`Input_IN1`	ARRAY [n..m] OF `BOOL`, ARRAY [n..m] OF `WORD`, ARRAY [n..m] OF `DWORD`, ARRAY [n..m] OF `INT`, ARRAY [n..m] OF `DINT`	n and m are positive or negative integers or nil.
`Input_IN2`	`WORD`, `DWORD`, `INT`, `DINT`, ARRAY [n..m] OF `BOOL`, ARRAY [n..m] OF `WORD`, ARRAY [n..m] OF `DWORD`, ARRAY [n..m] OF `INT`, ARRAY [n..m] OF `DINT`	n and m are positive or negative integers or nil.

The following table describes the output parameters:

Parameter	Type	Comment
`Array1`	ARRAY [n..m] OF `BOOL`, ARRAY [n..m] OF `WORD`, ARRAY [n..m] OF `DWORD`, ARRAY [n..m] OF `INT`, ARRAY [n..m] OF `DINT`	The elements of `Array1` are the result of the exclusive logical OR (bit to bit) between `Input_IN1` and `Input_IN2`, which can be respectively: ● a table and a single variable, ● a single variable and a table, ● a table and a table.

CLC_INT

III

Introduction

Overview

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

What's in this Part?

This part contains the following chapters:

Introduction to integer regulation functions

29

At a Glance

Subject of this Chapter

This chapter provides the basic notions necessary for the use and implementation of the following integer regulation functions:
- PID_INT,
- PWM_INT,
- SERVO_INT.

What's in this Chapter?

This chapter contains the following topics:

General Introduction

General

The regulation functions are the **standard elements** of the language. They are used to program regulation loops.

These functions are particularly adapted to:

- meeting the requirements of sequential processes which need auxiliary regulation functions (e.g.: plastic film packaging machines, finishing treatment machines, presses etc.)
- meeting the requirements of simple regulation processes (e.g.: metal furnaces, ceramic furnaces, small refrigerating units etc.),
- meeting the specific requirements of mechanical regulation or feedback control where sampling time is critical (e.g.: torque regulation, speed regulation).

> **Note:** There is no limit on the number of `PID_INT` functions that are available in an application. In practice, it is the maximum number of input and output modules which are accepted by the PLC that limits the number of loops.

Available functions

The basic regulation functions are the following:

- the `PID_INT` function to execute a mixed `PID_INT` correction (serial – parallel),
- the `PWM_INT` function to execute the modulation adjustment period on the discrete outputs,
- the `SERVO_INT` function to execute the motor command adaptations.

Principal of the regulation loop

At a Glance

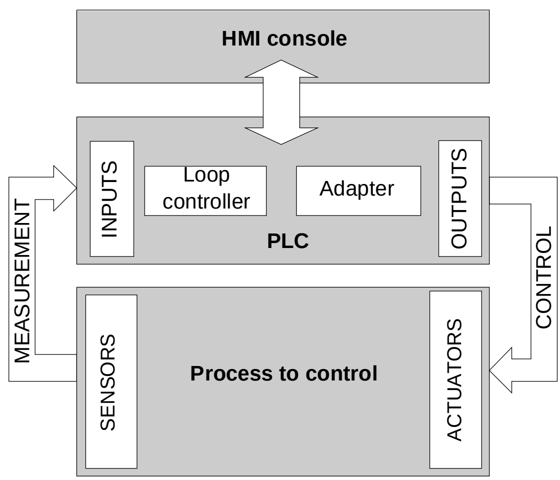
A regulation loop has three distinct operating phases:
- the acquisition of data:
  - measurements from the process' sensors (analog, encoders),
  - setpoint(s) generally from PLC internal variables or from data from the operator terminal.
- execution of the PID regulation algorithm,
- the sending of commands adapted to the characteristics of the actuators to be driven via the discrete or analog outputs.

The PID algorithm generates the command signal from:
- the measurement sampled by the input nodule,
- the setpoint value fixed by either the operator or the program,
- the values of the different corrector parameters.

The signal from the corrector is either directly handled by an analog output card of the PLC linked to the actuator, or handled via the PWM or SERVO adjustments depending on the types of actuator to be driven on a discrete output card of the PLC.

Illustration

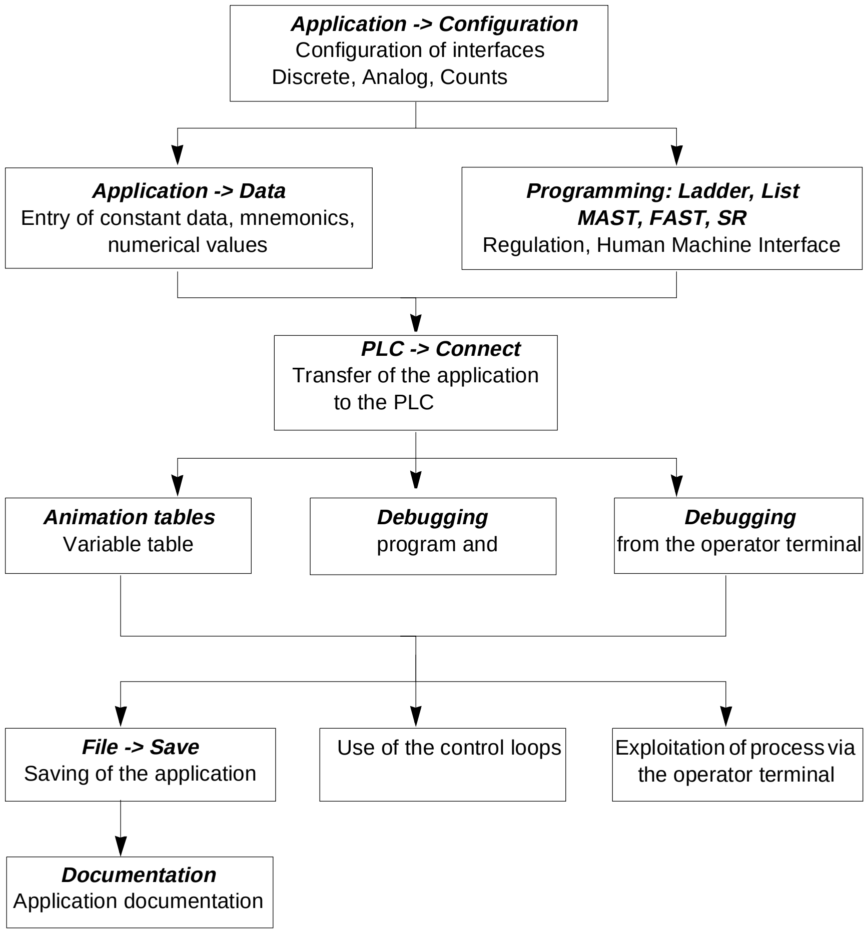
The following diagram schematizes the principal of a regulation loop.



Development methodology for a regulation application

Flow diagram

The following diagram shows the links between tasks to be carried out during the creation and debugging of a regulation application (the order shown here is for information purposes only).



Programming a regulation function

Programming rules

The regulation function parameters must all be entered. The functions use 3 kinds of parameters:
- read only parameters, considered at the beginning of the function’s execution,
- write only parameters, positioned at the conclusion of the function’s execution,
- the read and write parameters, whose contents are considered at the beginning of the function’s execution, are then updated by the results of the function.

Note: The regulation functions must be programmed in a periodic **task** (periodic MAST or FAST). They must not be conditioned.

Parametering

The word type input parameters are analog dimensions expressed on the scale [0, +10000] and can be directly connected to measurement sensors via the %IWr.m.c words of the analog inputs.

The bit type output parameters can be used to control discrete actuators and can be directly connected to the %Qr.m.c. type variables.

In the same way, the word type output parameters can be used to control analog actuators on the scale [0, +10000] and can be directly assigned to %QWr.m.c type variables.

The ARRAY [0..n] OF INT or %MWi:L integer table type parameters contain the user parameters and data necessary to the internal operation of the function.
If the length of a table is insufficient, the function is not executed.

Note: In order to keep the adjustment parameters of the regulation on cold start function, it is necessary to delete the %MWi reset to zero option (in the processor’s configuration screen)

Behavior of functions in operating modes

Introduction

This paragraph describes the behavior of the functions in different start-up scenarios:

- cold start (new application, change of cartridge…),
- warm restart (power return without changing the application context),
- first execution after adding a function via modification in connected mode.

Cold start

This type of start occurs for a new application or a change of cartridge
On a cold start, the PLC can start automatically in RUN (according to the application's configuration). The function correctors have the following security behavior: manual mode, outputs at 0. In addition, this supports the switching of the PLC into RUN mode without carrying out the PID adjustment, then its debugging with the operator terminal (the adjustment can only be performed in RUN).

Warm restart

This type of restart occurs for a power return, without changing the application context.
With a power return after an outage (regardless of how long it lasted) and if the application context is not lost or modified, the functions go back to their state before the outage. If the user wants to use another type of behavior, it is his responsibility to test the %S1 system bit and to associate the required processing with it (forcing in manual mode…).

> **Note:** The PLC's time-and-date stamp allows you find out the duration of the last outage.

Adding a new call in connected mode

Following the addition of a new function regulation call in connected mode, an identical initialization to the case of the cold start is carried out.

> **Note:** In order to be seen as a new function, this must use a new parameter table. Therefore, the removal of a `PID_INT` function, followed by the addition of a `PID_INT` function that uses the same parameter table is not considered as an addition of a new PID. In this case the PID is executed in the same state and with the same parameters as the preceding PID.

PID_INT: PID controller

30

Description

Subject of this Chapter

This chapter describes the PID_INT function.

What's in this Chapter?

This chapter contains the following topics:

Function description

Function description

The PID_INT function carries out PID-type regulation on INT type inputs and outputs.
The measurement and the setpoint are analog data in [0-10000] format and generate an analog command in the same format.

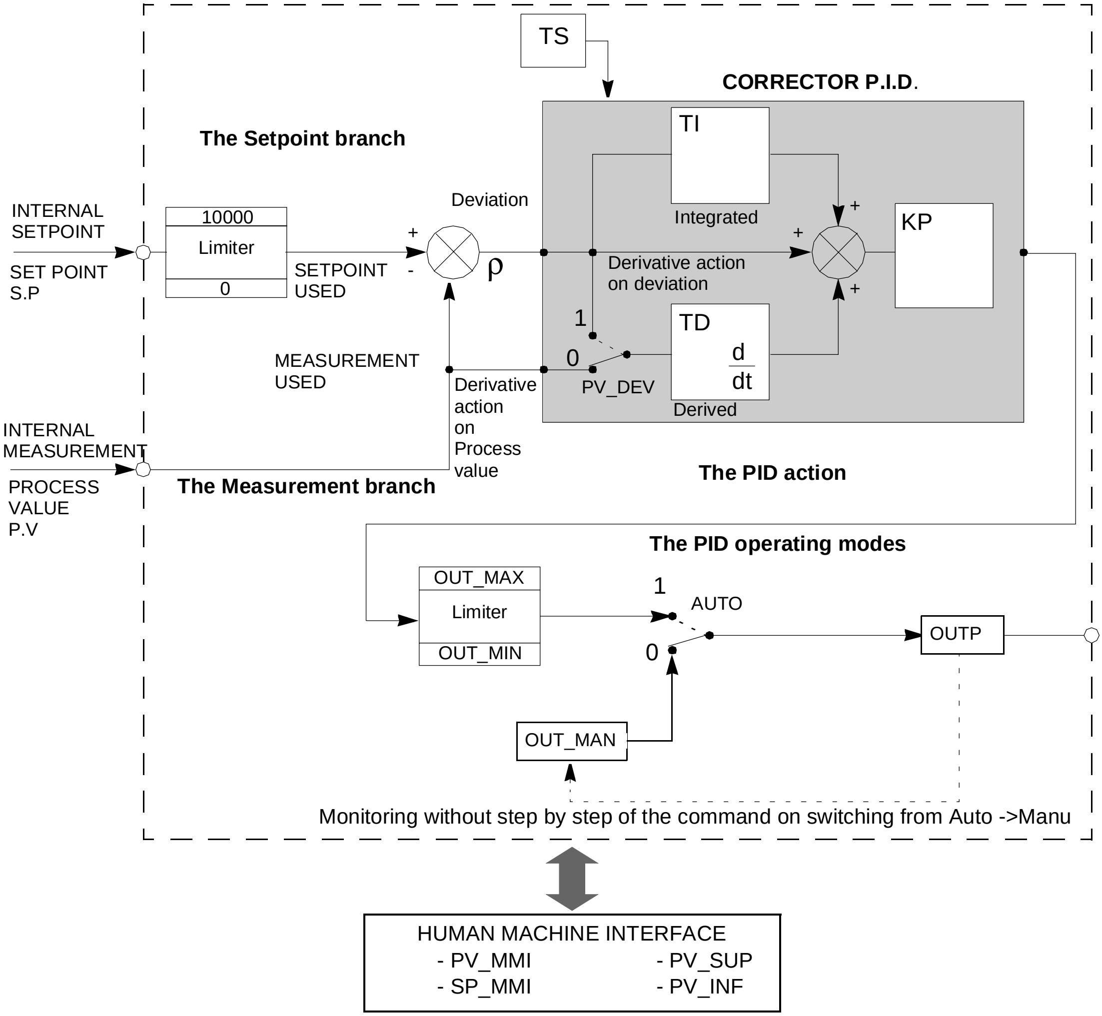
The PID_INT EF comprises the following functions:
- serial / parallel PID algorithm,
- forward / backward action (according to the KP gain sign),
- action derived from measurement or from distance,
- high and low limitation of the setpoint to [0-10000],
- high and low limitation of the output in automatic mode,
- anti-saturation of the integral action,
- Manual/Automatic operating modes without step by step on change,
- PID access control through the Human Machine Interface,
- operating in integrator for (KP = TD =0).

Note:
- The display parameters used by the operator terminal are shown in physical units.
- For a correct PID operation, you must stay within the scale of [0-1000] for the measurement and the setpoint.
- The PID function can be entered in any periodic task (MAST or FAST). The function does not have to be conditioned.

The additional parameters EN and ENO can be configured.

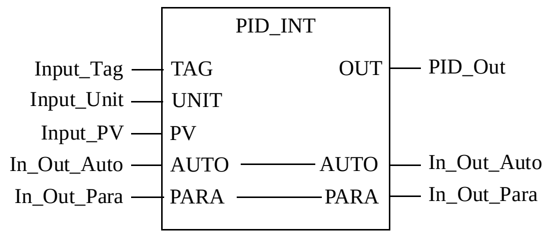
Operating synoptic

The following illustration provides the operating synoptic for the PID function.



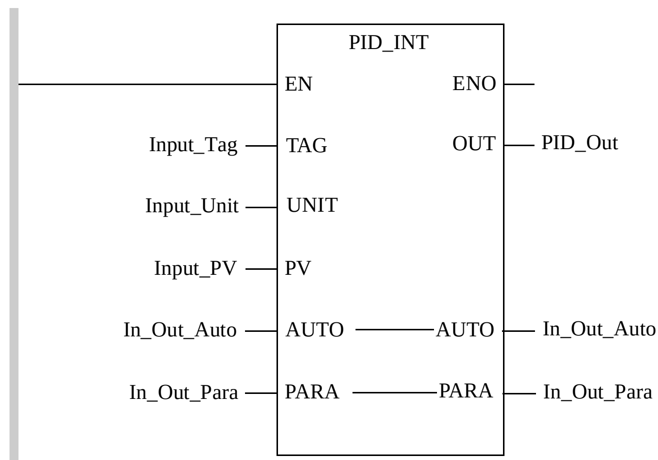
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Input_Tag
PID_INT Input_Unit, Input_PV, In_Out_Auto, In_Out_Para,
PID_Out
```

Representation in ST

Representation:

```
PID_INT(Input_Tag, Input_Unit, Input_PV, In_Out_Auto,
In_Out_Para, PID_Out);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`Input_Tag`	`STRING`	Name of PID used by the terminal. String of 8 characters
`Input_Unit`	`STRING`	Unit of measurement used by the terminal. String of 6 characters
`Input_PV`	`INT`	Process value input Measurement format [0..10000].

The following table describes the input/output parameters:

Parameter	Type	Comment
`In_Out_Auto`	`EBOOL`	Input/output bit which indicates and manages the operating modes of the PID and the terminal: ● 0 : manual, ● 1 : auto.
`In_Out_Para`	ARRAY [n..m] OF `INT`	n and m are positive or negative integers or nil. PID parameter input/output table, the first 16 values of which are described below, the other values being used for internal processing. Table of 43 integers.

The following table describes the output parameters:

Parameter	Type	Comment
`PID_Out`	`INT`	Analog output of PID, if TI = 0, an offset of 5000 is added to the `OUT` output in auto mode. Output format [0;+10000].

Description of Derived Data

Description of PARA Table

The table below presents the different parameters of the PARA table:

Parameter	Rank	Function
SP	**PARA[0]**	Internal setpoint in 0 - 10000 format.
OUT_MAN	**PARA[1]**	Value of the manual output of the PID (between 1 and 1000).
KP	**PARA[2]**	Proportional gain of the PID (x100), signed without unit (-10000<KP<+10000). The Kp sign determines the direction of the PID's action (negative: forward, positive: reverse).
TI	**PARA[3]**	The PID's integral time (between 0 and 20000) is shown in 0.1 seconds.
TD	**PARA[4]**	The PID's derivative time (between 0 and 10000) is shown in 0.1 seconds.
TS	**PARA[5]**	The PID's sampling period (between 1 and 32000) is shown in 0.01 seconds. The real sampling period will be the multiple of the period of the task in which the PID closest to the TS is introduced.
OUT_MAX	**PARA[6]**	Upper limit of the PID's output in automatic (between 0 and 10000).
OUT_MIN	**PARA[7]**	Lower limit of the PID's output in automatic (between 0 and 10000).
PV_DEV	**PARA[8].0**	Derived action choice 0 = on process variable, 1 = on deviation.
NO_BUMP	**PARA[8].4**	Bumpless or non-bumpless mode. 0 = non-bumpless, 1 = bumpless.
DEVAL_MMI	**PARA[8].8**	= 1: inhibits the acknowledgement of the PID by the Human Machine Interface. = 0: the PID is operated by the Human Machine Interface. This bit makes it possible to avoid performing scale conversions on the PIDs not operated by the terminal, and to select the operated PIDs, especially when there are more than 9 PIDs in the application.
PV_SUP*	**PARA[9]**	Upper limit of the measurement scale's range, in a physical unit (x100) (between –9 999 999 and +9 999 999).
PV_INF*	**PARA[11]** **PARA[12]**	These two integers are, respectively, the most significant and least significant of a double integer, that is the lower limit of the measurement scale's range, in a physical unit (x100) (between -9 99 999 and + 9 999 999).

Parameter	Rank	Function
PV_MMI*	**PARA[13]** **PARA[14]**	These two integers are, respectively, the most significant and least significant of a double integer, that is the image of the measurement in a physical unit (x100).
SP_MMI*	**PARA[15]** **PARA[16]**	These two integers are, respectively, the most significant and least significant of a double integer, that is the operator setpoint and image of the setpoint in a physical unit (x100).
* Value used by the operator terminal.		

Note:
- The other parameters that are used by the PID's internal management must never be modified by the application.
- The values used by the terminal are multiplied by 100 in order to support a display with 2 figures after the decimal point on the terminal (the terminal does not use floating point format but supports a fixed comma format).

Rules

There is no internal setpoint alignment on the measurement in manual mode.

The settings on the scale only take place on modification of one of the setpoints (SP or DOP_SP).

The algorithm without the integral action (TI = 0) carries out the following operation:

For	Then the output	With
$\varepsilon t = SP - PV$	OUT = KP [ε t+ Dt] / 100 + 5000	Dt = derived action

The algorithm with the integral action (TI <0) carries out the following operation:

For	Then the output	With
$\varepsilon t = SP - PV$	Δ OUT = KP [$\Delta\varepsilon$t+(TS/10.TI). ε t+ Δ Dt]/100 OUT = OUT + ΔOUT	Dt = derived action

On a cold start, the PID starts off again in manual, with the output at 0. To impose the automatic mode or a manual output that is not at 0 after a cold start, you will have to program the initialization sequence **after** the PID call.

PWM_INT: Pulse width modulation of a numerical value

31

Description

Function description

The PWM_INT function carries out pulse width regulation on a Discrete output. It is a function that formats a PID output.

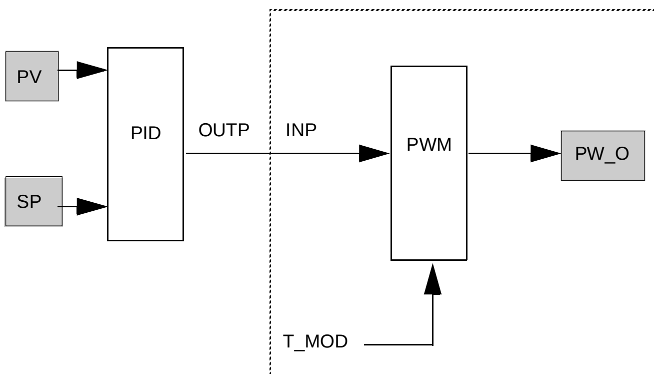
The pulse width depends on the PID's output (The PWM function's INP input) and the modulation period.

> **Note:** the PWM_INT function can be entered in any periodic task (MAST or FAST). The function does not have to be conditioned.

The additional parameters EN and ENO can be configured.

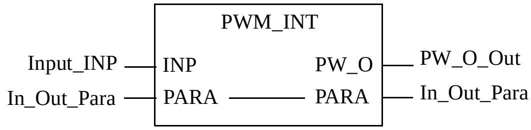
Operating synoptic

The following diagram shows the operating synoptic of the PWM function:



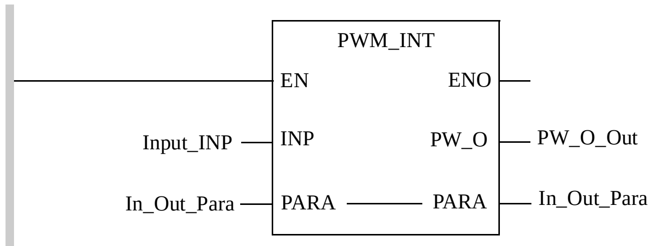
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Input_INP
PWM_INT In_Out_Para, PW_O_Out
```

Representation in ST

Representation:

```
PWM_INT(Input_INP, In_Out_Para, PW_O_Out);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Input_INP	INT	Analog value to be modulated in pulse width (format [0 – 10000]).

The following table describes the input/output parameters:

Parameter	Type	Comment
In_Out_Para	ARRAY [n..m] OF INT	n and m are positive or negative integers or nil. Input/output table of function parameters. The first word corresponds to the parameter T_MOD. Modulation period expressed in 1/100ths of seconds (between 0 and 32767). T_MOD must be greater than or equal to the current task period, and is adjusted by the system to be an integer that is a multiple of this. The following integers are used internally by the function and must never be modified by the application. Table of 5 integers.

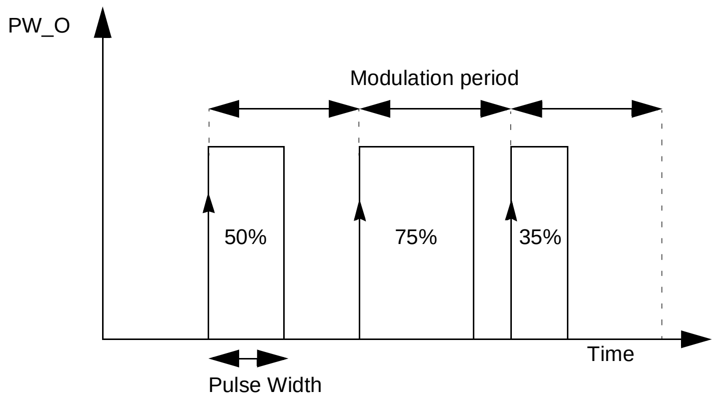
The following table describes the output parameters:

Parameter	Type	Comment
PW_O_Out	EBOOL	Analog output of PID, if TI = 0, an offset of 5000 is added to the OUT output in auto mode.

Pulse widths

To each Top of the T_MOD modulation period, the activation period in 10^{-3} seconds of the PW_O_Out (PW_O) output is calculated according to the following formula:
State 1 of the gap (shown in 10^{-2} seconds) = INP * T_MOD / 1000

The following timing diagram illustrates this formula:



Practical rules

T_MOD = TS (where TS is the sampling period of the upstream PID).
The period of the current task (expressed in 10^{-3} seconds) is equal to:
(Required resolution)* 10 * T_MOD.

The PID is in the MAST task, the MAST's period is $50*10^{-3}$ s, TS = $500*10^{-2}$ s and the required resolution is 1/50 (a T_MOD period must contain at least 50 periods of the current task).

Let T_MOD = TS = 500.

The period of the task in which the PWM is introduced must be less than
500 * 10 / 50 =100 10^{-3} s.
The PWM function can therefore be programmed in the MAST task.
The resolution will be 1/100.

SERVO_INT: Servo drive function

32

Description

Function description

The SERVO_INT function carries out a regulation with a motor-type actuator driven by two Discrete outputs (UP and DOWN).

Note:
- A SERVO_INT function can be entered in any periodic task (MAST or FAST). The function does not have to be conditioned.
- It must be connected in tandem with the analog output of a PID. It cannot be used alone.

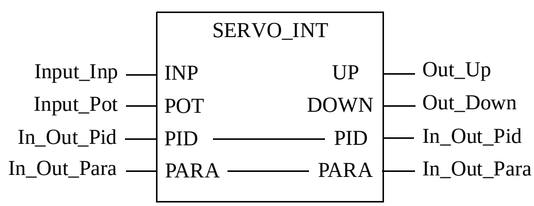
When there is a copy of a position, the valve's position is locked via the Input_Inp (setpoint) and Input_Pot (position measurement) inputs.

When the copy does not physically exist, the algorithm no longer uses the PID's absolute output but the output's variation. The Out_Up output (or Out_Down, according to the variation sign) is set to 1 for a length of time proportional to the actuator opening time and to the variation of the value. Also, the notion of minimum pulse time is introduced.

The additional parameters EN and ENO can be configured.

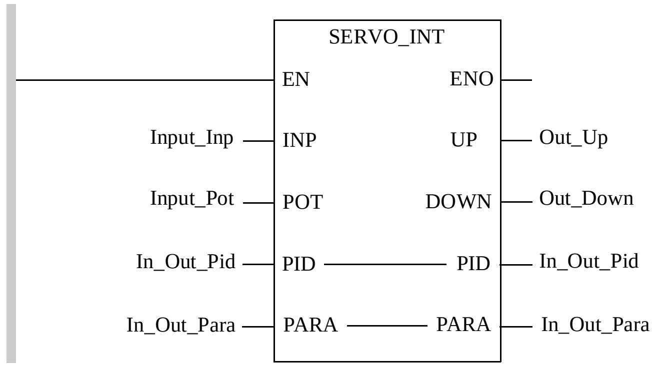
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Input_Inp
SERVO_INT Input_Pot, In_Out_Pid, In_Out_Para, Out_Up, Out_Down
```

Representation in ST

Representation:

```
SERVO_INT(Input_Inp, Input_Pot, In_Out_Pid, In_Out_Para,
Out_Up, Out_Down);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Input_Inp	INT	Position setpoint ([0 -10000] format) that has to be connected to the PID output.
Input_Pot	INT	Position copy, ([0 -10000] format) 0 : closed valve; 10000: open valve. If the copy does not exist, Input_POT must be initialized at -10000. This particular value indicates "no copy".

The following table describes the input/output parameters:

Parameter	Type	Comment
In_Out_Pid	ARRAY [n..m] OF INT	n and m are positive or negative integers or nil. Parameter table of upstream PID (See *Description of PARA Table, p. 130*), used if there are no copy words for the synchronization with the upstream PID. Table of 43 integers.
In_Out_Para	ARRAY [n..m] OF INT	n and m are positive or negative integers or nil. The first three parameters are used if the copy does not exist (Input_POT = -10000): • In_Out_PARA[0] also called T_MOTOR is the valve opening time expressed in 10^{-2} s. • In_Out_PARA[1] also called T_MINI is the minimum pulse expressed in 10^{-2} s. • In_Out_PARA[2] also called HYST is the hysteresis value in [0-10000] format. **Note**: the other parameters that are used by the function's internal management must never be modified by the application. All the parameters are obligatory, regardless of the operating mode. Table of 10 integers.

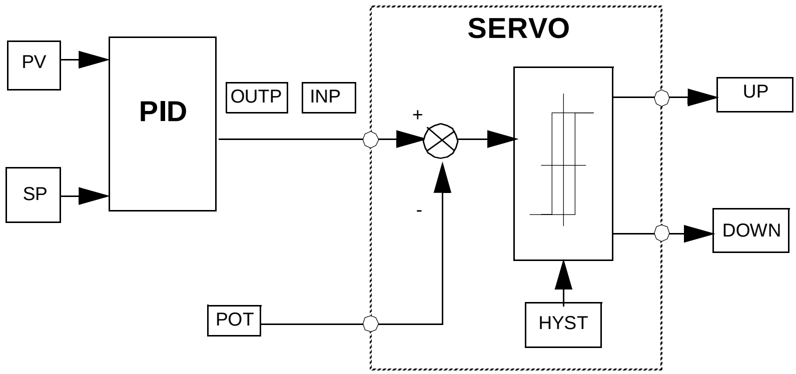
The following table describes the output parameters:

Parameter	Type	Comment
Out_Up	EBOOL	Output signal for the motor's Out_Up operating direction.
Out_Down	EBOOL	Output signal for the motor's Out_Down operating direction.

Principle of operating with a position copy

The SERVO_INT function locks the motor's position according to a setpoint of the Input_Inp (INP) position from a PID's output in [0 - 10000] format, and to a Input_Pot (POT) position measurement.
The locking algorithm is a relay with hysteresis.

In this case, the PID, T_MOTOR and T_MINI parameters are not used.



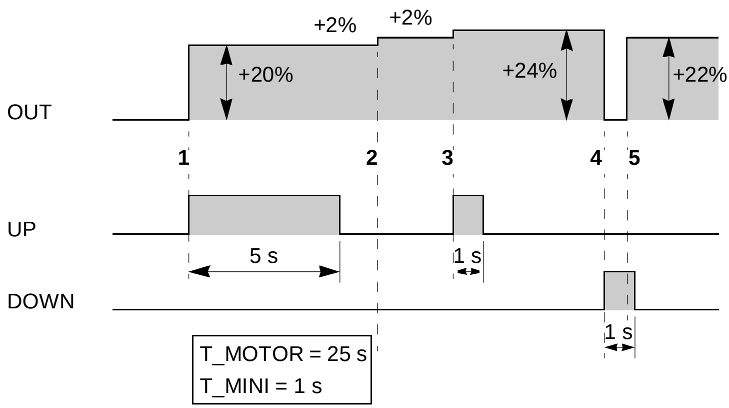
Principle of operating without a position copy (POT= -10000)

In this case, the SERVO_INT function is synchronized with the upstream PID by using the PID parameter table passed on to the SERVO_INT function as a parameter.

The algorithm receives in input the PID's variation output and converts it into pulse period, according to the following formula:
T_IMP (expressed in 10^{-3} s) = OUT x T_MOTOR / 1000

The acquired period is added to the remaining period of the preceding cycles: In fact, what is not "consumed" in a cycle is memorized for the following cycles. This ensures correct operation, especially where there are sudden variations in the command (e.g.: PID setpoint scale division) and in manual mode.

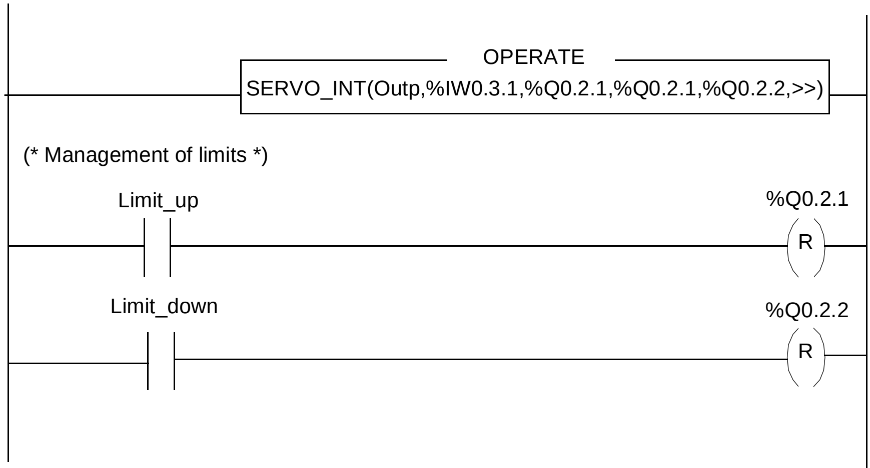
Example The example below is in Ladder language:



Label	Description
1	The PID output variation is +20 % (`T_MOTOR` pulse =25 s for a 100 % variation). In this case, the pulse affects the `UP` output for a period of 5 s.
2	The PID variation is +2 %, which would correspond to a pulse of 0.5 s. This pulse is less than `T_MINI` (=1 s.), and it does not affect the outputs.
3	A second variation of +2 % appears and the function adds this variation to the previous one (which corresponded to a variation less than the minimal value), which corresponds to a positive global variation of +4 %, and therefore to a pulse of 1 s on the `UP` output.
4	A variation of -24 % appears and the activated pulse is therefore of 6 s on the `DOWN` output.
5	Before the following second has elapsed, another variation of +22 % brings the system back to a global variation of 2 % < the variation of `T_MINI` (4 %). The function finishes carrying out the minimal pulse of 1 s.

Note 1: The `SERVO_INT` function does not manage the position limits. These must be managed by the application. If a limit is detected, you must force the corresponding output to 0 (`UP` for the high limit, `DOWN` for the low limit).

Example: in Ladder language



Note 2: It is possible to switch from the operating mode with copy to the mode without copy (for example: when a copy error occurs, go to mode without copy).

Comparison

IV

Introduction

Overview

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

What's in this Part?

This part contains the following chapters:

EQ: Equal to

33

Description

Function description

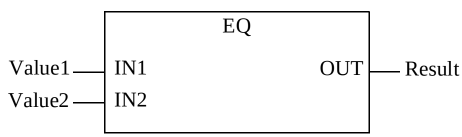
This function checks the inputs for equality, i.e. the output becomes "1" if there is equality at all inputs; otherwise, the output remains at "0".
The data types of all input values must be identical.
The number of inputs can be increased to a maximum of 31.
`EN` and `ENO` can be configured as additional parameters.

Formula

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

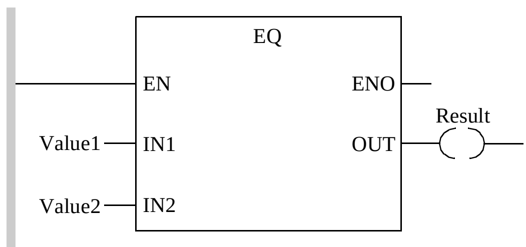
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Value1
EQ Value2
ST Result
```

Representation in ST

Representation:

```
Result := EQ (Value1, Value2) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
Value1	BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD	1. Input
Value2	BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD	2. Input
Valuen	BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD	n. input n = max 31

Description of the output parameter:

Parameter	Data type	Meaning
Result	BOOL	Output

Runtime error

If an unauthorized floating point number is created for an input parameter of data type REAL, the system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1 and the status is placed in %SW17 (See *Description of system words %SW12 to %SW18, p. 451*).

GE: Greater than or equal to

34

Description

Function description

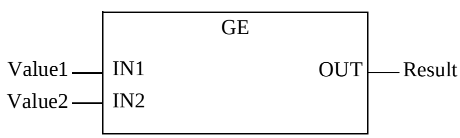
The function checks the values of successive inputs for a decreasing sequence or equality.
The data types of all input values must be identical.
The number of inputs can be increased to a maximum of 31.
When comparing variables of the `BOOL`, `BYTE`, `WORD`, `DWORD`, `INT`, `DINT`, `UINT`, `UDINT`, `REAL`, `TIME`, `DATE`, `DT` and `TOD` data types, the values are compared with each other.
`STRING` variables are compared using the alphabet; variables at the end of the alphabet are higher priority expressions than those at the front.
`EN` and `ENO` can be configured as additional parameters.

Formula

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

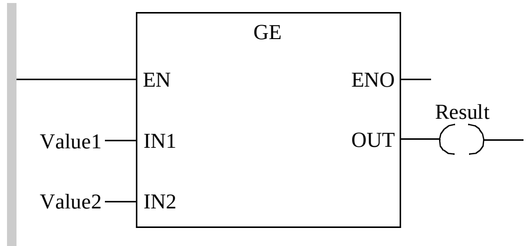
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Value1
GE Value2
ST Result
```

Representation in ST

Representation:

```
Result := GE (Value1, Value2) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
Value1	BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD	1. Input
Value2	BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD	2. Input
Valuen	BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD	n. input n = max 31

Description of the output parameter:

Parameter	Data type	Meaning
Result	BOOL	Output

Runtime error

If an unauthorized floating point number is created for an input parameter of data type `REAL`, the system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1 and the status is placed in %SW17 (See *Description of system words %SW12 to %SW18, p. 451*).

GT: Greater than

35

Description

Function description

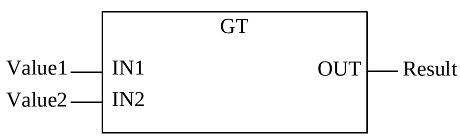
The function checks the values of successive inputs for a decreasing sequence.
The data types of all input values must be identical.
The number of inputs can be increased to a maximum of 31.
When comparing variables of the `BOOL`, `BYTE`, `WORD`, `DWORD`, `INT`, `DINT`, `UINT`, `UDINT`, `REAL`, `TIME`, `DATE`, `DT` and `TOD` data types, the values are compared with each other.
`STRING` variables are compared using the alphabet; variables at the end of the alphabet are higher priority expressions than those at the front.
`EN` and `ENO` can be configured as additional parameters.

Formula

`OUT` = 1, if (`IN1` > `IN2`) & (`IN2` > `IN3`) & .. ($\mathrm{IN}_{(n-1)}$ > IN_n)

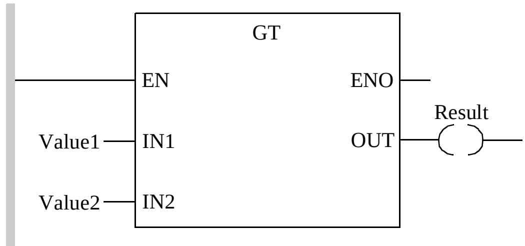
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Value1
GT Value2
ST Result
```

Representation in ST

Representation:

```
Result := GT (Value1, Value2) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
`Value1`	`BOOL`, `BYTE`, `WORD`, `DWORD`, `STRING`, `INT`, `DINT`, `UINT`, `UDINT`, `REAL`, `TIME`, `DATE`, `DT`, `TOD`	1. Input
`Value2`	`BOOL`, `BYTE`, `WORD`, `DWORD`, `STRING`, `INT`, `DINT`, `UINT`, `UDINT`, `REAL`, `TIME`, `DATE`, `DT`, `TOD`	2. Input
`Valuen`	`BOOL`, `BYTE`, `WORD`, `DWORD`, `STRING`, `INT`, `DINT`, `UINT`, `UDINT`, `REAL`, `TIME`, `DATE`, `DT`, `TOD`	n. input n = max 31

Description of the output parameter:

Parameter	Data type	Meaning
`Result`	`BOOL`	Output

Runtime error

If an unauthorized floating point number is created for an input parameter of data type `REAL`, the system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1 and the status is placed in %SW17 (See *Description of system words %SW12 to %SW18, p. 451*).

LE: Less than or equal to

36

Description

Function description

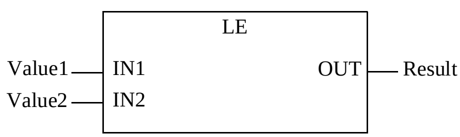
The function checks the values of successive inputs for an increasing sequence or equality.
The data types of all input values must be identical.
The number of inputs can be increased to a maximum of 31.
When comparing variables of the `BOOL`, `BYTE`, `WORD`, `DWORD`, `INT`, `DINT`, `UINT`, `UDINT`, `REAL`, `TIME`, `DATE`, `DT` and `TOD` data types, the values are compared with each other.
`STRING` variables are compared using the alphabet; variables at the end of the alphabet are higher priority expressions than those at the front.
`EN` and `ENO` can be configured as additional parameters.

Formula

$\mathtt{OUT} = 1$, if $(\mathtt{IN1} \le \mathtt{IN2})$ & $(\mathtt{IN2} \le \mathtt{IN3})$ & .. & $(\mathtt{IN}_{(n-1)} \le \mathtt{IN}_n)$

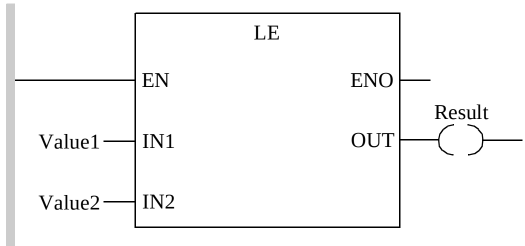
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Value1
LE Value2
ST Result
```

Representation in ST

Representation:

```
Result := LE (Value1, Value2) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
Value1	BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD	1. Input
Value2	BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD	2. Input
Valuen	BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD	n. input n = max 31

Description of the output parameter:

Parameter	Data type	Meaning
Result	BOOL	Output

Runtime error

If an unauthorized floating point number is created for an input parameter of data type `REAL`, the system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1 and the status is placed in %SW17 (See *Description of system words %SW12 to %SW18, p. 451*).

LT: Less than

37

Description

Function description

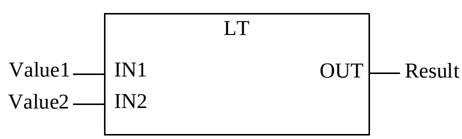
The function checks the values of successive inputs for an increasing sequence.
The data types of all input values must be identical.
The number of inputs can be increased to a maximum of 31.
When comparing variables of the `BOOL`, `BYTE`, `WORD`, `DWORD`, `INT`, `DINT`, `UINT`, `UDINT`, `REAL`, `TIME`, `DATE`, `DT` and `TOD` data types, the values are compared with each other.
`STRING` variables are compared using the alphabet; variables at the end of the alphabet are higher priority expressions than those at the front.
`EN` and `ENO` can be configured as additional parameters.

Formula

$\mathtt{OUT} = 1$, if $(\mathtt{IN1} < \mathtt{IN2})$ & $(\mathtt{IN2} < \mathtt{IN3})$ & .. & $(\mathtt{IN}_{(n-1)} < \mathtt{IN}_{n})$

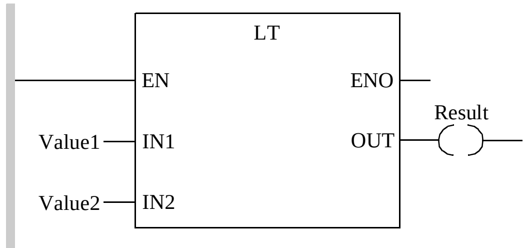
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Value1
LT Value2
ST Result
```

Representation in ST

Representation:

```
Result := LT (Value1, Value2) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
`Value1`	`BOOL`, `BYTE`, `WORD`, `DWORD`, `STRING`, `INT`, `DINT`, `UINT`, `UDINT`, `REAL`, `TIME`, `DATE`, `DT`, `TOD`	1. Input value
`Value2`	`BOOL`, `BYTE`, `WORD`, `DWORD`, `STRING`, `INT`, `DINT`, `UINT`, `UDINT`, `REAL`, `TIME`, `DATE`, `DT`, `TOD`	2. Input value
`Valuen`	`BOOL`, `BYTE`, `WORD`, `DWORD`, `STRING`, `INT`, `DINT`, `UINT`, `UDINT`, `REAL`, `TIME`, `DATE`, `DT`, `TOD`	n. input value n = max 31

Description of the output parameter:

Parameter	Data type	Meaning
`Result`	`BOOL`	Output value

Runtime error

If an unauthorized floating point number is created for an input parameter of data type `REAL`, the system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1 and the status is placed in %SW17 (See *Description of system words %SW12 to %SW18, p. 451*).

NE: Not equal to

38

Description

Function description

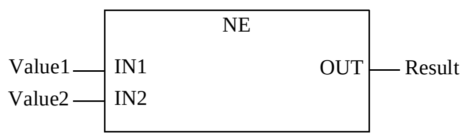
The function checks the input values for inequality.
The data types of the input values must be identical.
`EN` and `ENO` can be configured as additional parameters.

Formula

`OUT` = 1, if `IN1` < > `IN2`

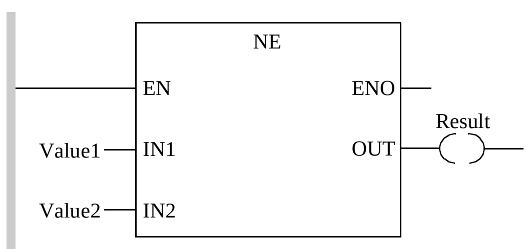
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Value1
NE Value2
ST Result
```

Representation in ST

Representation:

```
Result := NE (Value1, Value2) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
Value1	BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD	1. Input
Value2	BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD	2. Input

Description of the output parameter:

Parameter	Data type	Meaning
Result	BOOL	Output

Runtime error

If an unauthorized floating point number is created for an input parameter of data type REAL, the system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1 and the status is placed in %SW17 (See *Description of system words %SW12 to %SW18, p. 451*).

Date & Time

V

Introduction

Overview

This section describes the elementary functions and elementary function blocks of the `Date & Time` family.

What's in this Part?

This part contains the following chapters:

ADD_***_TIME: Addition of a duration to a date

39

Description

Function description

The ADD_***_TIME function adds a duration to a date or a time.

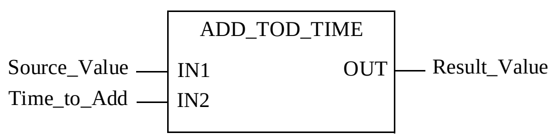
The additional parameters EN and ENO can be configured.

Available functions

The available functions are as follows:
- ADD_DT_TIME,
- ADD_TOD_TIME.

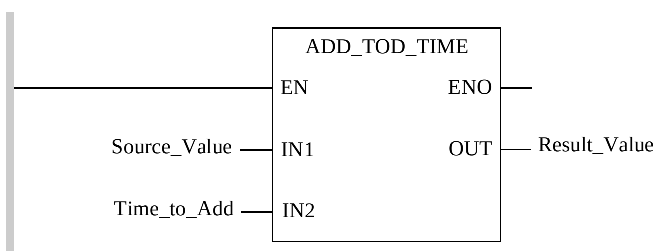
Representation in FBD

Representation applied to a time of day:



Representation in LD

Representation applied to a time of day:



Representation in IL

Representation applied to a time of day:

```
LD Source_Value
ADD_TOD_TIME Time_to_Add
ST Result_Value
```

Representation in ST

Representation applied to a time of day:

```
Result_Value:= ADD_TOD_TIME(Source_Value, Time_to_Add);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`Source_Value`	`DT`, `TOD`	Date or time.
`Time_to_Add`	`TIME`	Duration to be added to `Source_Value` **Note**: this duration is expressed in TIME format with a precision to the order of tenths of a second. As the types DT and TOD are expressed to the nearest second, `Time_to_Add` is rounded off to the second.

The following table describes the output parameters:

Parameter	Type	Comment
`Result_Value`	`DT`, `TOD`	`Result_Value` is of the same type as `Source_Value`.

Note: the management of leap years is to be provided for in the application.

Runtime errors

For the type TOD, there is a change of day if `Result_Value` is outside the authorized values. In this case, the system bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) is set at 1 and the value of `Result_Value` is only significant with a modulo 24:00:00.

For the type DT, if `Result_Value` is outside the interval of authorized values, the system bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) is set to 1 and the value of `Result_Value` is equal to the maximum limit.

If one of the input parameters cannot be interpreted and is inconsistent with the function format then the system bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) is set at 1 and `Result_Value` applies:

- 00:00:00 for the type TOD.
- 00001-01-01-00:00:00 for the type DT.

DIVTIME: Division

40

Description

Function description

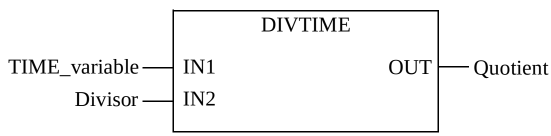
The function divides the value at the `TIME_variable` (data type `TIME`) input with the value at the `Divisor` input and assigns the result to the output.
`EN` and `ENO` can be configured as additional parameters.

Formula

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

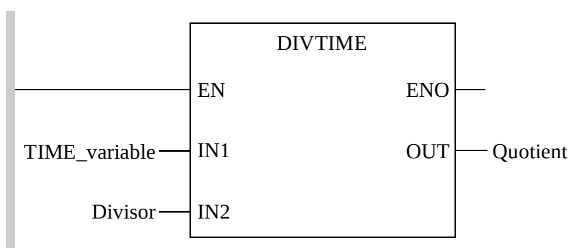
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD TIME_variable
DIVTIME Divisor
ST Quotient
```

Representation in ST

Representation:

```
Quotient := DIVTIME (TIME_variable, Divisor) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
TIME_variable	TIME	Dividend
Divisor	INT, DINT, UINT, UDINT, REAL	Divisor

Description of output parameters:

Parameter	Data type	Meaning
Quotient	TIME	Quotient

Runtime error

The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1, if

- an invalid division by 0 is executed (all available data types)
  or
- an unauthorized floating point number is set at an input parameter of data type REAL. In this case, the status is also placed in %SW17 (See *Description of system words %SW12 to %SW18, p. 451*).

Note: For a list of all block error codes and values, see *Date & Time, p. 442*.

MULTIME: Multiplication

41

Description

Function description

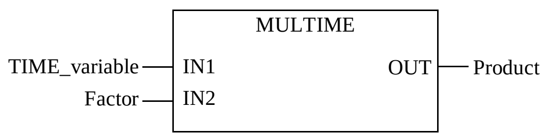
The function multiplies the input values and assigns the result to the output.
The data type of the 1st input value (`TIME_variable`) must be a `TIME` data type.
`EN` and `ENO` can be configured as additional parameters.

Formula

`OUT` = `IN1` x `IN2`

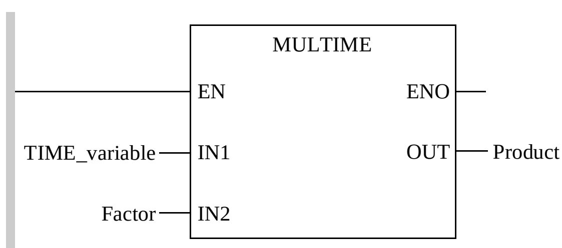
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD TIME_variable
MULTIME Factor
ST Product
```

Representation in ST

Representation:

```
Product := MULTIME (TIME_variable, Factor) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
`TIME_variable`	`TIME`	Multiplicand (factor)
`Factor`	`INT`, `DINT`, `UINT`, `UDINT`, `REAL`	Multiplier (factor)

Description of output parameters:

Parameter	Data type	Meaning
`Product`	`TIME`	Product

Runtime error

The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1, if

- the value range at the output has been exceeded (all available data types) or
- an unauthorized floating-point number is set at an input parameter of data type `REAL`. In this case, the status is also placed in %SW17 (See *Description of system words %SW12 to %SW18, p. 451*).

Note: For a list of all block error codes and values, see *Date & Time, p. 442*.

SUB_***_***: Calculates the time difference between two dates or times

42

Description

Function description

The SUB_***_*** function calculates the time difference between two dates or times.

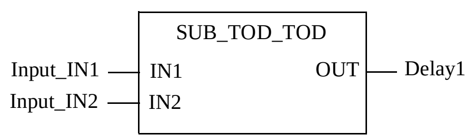
The additional parameters EN and ENO can be configured.

Available functions

The available functions are as follows:
- SUB_DATE_DATE,
- SUB_DT_DT,
- SUB_TOD_TOD.

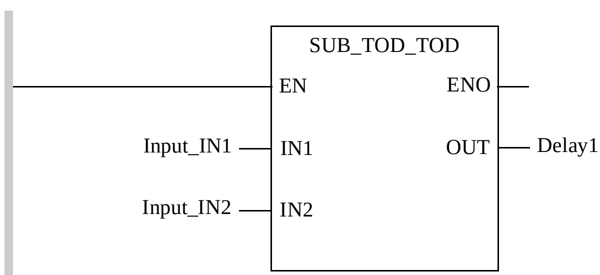
Representation in FBD

Representation applied to a time of day:



Representation in LD

Representation applied to a time of day:



Representation in IL

Representation applied to a time of day:

```
LD Input_IN1
SUB_TOD_TOD Input_IN2
ST Delay1
```

Representation in ST

Representation applied to a time of day:

```
Delay1:= =SUB_TOD_TOD(Input_IN1, Input_IN2);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Input_IN1	DATE, DT, TOD	Date or time for which we wish to calculate the difference with Input_IN2.
Input_IN2	DATE, DT, TOD	Date or time for which we wish to calculate the difference with Input_IN2. Of the same type as the elements of the table Input_IN1.

Note: Input_IN1 and Input_IN2 must be of the same type.

The following table describes the output parameters:

Parameter	Type	Comment
Delay1	TIME	Delay1 contains the time expressed as an absolute value elapsed between the two entries Input_IN1 and Input_IN2.

Runtime errors

If Delay1 exceeds the maximum value allowed for a TIME format, there is overrun, then Delay1 = 0 and the system bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) is set to 1.

If one of the input parameters is not interpretable and coherent in the function format, then Delay1 = 0 and the system bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) is set to 1.

SUB_***_TIME: Subtraction of a duration from a date

43

Description

Function description

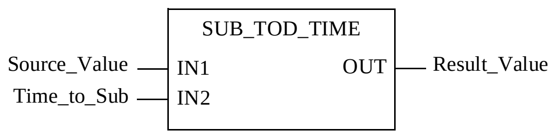
The `SUB_***_TIME` function removes a duration from a date or a time.
The additional parameters `EN` and `ENO` can be configured.

Available functions

The available functions are as follows:
- `SUB_DT_TIME`,
- `SUB_TOD_TIME`.

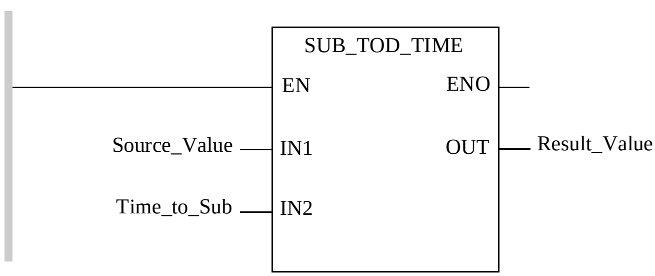
Representation in FBD

Representation applied to a time of day:



Representation in LD

Representation applied to a time of day:



Representation in IL

Representation applied to a time of day:

```
LD Source_Value
SUB_TOD_TIME Time_to_Sub
ST Result_Value
```

Representation in ST

Representation applied to a time of day:

```
Result_Value:= SUB_TOD_TIME(Source_Value, Time_to_Sub);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`Source_Value`	`DT`, `TOD`	Date or time.
`Time_to_Sub`	`TIME`	Duration to subtract from `Source_Value` **Note**: this duration is expressed in TIME format (with a precision to the order of tenths of a second). As the types DT and TOD are expressed to the nearest second, `Time_to_Sub` is rounded off to the second.

The following table describes the output parameters:

Parameter	Type	Comment
`Result_Value`	`DT`, `TOD`	`Result_Value` is of the same type as `Source_Value`.

Note: the management of leap years is to be provided for in the application.

Runtime errors

For the type TOD, there is a change of day if `Result_Value` is outside the interval of authorized values. In this case the system bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) is set at 1 and the value of `Result_Value` is only significant with a modulo 24:00:00.

For the type DT, if `Result_Value` is outside the interval of authorized values, the system bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) is set to 1 and the value of `Result_Value` is equal to the minimum limit.

If one of the input parameters cannot be interpreted and is inconsistent with the function format then the system bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) is set at 1 and `Result_Value` applies:

- 00:00:00 for the type TOD.
- 00001-01-01-00:00:00 for the type DT.

Logic

VI

Introduction

Overview

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

What's in this Part?

This part contains the following chapters:

AND: AND function

44

Description

Function description

The function for a bit-by-bit AND link of the bit sequences at the inputs and assigns the result to the output.
The data types of all input values and output values must be identical.
The number of inputs can be increased to a maximum of 32.
`EN` and `ENO` can be configured as additional parameters.

Further available functions

When using a Premium PLC, the following functions are also available in the Obsolete library:

- `AND_DINT`
- `AND_INT`

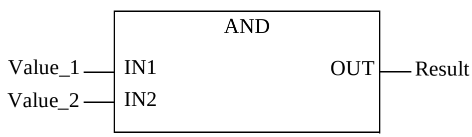
The functionality of these functions is identical to the function `AND`.

Formula

`OUT` = `IN1` & `IN2` & `INn`

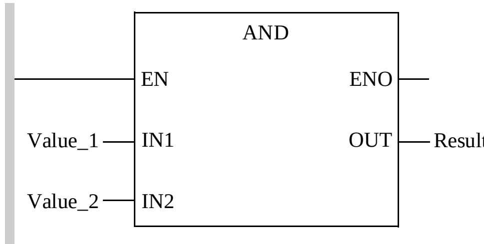
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Value_1
AND Value_2
ST Result
```

Representation in ST

Representation:

```
Result := AND (Value_1, Value_2) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
Value_1	BOOL, BYTE, WORD, DWORD	Input bit sequence
Value_2	BOOL, BYTE, WORD, DWORD	Input bit sequence
Value_n	BOOL, BYTE, WORD, DWORD	Input bit sequence (n = max. 32)

Description of the output parameter:

Parameter	Data type	Meaning
Result	BOOL, BYTE, WORD, DWORD	Output bit sequence

F_TRIG: Falling edge detection

45

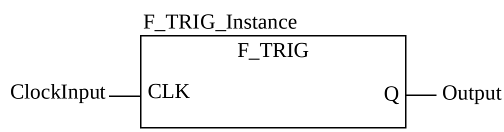
Description

Function description

This function block is used for the detection of falling edges 1 -> 0.
Output `Q` becomes "1" if there is a transition from "1" to "0" at the `CLK` input. The output will remain at "1" from one function block execution to the next; the output subsequently returns to "0".
`EN` and `ENO` can be configured as additional parameters.

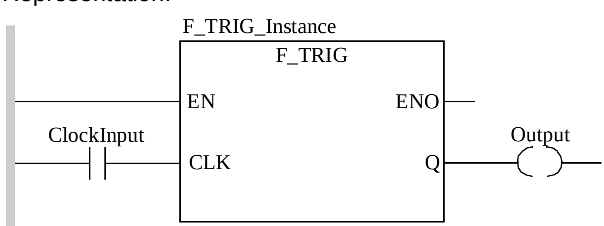
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL F_TRIG_Instance (CLK:=ClockInput, Q=>Output)
```

Representation in ST

Representation:

```
F_TRIG_Instance (CLK:=ClockInput, Q=>Output) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
CLK	BOOL	Clock input

Description of the output parameter:

Parameter	Data type	Meaning
Q	BOOL	Output

46

FE: Detection of Falling Edge

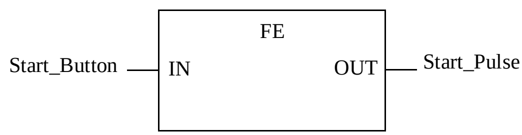
Description

Function description

The FE function detects the passage from 1 to 0 (Falling Edge) of the bit associated with it.
The additional parameters EN and ENO can be configured.

Representation in FBD

Representation:



Representation in LD

Representation:

Representation in IL

Representation:

```
LD Start_Button
FE
ST Start_Pulse
```

Representation in ST

Representation:

```
Start_Pulse:= FE (Start_Button);
```

Description of parameters

The following table describes the input parameters:

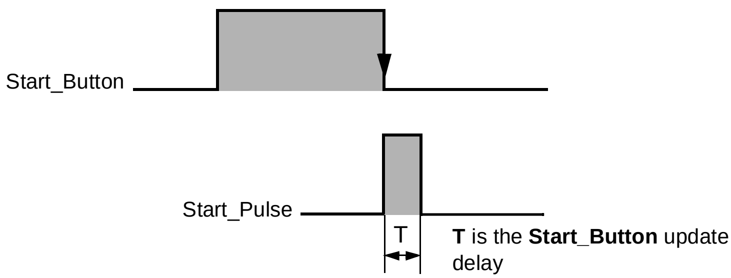
Parameter	Type	Comment
Start_Button	EBOOL	Discrete input or output or internal bit whose Falling Edge we wish to detect.

The following table describes the output parameters:

Parameter	Type	Comment
Start_Pulse	BOOL	Internal bit or output representing the Falling Edge.

Trend diagram

Timing diagram:



T T is equal to a PLC cycle time for an input and is the delay between two assignments for a discrete output or an internal bit.

NOT: Negation

47

Description

Function description

The function negates the input bit sequence bit-by-bit and assigns the result to the output.
The data types of the input and output values must be identical.
`EN` and `ENO` can be configured as additional parameters.

Further available functions

The Obsolete library provides the following additional functions:
- `NOT_DINT`
- `NOT_INT`

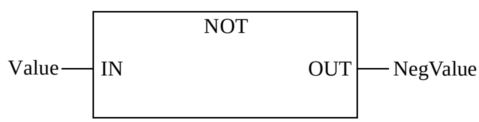
The functionality of these functions is identical to the function `NOT`.

Formula

`OUT` = NOT `IN`

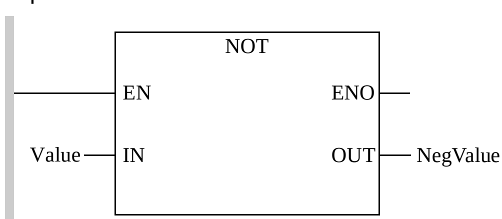
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Value
NOT
ST NegValue
```

Representation in ST

Representation:

```
NegValue := NOT (Value) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
`Value`	`BOOL`, `BYTE`, `WORD`, `DWORD`	Input bit sequence

Description of the output parameter:

Parameter	Data type	Meaning
`NegValue`	`BOOL`, `BYTE`, `WORD`, `DWORD`	Negated bit sequence

OR: OR function

48

Description

Function description

The function for a bit OR link of the bit sequences at the inputs and returns the result at the output.
The data types of all input values and output values must be identical.
The number of inputs can be increased to a maximum of 32.
`EN` and `ENO` can be configured as additional parameters.

Further available functions

When using a Premium PLC, the following functions are also available in the Obsolete library:

- `OR_DINT`
- `OR_INT`

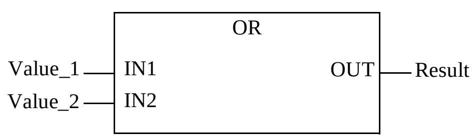
The functionality of these functions is identical to the function `OR`.

Formula

`OUT` = `IN1` OR `IN2` OR .. OR `INn`

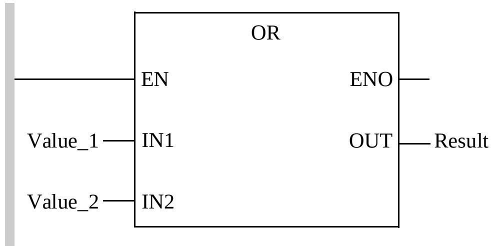
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Value_1
OR Value_2
ST Result
```

Representation in ST

Representation:

```
Result := OR (Value_1, Value_2) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
Value_1	BOOL, BYTE, WORD, DWORD	Input bit sequence
Value_2	BOOL, BYTE, WORD, DWORD	Input bit sequence
Value_n	BOOL, BYTE, WORD, DWORD	Input bit sequence n = max. 32

Description of the output parameter:

Parameter	Data type	Meaning
Result	BOOL, BYTE, WORD, DWORD	Output bit sequence

R_TRIG: Rising edge detection

49

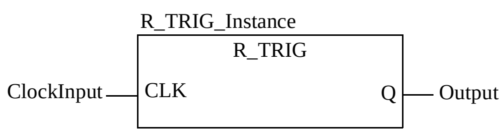
Description

Function description

This function block is used for the detection of rising edges 0 -> 1.
Output `Q` becomes "1" if there is a transition from "0" to "1" at the `CLK` input. The output remains at "1" from one function block execution to the next (one cycle); the output subsequently returns to "0".
`EN` and `ENO` can be configured as additional parameters.

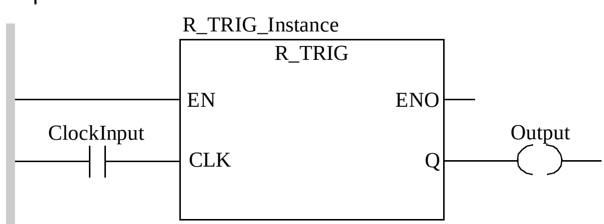
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL R_TRIG_Instance (CLK:=ClockInput, Q=>Output)
```

Representation in ST

Representation:

```
R_TRIG_Instance (CLK:=ClockInput, Q=>Output) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
CLK	BOOL	Clock input

Description of the output parameter:

Parameter	Data type	Meaning
Q	BOOL	Output

RE: Detection of Rising Edge

50

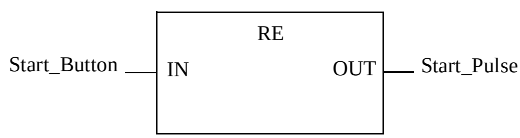
Description

Function description

The `RE` function detects the passage from 0 to 1(Rising Edge) of the bit associated with it.
The additional parameters `EN` and `ENO` can be configured.

Representation in FBD

Representation:



Representation in LD

Representation:

Representation in IL

Representation:

```
LD Start_Button
RE
ST Start_Pulse
```

Representation in ST

Representation:

```
Start_Pulse := RE (Start_Button);
```

Description of parameters

The following table describes the input parameters:

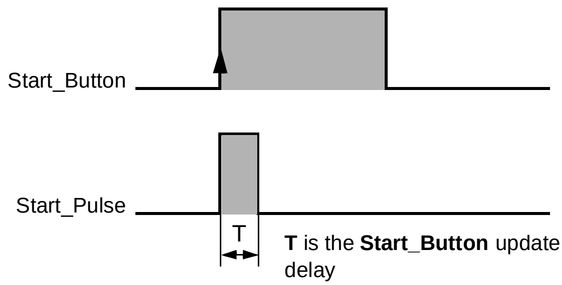
Parameter	Type	Comment
`Start_Button`	`EBOOL`	Discrete input or output, internal bit whose Rising Edge we wish to detect

The following table describes the output parameters:

Parameter	Type	Comment
`Start_Pulse`	`BOOL`	Internal bit or output representing the Rising Edge.

Trend diagram

Timing diagram:



T T is equal to a PLC cycle time for an input and is the delay between two assignments for a discrete output or an internal bit.

RESET: Setting of a bit to 0

51

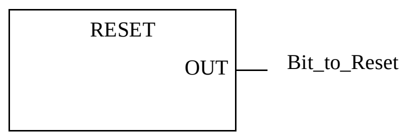
Description

Function description

The `RESET` function sets the bit associated with it to zero.

Representation in FBD

Representation:



Representation in LD

Representation:

Representation in IL

Representation:

```
CAL RESET(Bit_to_Reset)
```

Representation in ST

Representation:

```
RESET (Bit_to_Reset);
```

Description of parameters

The following table describes the output parameters:

Parameter	Type	Comment
Bit_to_Reset	BOOL	Discrete input or output or internal bit we wish to set to 0.

ROL: Rotate left

52

Description

Function description

This function rotates the bit pattern at the IN input circularly to the left by n bits (value at input Number).
System bit %S17 is used as CARRY bit, i.e. the status of the bit that is shifted out is stored there.
The data types of the IN input and OUT output must be identical.

> **Note:** Because of IEC 61131-3 conformity, this function also works with the BOOL data type. This is not significant here.

EN and ENO can be configured as additional parameters.

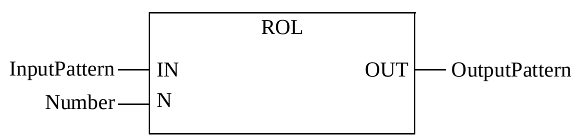
Further available functions

When using a Premium PLC, the following functions are also available in the Obsolete library:
- ROL_DINT
- ROL_INT

The functionality of these functions is identical to the function ROL.

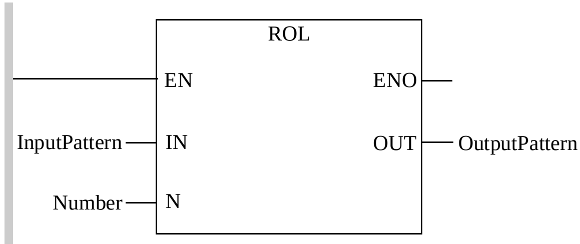
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD InputPattern
ROL Number
ST OutputPattern
```

Representation in ST

Representation:

```
OutputPattern := ROL (InputPattern, Number) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
InputPattern	For ROL: BOOL, BYTE, WORD, DWORD For ROL_INT: INT For ROL_DINT: DINT	this is the bit pattern to be rotated
Number	For ROL: UINT For ROL_INT, ROL_DINT: INT	this is the number of spaces to be rotated

Description of the output parameter:

Parameter	Data type	Meaning
OutputPattern	For ROL: BOOL, BYTE, WORD, DWORD For ROL_INT: INT For ROL_DINT: DINT	this is the bit pattern rotated

ROR: Rotate right

53

Description

Function description

This function rotates the bit pattern at the `In` input circularly to the right by n bits (value at input `Number`).
System bit %S17 is used as CARRY bit, i.e. the status of the bit that is shifted out is stored there.
The data types of the `IN` input and `OUT` output must be identical.

> **Note:** Because of IEC 61131-3 conformity, this function also works with the `BOOL` data type. This is not significant here.

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

Further available functions

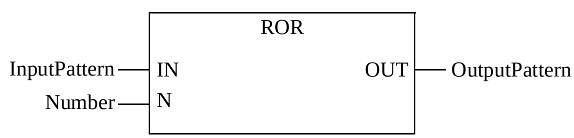
When using a Premium PLC, the following functions are also available in the Obsolete library:

- `ROR_DINT`
- `ROR_INT`

The functionality of these functions is identical to the function `ROR`.

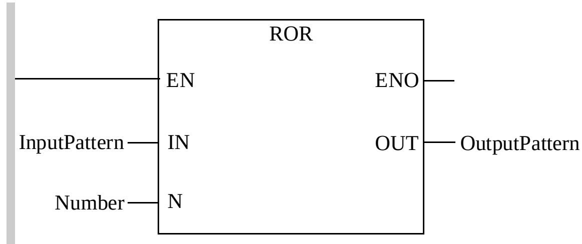
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD InputPattern
ROR Number
ST OutputPattern
```

Representation in ST

Representation:

```
OutputPattern := ROR (InputPattern, Number) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
`InputPattern`	For `ROR`: `BOOL`, `BYTE`, `WORD`, `DWORD` For `ROR_INT`: `INT` For `ROR_DINT`: `DINT`	this is the bit pattern to be rotated
`Number`	For `ROR`: `UINT` For `ROR_INT`, `ROR_DINT`: `INT`	this is the number of spaces to be rotated

Description of the output parameter:

Parameter	Data type	Meaning
`OutputPattern`	For `ROR`: `BOOL`, `BYTE`, `WORD`, `DWORD` For `ROR_INT`: `INT` For `ROR_DINT`: `DINT`	this is the bit pattern rotated

RS: Bistable function block, reset dominant

54

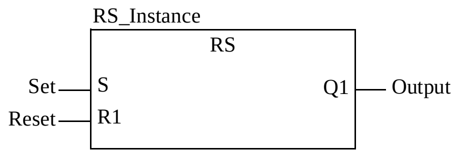
Description

Function description

The function block is used as `RS` memory with the property "Reset dominant". Output `Q1` becomes "1" when the `S` input becomes "1". This state remains even if input `S` reverts back to "0". Output `Q1` changes back to "0" when input `R1` becomes "1". If the inputs `S` and `R1` are "1" simultaneously, the dominating input `R1` will set the output `Q1` to "0".
When the function block is called for the first time, the initial state of `Q1` is "0".
`EN` and `ENO` can be configured as additional parameters.

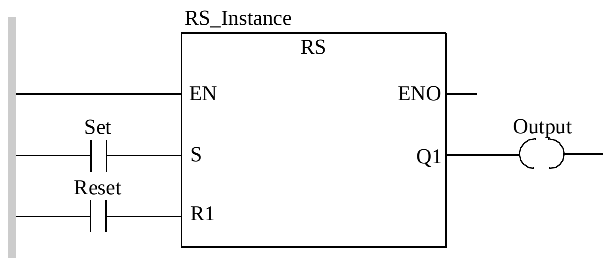
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL RS_Instance (S:=Set, R1:=Reset, Q1=>Output)
```

Representation in ST

Representation:
```
RS_Instance (S:=Set, R1:=Reset, Q1=>Output) ;
```

Parameter description

Description of the input parameters:

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

Description of the output parameter:

Parameter	Data type	Meaning
Q1	BOOL	Output

SET: Setting of a bit to 1

55

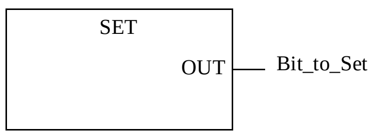
Description

Function description

The `SET` function sets the bit associated with it to 1.

Representation in FBD

Representation:



Representation in LD

Representation:

Representation in IL

Representation:

```
CAL SET(Bit_to_Set)
```

Representation in ST

Representation:

```
SET (Bit_to_Set);
```

Description of parameters

The following table describes the output parameters:

Parameter	Type	Comment
`Bit_to_Set`	BOOL	Discrete input or output or internal bit we wish to set to 1.

SHL: Shift left

56

Description

Function description

This function shifts the bit pattern at the `IN` input to the left by n bits (value at input `N`). System bit %S17 is used as CARRY bit, i.e. the status of the bit that is shifted out is stored there.
Zeros are filled in from the right.
The data types of the `IN` input and `OUT` output must be identical.

> **Note:** Because of IEC 61131-3 conformity, this function also works with the `BOOL` data type. This is not significant here.

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

Further available functions

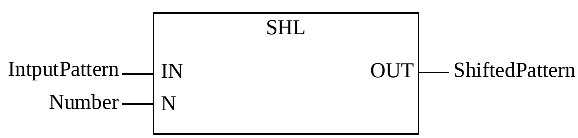
Additionally, the Obsolete library provides the following functions:

- `SHL_DINT`
- `SHL_INT`

The functionality of these functions is identical to the function `SHL`.

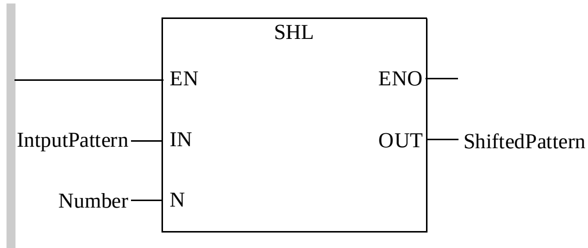
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD IntputPattern
SHL Number
ST ShiftedPattern
```

Representation in ST

Representation:

```
ShiftedPattern := SHL (IntputPattern, Number) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
IntputPattern	For SHL: BOOL, BYTE, WORD, DWORD For SHL_INT: INT For SHL_DINT: DINT	this is the bit pattern to be shifted **For example:** IntputPattern = 2#010000001111**0001**.
Number	For SHL: UINT For SHL_INT, SHL_DINT: INT	this is the number of spaces to be shifted **Example:** Number = 4.

Description of the output parameter:

Parameter	Data type	Meaning
ShiftedPattern	For SHL: BOOL, BYTE, WORD, DWORD For SHL_INT: INT For SHL_DINT: DINT	this is the bit pattern shifted **For example:** with the data from the previous table, the result is: ShiftedPattern = 2#000011100001**0000**

57

SHR: Shift right

Description

Function description

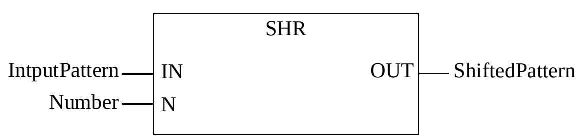
This function shifts the bit pattern at the IN input to the right by n bits (value at input N).
System bit %S17 is used as CARRY bit, i.e. the status of the bit that is shifted out is stored there.
Zeros are filled in from the left.
Special case: If in the dialog box **Tools** → **Project Settings** → **Language extensions**, the option **INT/DINT is activated instead of ANY_BIT valid** and input IN uses data types INT or DINT, then zeros are filled in from the left when the most significant bit is 0. If the most significant bit is 1, ones are filled in. The most significant bit contains the sign bit for data types INT and DINT. This guarantees that the sign is not lost when shifting. If the sign is not to be considered and zeros are always filled in, the function SHRZ_*** from the Obsolete library can be used for Premium controllers.
The data types of the In input and OUT output must be identical.

> **Note:** Because of IEC 61131-3 conformity, this function also works with the BOOL data type. This is not significant here.

EN and ENO can be configured as additional parameters.

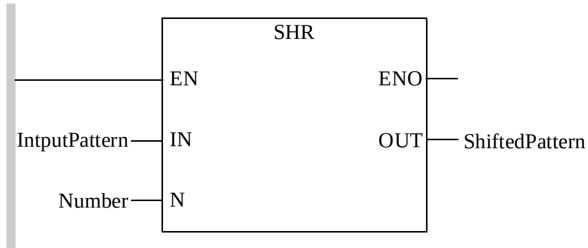
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD IntputPattern
SHR Number
ST ShiftedPattern
```

Representation in ST

Representation:

```
ShiftedPattern := SHR (IntputPattern, Number) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
`IntputPattern`	`BOOL`, `BYTE`, `WORD`, `DWORD`	this is the bit pattern to be shifted **For example:** `IntputPattern` = 2#010000001111**0001**.
`Number`	`UINT`	this is the number of spaces to be shifted **Example:** Number = 4.

Description of the output parameter:

Parameter	Data type	Meaning
`ShiftedPattern`	`BOOL`, `BYTE`, `WORD`, `DWORD`	this is the bit pattern shifted **For example:** with the data from the previous table, the result is: ShiftedPattern = 2#**0000**010000001111

SR: Bistable function block, set dominant

58

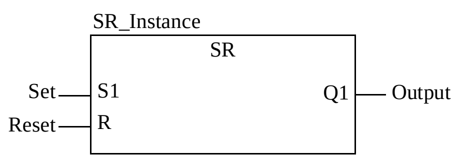
Description

Function description

The function block is used as SR memory with the property "Set dominant". Output Q1 becomes "1" when the S1 input becomes "1". This state remains even if input S1 reverts back to "0". Output Q1 changes back to "0" when input R becomes "1". If the inputs S1 and R are both "1" simultaneously, the dominating input S1 will set the output Q1 to "1".
When the function block is called for the first time, the initial state of Q1 is "0".
EN and ENO can be configured as additional parameters.

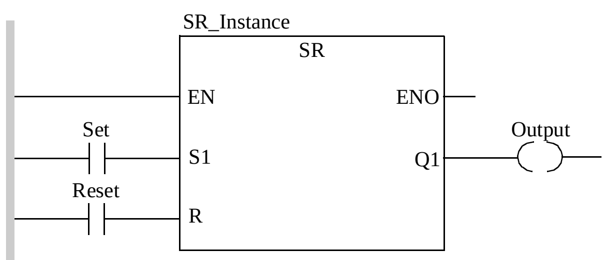
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL SR_Instance (S1:=Set, R:=Reset, Q1=>Output)
```

Representation in ST

Representation:

```
SR_Instance (S1:=Set, R:=Reset, Q1=>Output) ;
```

Parameter description

Description of the input parameters:

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

Description of the output parameter:

Parameter	Data type	Meaning
Q1	BOOL	Output

TRIGGER: Detection of all edges

59

Description

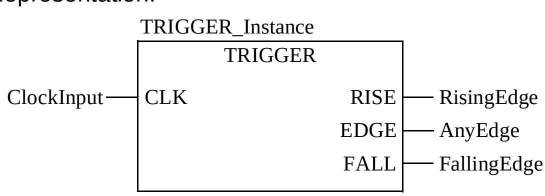
Function description

The function block recognizes all types of edges (1 -> 0 and 0 -> 1) at the `CLK` input. At a rising edge, a transition from "0" to "1"occurs on the `CLK` input; at a falling edge, a transition from "1" to "0" occurs on the `CLK` input.
At any edge, the `EDGE` output becomes "1".
At a rising edge, the `EDGE` output and the `RISE` output become "1".
At a falling edge, the `EDGE` output and the `FALL` output become "1". If no edge occurs, all outputs are "0".

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

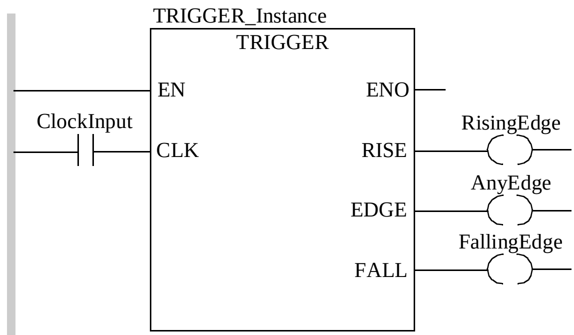
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL TRIGGER_Instance (CLK:=ClockInput, RISE=>RisingEdge,
    EDGE=>AnyEdge, FALL=>FallingEdge)
```

Representation in ST

Representation:

```
TRIGGER_Instance (CLK:=ClockInput, RISE=>RisingEdge,
    EDGE=>AnyEdge, FALL=>FallingEdge) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
CLK	BOOL	Clock input

Description of the output parameter:

Parameter	Data type	Meaning
RISE	BOOL	Indicator of a rising edge
EDGE	BOOL	Indicator of all types of edges
FALL	BOOL	Indicator of a falling edge

XOR: Exclusive OR function

60

Description

Function description

The function for a bit XOR link of the bit sequences at the inputs and returns the result at the output.
The data types of all input values and output values must be identical.
The number of inputs can be increased to a maximum of 32.
`EN` and `ENO` can be configured as additional parameters.

Further available functions

When using a Premium PLC, the following functions are also available in the Obsolete library:

- `XOR_DINT`
- `XOR_INT`

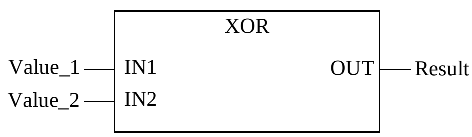
The functionality of these functions is identical to the function `XOR`.

Formula

`OUT` = `IN1` XOR `IN2` XOR .. XOR `INn`

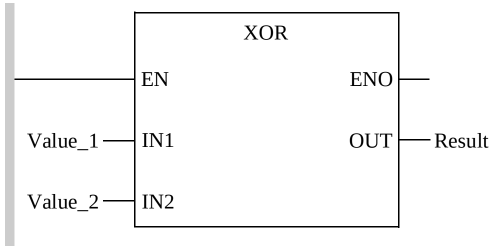
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Value_1
XOR Value_2
ST Result
```

Representation in ST

Representation:

```
Result := XOR (Value_1, Value_2) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
`Value_1`	`BOOL`, `BYTE`, `WORD`, `DWORD`	Input bit sequence
`Value_2`	`BOOL`, `BYTE`, `WORD`, `DWORD`	Input bit sequence
`Value_n`	`BOOL`, `BYTE`, `WORD`, `DWORD`	Input bit sequence n = max 32

Description of the output parameter:

Parameter	Data type	Meaning
`Result`	`BOOL`, `BYTE`, `WORD`, `DWORD`	Output bit sequence

Mathematics

VII

Introduction

Overview

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

What's in this Part?

This part contains the following chapters:

ABS: Absolute value computation

61

Description

Function description

The function computes the absolute value of the input value and assigns the result to the output.
The data types of the input and output values must be identical.

> **Note:** Because of IEC 61131-3 conformity, this function also works with the `UINT` and `UDINT` data types. This is not significant here.

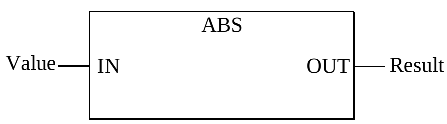
`EN` and `ENO` can be configured as additional parameters.

Formula

$$OUT = |IN|$$

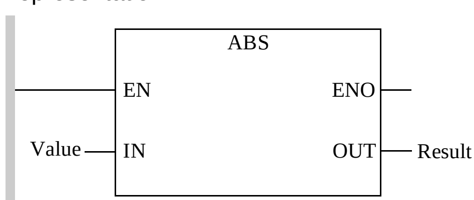
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Value
ABS
ST Result
```

Representation in ST

Representation:

```
Result := ABS (Value) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
`Value`	`INT`, `DINT`, `UINT`, `UDINT`, `REAL`	Input value

Description of the output parameter:

Parameter	Data type	Meaning
`Result`	`INT`, `DINT`, `UINT`, `UDINT`, `REAL`	Output value

Runtime error

The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1, if

- a value is below a limit value (data types `INT` and `DINT`)
  or
- an unauthorized floating point number is set at an input parameter of data type `REAL`. In this case, the status is also placed in %SW17 (See *Description of system words %SW12 to %SW18, p. 451*).

ACOS: Arc cosine

62

Description

Function description

The ACOS function calculates the principal arc cosine of a real value. The result is given in the form of an angle in radians.

The function call can also be carried out by ACOS_REAL.

The additional parameters EN and ENO can be configured.

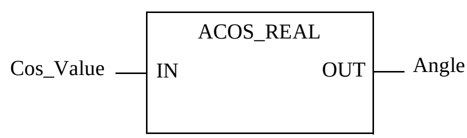
Formula

The formula is as follows:

$$\text{Angle} = \text{Arc}\cos(\text{Cos\_Value})$$

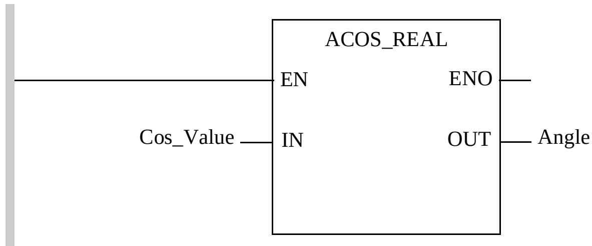
Representation in FBD

Representation:



Representation in LD

Illustration:



Representation in IL

Representation:

```
LD Cos_Value
ACOS_REAL
ST Angle
```

Representation in ST

Representation:

```
Angle:= ACOS_REAL(Cos_Value);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Cos_Value	REAL	Cosine of angle calculated at block output. $-1 \leq$ Cos_Value ≤ 1

The following table describes the output parameters:

Parameter	Type	Comment
Angle	REAL	Angle expressed in radians, whose cosine has the value Cos_Value. $0 \leq$ Angle $\leq \pi$

Runtime errors

When the absolute value Cos_Value is greater than 1, the system bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) changes to 1 and the system word **%SW17** (See *Description of system words %SW12 to %SW18, p. 451*) indicates the type of fault.

ADD: Addition

63

Description

Function description

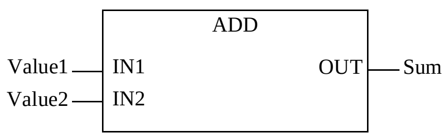
The function adds the input values and assigns the result to the output.
The data types of all input values and output values must be identical.
The number of inputs can be increased to a maximum of 32 for all functions.
For addition with values of the `TIME` data type, there is the block `ADD_TIME`
(See *ADD_TIME: Addition, p. 221*)
`EN` and `ENO` can be configured as additional parameters.

Formula

`INT, DINT, UINT, UDINT, REAL`:
`OUT = IN1 + IN2 + .. INn`

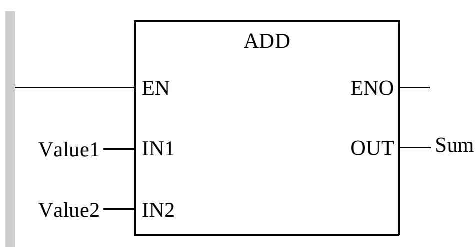
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Value1
ADD Value2
ST Sum
```

Representation in ST

Representation:

```
Sum := ADD (Value1, Value2) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
`Value1`	`INT`, `DINT`, `UINT`, `UDINT`, `REAL`	Summand
`Value2`	`INT`, `DINT`, `UINT`, `UDINT`, `REAL`	Summand
`Valuen`	`INT`, `DINT`, `UINT`, `UDINT`, `REAL`	Summand n = max 32

Description of the output parameter:

Parameter	Data type	Meaning
`Sum`	`INT`, `DINT`, `UINT`, `UDINT`, `REAL`	Sum

Runtime error

The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1, if

- the value range on the output is exceeded (all available data types)
  or
- an unauthorized floating point number is set at an input parameter of data type `REAL`. In this case, the status is also placed in %SW17 (See *Description of system words %SW12 to %SW18, p. 451*).

ADD_TIME: Addition

64

Description

Function description

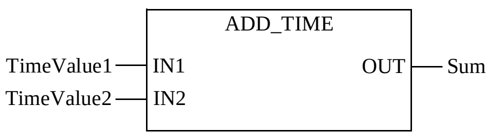
This function adds 2 input values of data type `TIME` and assigns the result to the output (also data type `TIME`).
`EN` and `ENO` can be configured as additional parameters.

Formula

`OUT = IN1 + IN2`

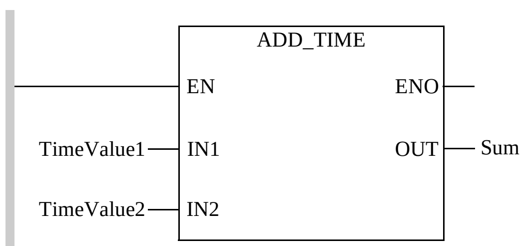
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD TimeValue1
ADD_TIME TimeValue2
ST Sum
```

Representation in ST

Representation:

```
Sum := ADD_TIME (TimeValue1, TimeValue2) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
TimeValue1	TIME	Summand
TimeValue2	TIME	Summand

Description of the output parameter:

Parameter	Data type	Meaning
Sum	TIME	Sum

Runtime error

System bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1 if the value range at the output is exceeded.

ASIN: Arc sine

65

Description

Function description

The `ASIN` function calculates the principal sine arc of a real value. The result is given in the form of an angle in radians.
The function call can also be carried out by `ASIN_REAL`.
The additional parameters `EN` and `ENO` can be configured.

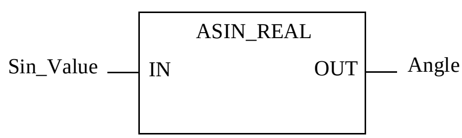
Formula

The formula is as follows:

$$\mathrm{Angle} = \mathrm{Arcsin}(\mathrm{Sin\_Value})$$

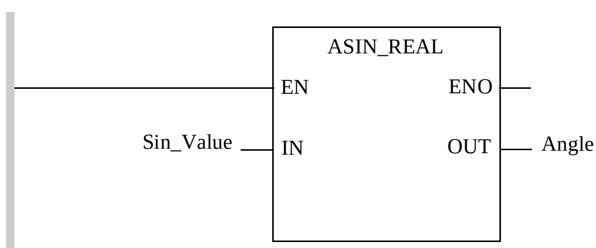
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Sin_Value
ASIN_REAL
ST Angle
```

Representation in ST

Representation:

```
Angle:= ASIN_REAL(Sin_Value);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Sin_Value	REAL	Sine of angle calculated at block output. $-1 \leq$ Sin_Value ≤ 1

The following table describes the output parameters:

Parameter	Type	Comment
Angle	REAL	Angle expressed in radians, whose sine has the value Sin_Value. $-\pi/2 \leq$ Angle $\leq +\pi/2$

Runtime errors

When the absolute value Sin_Value is greater than 1, the system bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) changes to 1 and the system word **%SW17** (See *Description of system words %SW12 to %SW18, p. 451*) indicates the type of fault.

ATAN: Arc tangent

66

Description

Function description

The `ACOS` function calculates the principal arc tangent of a real value. The result is given in the form of an angle in radians.
The function call can also be carried out by `ATAN_REAL`.
The additional parameters `EN` and `ENO` can be configured.

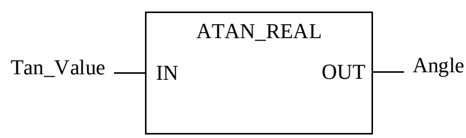
Formula

The formula is as follows:

$$Angle = Arctan(Tan\_Value)$$

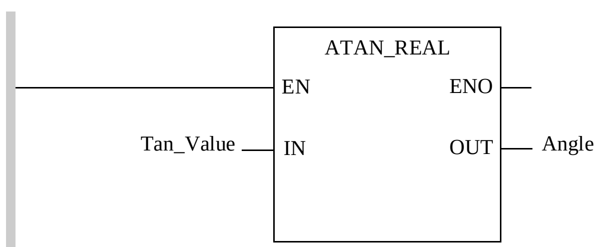
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Tan_Value
ATAN_REAL
ST Angle
```

Representation in ST

Representation:

```
Angle:= ATAN_REAL(Tan_Value);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`Tan_Value`	`REAL`	Tangent of angle calculated at block output. -1.#INF < `Tan_Value` < +1.#INF

The following table describes the output parameters:

Parameter	Type	Comment
`Angle`	`REAL`	Angle expressed in radians, whose tangent has the value `Tan_Value`. $-\pi/2 <$ `Angle` $< +\pi/2$

Runtime errors

When the absolute value `Tan_Value` is greater than 1, the system bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) changes to 1 and the system word **%SW17** (See *Description of system words %SW12 to %SW18, p. 451*) indicates the type of fault.

COS: Cosine

67

Description

Function description

The COS function calculates the cosine of an angle.
The function call can also be carried out by COS_REAL.
The additional parameters EN and ENO can be configured.

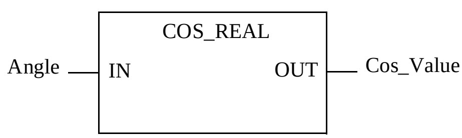
Formula

The formula is as follows:

$$\mathrm{Cos\_Value} = \mathrm{Cos(Angle)}$$

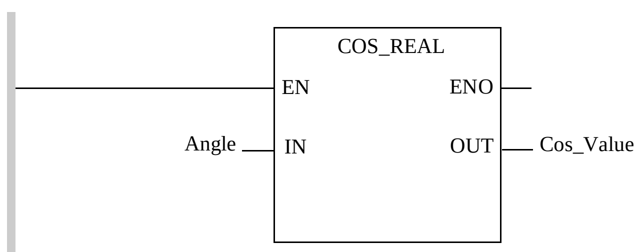
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Angle
COS_REAL
ST Cos_Value
```

Representation in ST

Representation:

```
Cos_Value:= COS_REAL(Angle);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Angle	REAL	Angle expressed in radians. $-2^{63} \leq$ Angle $\leq +2^{63}$

The following table describes the output parameters:

Parameter	Type	Comment
Cos_Value	REAL	Cosine of Angle expressed in radians. $-1 \leq$ Cos_Value ≤ 1

Runtime errors

When the absolute value of Angle is greater than 2^{63}, the system bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) changes to 1 and the system word **%SW17** (See *Description of system words %SW12 to %SW18, p. 451*) indicates the type of fault.

DEC: Decrementation of a variable

68

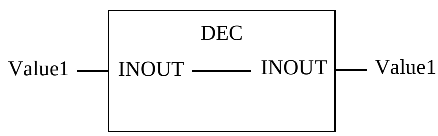
Description

Function description

The `DEC` function decrements a variable by 1.
The parameter of this function can be declared of type `ANY_INT`.
The additional parameters `EN` and `ENO` can be configured.

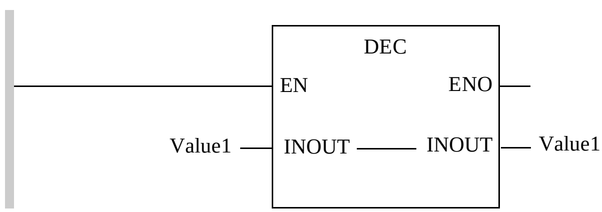
Representation in FBD

Representation applied to an integer:



Representation in LD

Representation applied to an integer:



Representation in IL

Representation applied to an integer:
`CAL DEC(Value1)`

Representation in ST

Representation applied to an integer:
`DEC(Value1);`

Description of parameters

The following table describes the input/output parameters:

Parameter	Type	Comment
`Value1`	`INT`, `DINT`, `UINT`, `UDINT`.	Each time the program uses this EF the variable `Value1` is decremented by one unit.

Runtime errors

In the case of overrun, the system bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) is set to 1 and the decremented value becomes positive (32767 for an integer for example).

DIV: Division

69

Description

Function description

The function divides the value at the `Dividend` with the value at the `Divisor` input and assigns the result to the output.
The data types of the input values and the output values must be identical.
For division with values of the `TIME` data type, you can use the block `DIVTIME` (See *DIVTIME: Division, p. 169*).
When dividing `INT`, `DINT`, `UINT` and `UDINT` data types, any decimal places in the result are omitted, e.g.

$7 \div 3 = 2$

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

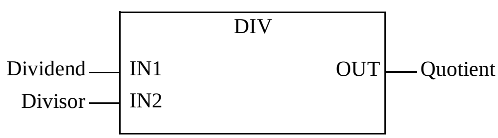
`EN` and `ENO` can be configured as additional parameters.

Formula

$$\mathrm{OUT} = ((\mathrm{IN1}) \div (\mathrm{IN2}))$$

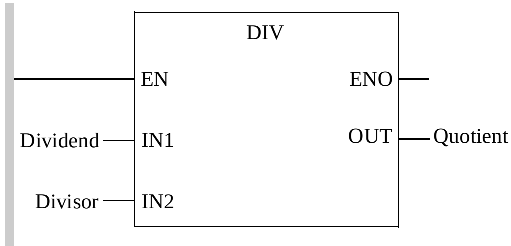
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Dividend
DIV Divisor
ST Quotient
```

Representation in ST

Representation:

```
Quotient := DIV (Dividend, Divisor) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
`Dividend`	`INT`, `DINT`, `UINT`, `UDINT`, `REAL`	Dividend
`Divisor`	`INT`, `DINT`, `UINT`, `UDINT`, `REAL`	Divisor

Description of the output parameter:

Parameter	Data type	Meaning
`Quotient`	`INT`, `DINT`, `UINT`, `UDINT`, `REAL`	Quotient

Runtime error

The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1, if

- an invalid division by 0 is executed (all available data types)
  or
- an unauthorized floating point number is set at an input parameter of data type `REAL`. In this case, the status is also placed in %SW17 (See *Description of system words %SW12 to %SW18, p. 451*).

DIVMOD: Division and Modulo

70

Description

Function description

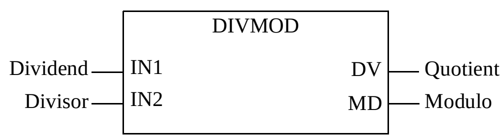
This procedure divides the value at the `Dividend` input by the value at the `Divisor` input. The result of the division is delivered at the `Quotient` output. The remainder of the division is delivered at the `Modulo` output.
If there is a decimal place in the division result, the division will truncate it.
The data types of all input and output values must be identical.
`EN` and `ENO` can be configured as additional parameters.

Formula

Block formula:
`DV` = `IN1` / `IN2`
`MD` = `IN1` mod `IN2`

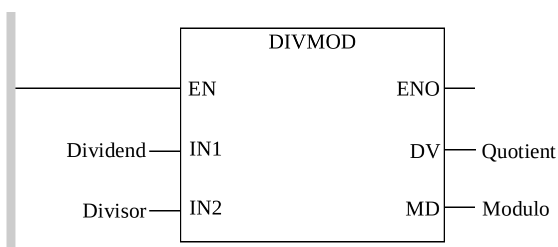
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Dividend
DIVMOD Divisor, Quotient, Modulo
```

Representation in ST

Representation:

```
DIVMOD (Dividend, Divisor, Quotient, Modulo);
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
`Dividend`	`INT`, `DINT`, `UINT`, `UDINT`	Dividend
`Divisor`	`INT`, `DINT`, `UINT`, `UDINT`	Divisor

Description of the output parameter:

Parameter	Data type	Meaning
`Quotient`	`INT`, `DINT`, `UINT`, `UDINT`	Quotient
`Modulo`	`INT`, `DINT`, `UINT`, `UDINT`	Modulo

Runtime error

The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1, if an invalid division by 0 is executed.

71

EXP: Natural exponential

Description

Function description

The EXP function calculates the natural exponential of a real.
The function call can also be carried out by EXP_REAL.
The additional parameters EN and ENO can be configured.

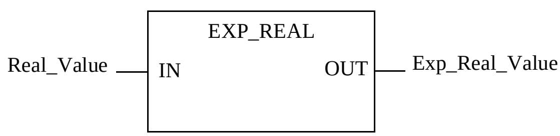
Formula

The formula is as follows:

$$\text{Exp\_Real\_Value} = \text{Exp}(\text{Real\_Value})$$

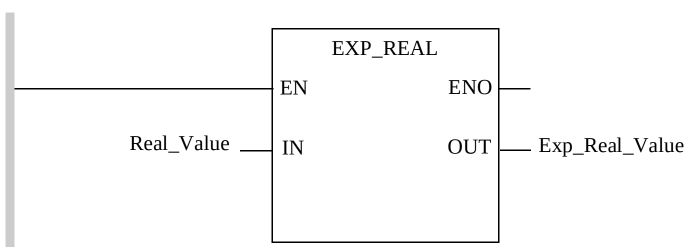
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Real_Value
EXP_REAL
ST Exp_Real_Value
```

Representation in ST

Representation:

```
Log_Real_Value:= EXP_REAL(Real_Value);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`Real_Value`	`REAL`	Real value of which we wish to obtain the Natural exponential -87.33654 < `Real_Value` < 88.72283

The following table describes the output parameters:

Parameter	Type	Comment
`Exp_Real_Value`	`REAL`	Natural exponential of Real_Value 0 < `Exp_Real_Value` < 1.#INF

Runtime errors

When `Real_Value` is situated outside the interval]-87.33654, 88.72283[, the system bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) changes to 1 and the system word **%SW17** (See *Description of system words %SW12 to %SW18, p. 451*) indicates the type of fault.

EXPT_REAL_***: Exponentiation of one value by another value

72

Description

Function Description

The `EXPT_REAL_***` function calculates the exponentiation of one value by another value.
The additional parameters `EN` and `ENO` can be configured.

Available functions

The available functions are as follows:
- `EXPT_REAL_INT`,
- `EXPT_REAL_DINT`,
- `EXPT_REAL_UINT`,
- `EXPT_REAL_UDINT`,
- `EXPT_REAL_REAL`.

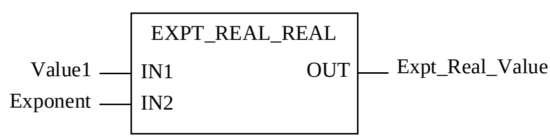
Formula

The formula is as follows:

$$\text{Expt\_Real\_Value} = \text{Value1}^{\text{Exponent}}$$

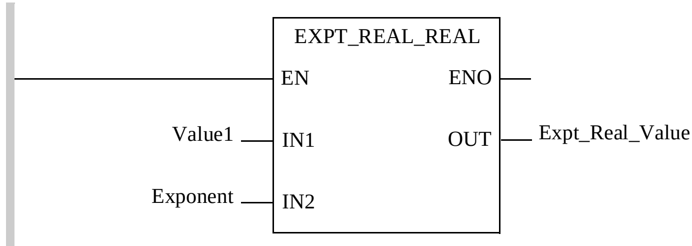
Representation in FBD

Representation applied to a real number:



Representation in LD

Representation applied to a real number:



Representation in IL

Representation applied to a real number:

```
LD Value1
EXPT_REAL_REAL Exponent
ST Expt_Real_Value
```

Representation in ST

Representation applied to a real number:

```
Expt_Real_Value := EXPT_REAL_REAL(Value1, Exponent);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Value1	REAL	Value for which you want to find the exponential by Exponent 0 ≤ Value1 < INF
Exponent	INT, UINT, DINT, UDINT, REAL.	Exponent of the exponential -INF < Exponent < +INF

The following table describes the output parameters:

Parameter	Type	Comment
Expt_Real__Value	REAL	Natural exponential of Value1 -1 < Expt_Real_Value < +INF

Runtime Errors

When Value1 is negative or when there is an Expt_Real_Value overrun the system bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) changes to 1 and the system word **%SW17** (See *Description of system words %SW12 to %SW18, p. 451*) indicates the type of fault.

INC: Incrementation of a variable

73

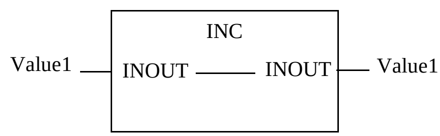
Description

Function description

The `INC` function increments a variable by 1.
The parameter of this function can be declared of type `ANY_INT`.
The additional parameters `EN` and `ENO` can be configured.

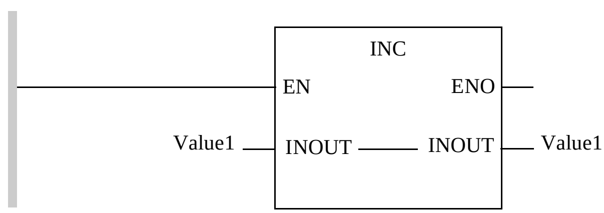
Representation in FBD

Representation applied to an integer:



Representation in LD

Representation applied to an integer:



Representation in IL

Representation applied to an integer:

```
CAL INC(Value1)
```

Representation in ST

Representation applied to an integer:

```
INC(Value1);
```

Description of parameters

The following table describes the input/output parameters:

Parameter	Type	Comment
`Value1`	`INT`, `DINT`, `UINT`, `UDINT`.	Each time the program uses this EF, the variable `Value1` is incremented by one unit.

Runtime errors

In the case of overrun, the system bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) is set to 1 and the incremented value becomes negative (-32768 for an integer for example).

LN: Natural logarithm

74

Description

Function description

The LN function calculates the natural logarithm of a real.
The function call can also be carried out by LN_REAL.
The additional parameters EN and ENO can be configured.

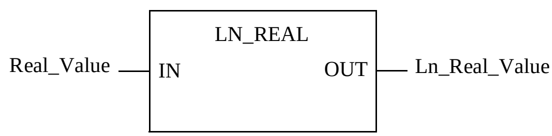
Formula

The formula is as follows:

$$Ln\_Real\_Value = Ln(Real\_Value)$$

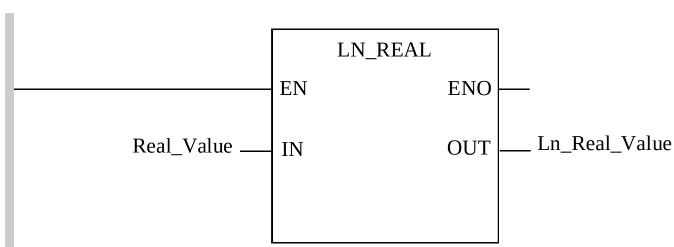
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Real_Value
LN_REAL
ST Ln_Real_Value
```

Representation in ST

Representation:

```
Ln_Real_Value := LN_REAL(Real_Value);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Real_Value	REAL	Real value of which we wish to obtain the natural logarithm. 0 < Real_Value < 1.#INF

The following table describes the output parameters:

Parameter	Type	Comment
Ln_Real_Value	REAL	Natural logarithm of Real_Value -1.#INF < Ln_Real_Value < +1.#INF

Runtime errors

When Real_Value is negative, the system bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) changes to 1 and the system word **%SW17** (See *Description of system words %SW12 to %SW18, p. 451*) indicates the type of fault.

LOG : Base 10 logarithm

75

Description

Function description

The `LOG` function calculates the base 10 logarithm of a real number.
The function call can also be carried out by `LOG_REAL`.
The additional parameters `EN` and `ENO` can be configured.

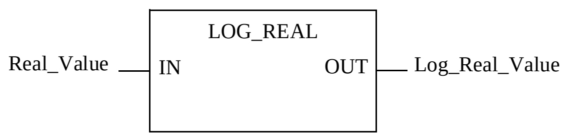
Formula

The formula is as follows:

$$\text{Log\_Real\_Value} = \text{Log}(\text{Real\_Value})$$

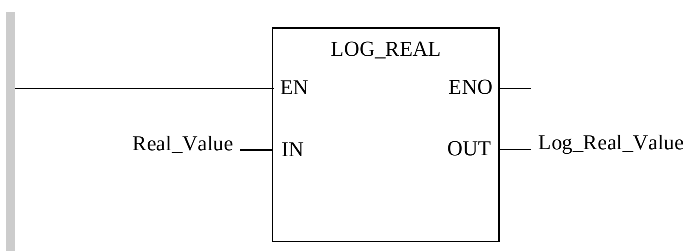
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Real_Value
LOG_REAL
ST Log_Real_Value
```

Representation in ST

Representation:

```
Log_Real_Value:= LOG_REAL(Real_Value);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`Real_Value`	`REAL`	Real value of which we wish to obtain the natural logarithm. 0 < `Real_Value` < 1.#INF

The following table describes the output parameters:

Parameter	Type	Comment
`Log_Real_Value`	`REAL`	Natural logarithm of Real_Value -1.#INF < `Log_Real_Value` < +1.#INF

Runtime errors

When`Real_Value` is negative, the system bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) changes to 1 and the system word **%SW17** (See *Description of system words %SW12 to %SW18, p. 451*) indicates the type of fault.

MOD: Modulo

76

Description

Function description

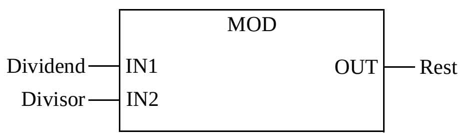
The function divides the value at the `Dividend` with the value at the `Divisor` input and assigns the modulo to the output.
The data types of all input values and output values must be identical.
`EN` and `ENO` can be configured as additional parameters.

Formula

`OUT` = `IN1` mod `IN2`

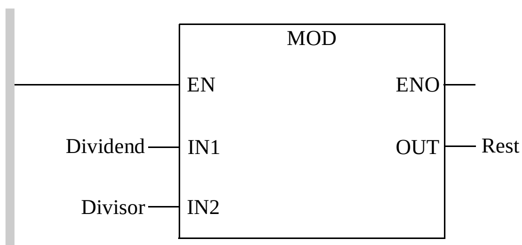
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Dividend
MOD Divisor
ST Rest
```

Representation in ST

Representation:

```
Rest := MOD (Dividend, Divisor) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
`Dividend`	`INT`, `DINT`, `UINT`, `UDINT`	Dividend
`Divisor`	`INT`, `DINT`, `UINT`, `UDINT`	Divisor

Description of the output parameter:

Parameter	Data type	Meaning
`Remainder`	`INT`, `DINT`, `UINT`, `UDINT`	Modulo

Runtime error

The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1, if an invalid division by 0 is executed.

MOVE: Assignment

77

Description

Function description

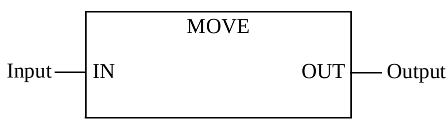
The function assigns the input value to the output.
This is a generic function, i.e. the data type to be processed will be determined by the variable that was first assigned to the function.
If a direct address of a variable is to be assigned or vice versa, always assign the variable to the function first. A direct address at input and output of the function is not authorized since this does not allow a clear definition of the data type.
The data types of the input and output values must be identical.
`EN` and `ENO` can be configured as additional parameters.

Formula

`OUT` = `IN`

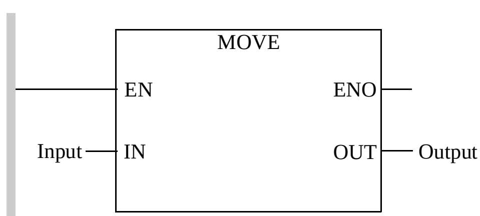
Representation in FBD

Representation:



Representation in LD

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



Representation in IL

Representation:

```
LD Input
MOVE
ST Output
```

Representation in ST

Representation:

```
Output := MOVE (Input) ;
```

Parameter description

Description of the input parameter:

Parameter	Data type	Meaning
Input	ANY	Input value

Description of the output parameter:

Parameter	Data type	Meaning
Output	ANY	Output value

MUL: Multiplication

78

Description

Function description

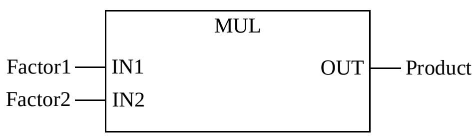
The function multiplies the input values and assigns the result to the output.
The data types of all input values and output values must be identical.
The number of inputs can be increased to a maximum of 32.
For multiplication with values of the `TIME` data type, you can use the block `MULTIME` (See *MULTIME: Multiplication, p. 171*).
`EN` and `ENO` can be configured as additional parameters.

Formula

`OUT` = `IN1` x `IN2` x .. x $\mathtt{IN}_n$

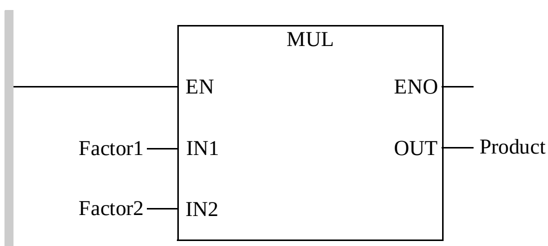
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Factor1
MUL Factor2
ST Product
```

Representation in ST

Representation:

```
Product := MUL (Factor1, Factor2) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
Factor1	INT, DINT, UINT, UDINT, REAL	Multiplicand (factor)
Factor2	INT, DINT, UINT, UDINT, REAL	Multiplier (factor)
Factorn	INT, DINT, UINT, UDINT, REAL	Multiplier (factor) n = max 32

Description of the output parameter:

Parameter	Data type	Meaning
Product	INT, DINT, UINT, UDINT, REAL	Product

Runtime error

The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1, if

- the value range at the output has been exceeded (all available data types) or
- an unauthorized floating point number is set at an input parameter of data type REAL. In this case, the status is also placed in %SW17 (See *Description of system words %SW12 to %SW18, p. 451*).

NEG: Negation

79

Description

Function description

The function negates the input value and delivers the result at the `NegatedOutput` output.
The negation causes a sign reversal, e.g.
 6 -> -6
-4 ->   4

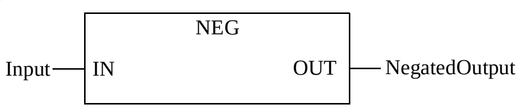
> **Note:** When the `INT` and `DINT` data types are processed, it is not possible to convert very long negative values into positive ones. However, the `ENO` output is not set to 0 when this error occurs.

> **Note:** When the `UINT` and `UDINT` data types are processed, an error message is always returned.

The data types of the input and output values must be identical.
`EN` and `ENO` can be configured as additional parameters.

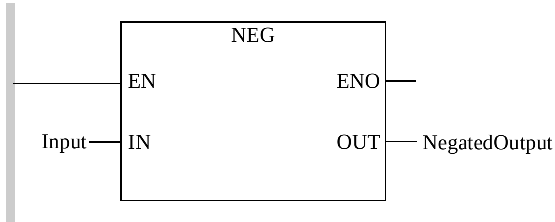
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Input
NEG
ST NegatedOutput
```

Representation in ST

Representation:

```
NegatedOutput := NEG (Input) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
`Input`	`INT`, `DINT`, `UINT`, `UDINT`, `REAL`	Input

Description of the output parameter:

Parameter	Data type	Meaning
`NegatedOutput`	`INT`, `DINT`, `UINT`, `UDINT`, `REAL`	Negated output

Runtime error

The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1, if

- a violation of the value range at the input occurs during the execution of the function (data types `INT` and `DINT`)
  or
- an input value of the data type UDINT or UINT is to be converted.

SIGN: Sign evaluation

80

Description

Function description

The function is used for the detection of negative signs.
With a value ≥ 0 at the input, the output becomes "0". With a value < 0 at the input, the output becomes "1".

Note: Because of IEC 61131-3 conformity, this function also works with the `UINT` and `UDINT`data types. This is not significant since these functions always return a 0 result.

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

Formula

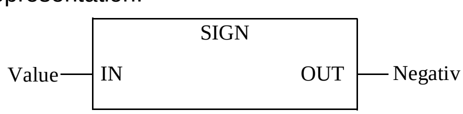
Block formula:
`OUT` = 1, if `IN` < 0
`OUT` = 0, if `IN` ≥ 0

Note: Because of the different processing of `REAL` and `INT` values, the following behavior results for signed 0 (+/-0):

- -0.0 -> `SIGN_REAL` -> 1
- +0.0 -> `SIGN_REAL` -> 0
- -0 -> `SIGN_INT`/`DINT` -> 0
- +0 -> `SIGN_INT`/`DINT` -> 0

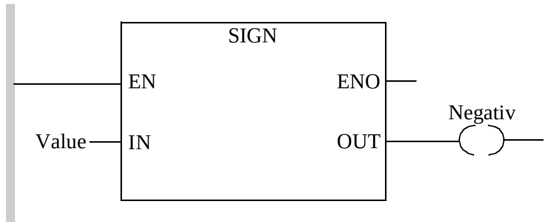
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Value
SIGN
ST Negativ
```

Representation in ST

Representation:

```
Negativ := SIGN (Value) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
`Value`	`INT`, `DINT`, `REAL`	Signed input

Description of output parameters:

Parameter	Data type	Meaning
`Negative`	`BOOL`	Sign evaluation

Runtime error

The system bit %S18 is set to 1 and `ENO` to 0 if

- an input value of the data type `UINT` or `UDINT` is to set.

SIN: Sine

81

Description

Function description

The `SIN` function calculates the sine of an angle.

The function call can also be carried out by `SIN_REAL`.

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

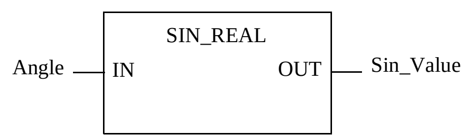
Formula

The formula is as follows:

$$\text{Sin\_Value} = \text{Sin}(\text{Angle})$$

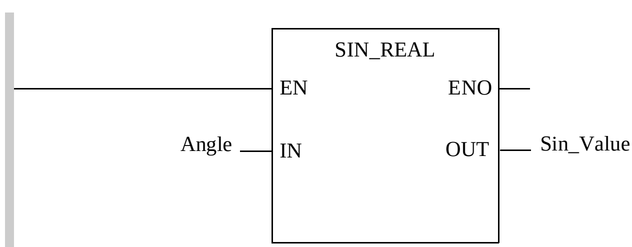
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Angle
SIN_REAL
ST Sin_Value
```

Representation in ST

Representation:

```
Sin_Value:= SIN_REAL(Angle);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`Angle`	`REAL`	Angle expressed in radians. $-2^{63} \leq$ `Angle` $\leq +2^{63}$

The following table describes the output parameters:

Parameter	Type	Comment
`Sin_Value`	`REAL`	Sine of `Angle`. $-1 \leq$ `Sin_Value` ≤ 1

Runtime errors

When the absolute value of `Angle` is greater than 2^{63}, the system bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) changes to 1 and the system word **%SW17** (See *Description of system words %SW12 to %SW18, p. 451*) indicates the type of fault.

SUB: Subtraction

82

Description

Function description

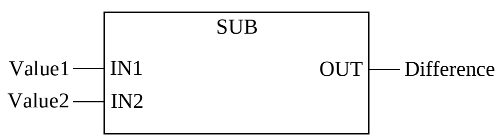
The function subtracts the value at the `Value2` input from the value at the `Value1` input and assigns the result to the output.
The data types of the input values and the output values must be identical.
For subtraction with values of the `TIME` data type, you can use the block `SUB_TIME` (See *SUB_TIME: Subtraction, p. 259*).
`EN` and `ENO` can be configured as additional parameters.

Formula

`Difference` = `Value1` - `Value2`

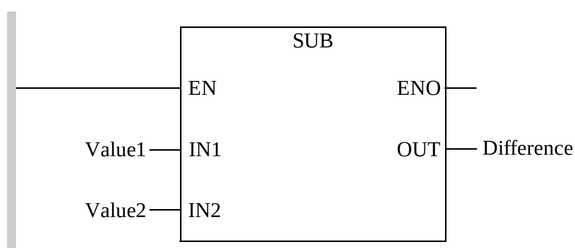
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Value1
SUB Value2
ST Difference
```

Representation in ST

Representation:

```
Difference := SUB (Value1, Value2) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
`Value1`	`INT`, `DINT`, `UINT`, `UDINT`, `REAL`	Minuend
`Value2`	`INT`, `DINT`, `UINT`, `UDINT`, `REAL`	Subtrahend

Description of the output parameter:

Parameter	Data type	Meaning
`Difference`	`INT`, `DINT`, `UINT`, `UDINT`, `REAL`	Difference

Runtime error

The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1, if

- the value range at the output has been exceeded (all available data types) or
- an unauthorized floating point number is set at an input parameter of data type `REAL`. In this case, the status is also placed in %SW17 (See *Description of system words %SW12 to %SW18, p. 451*).

SUB_TIME: Subtraction

83

Description

Function description

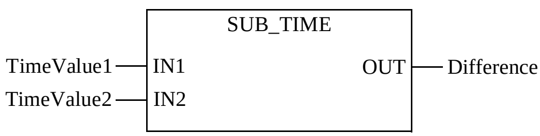
The function subtracts the value at the `TimeValue2` input from the value at the `TimeValue1` input and assigns the result to the output.
The data types of the input values and the output be `TIME`.
`EN` and `ENO` can be configured as additional parameters.

Formula

`Difference` = `TimeValue1` - `TimeValue2`

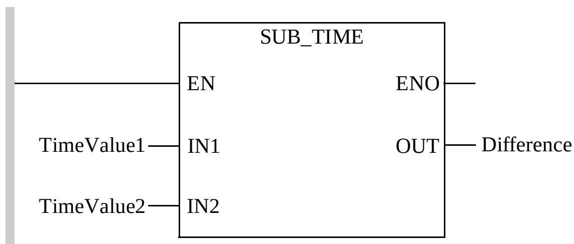
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD TimeValue1
SUB TimeValue2
ST Difference
```

Representation in ST

Representation:

```
Difference := SUB (TimeValue1, TimeValue2) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
TimeValue1	TIME	Minuend
TimeValue2	TIME	Subtrahend

Description of the output parameter:

Parameter	Data type	Meaning
Difference	TIME	Difference

Runtime error

The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1, if

- the value range at the output has been exceeded

SQRT_*** : Square root

84

Description

Function description

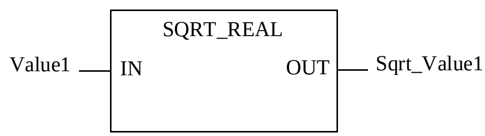
The `SQRT_***` function extracts the square root from a variable. This function can be called using its generic name or one of the function names described below.
The additional parameters `EN` and `ENO` can be configured.

Available functions

The available functions are as follows:
- `SQRT_INT`,
- `SQRT_DINT`,
- `SQRT_REAL`.

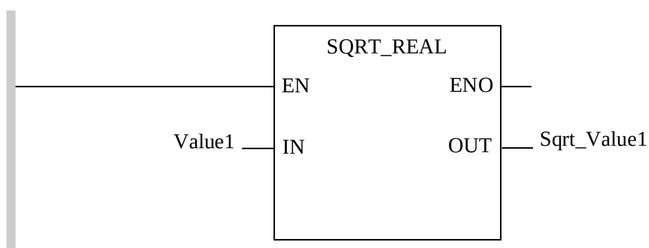
Representation in FBD

Representation applied to an integer:



Representation in LD

Representation applied to an integer:



Representation in IL

Representation applied to an integer:

```
LD Value1
SQRT_REAL
ST Sqrt_Value1
```

Representation in ST

Representation applied to an integer:

```
Sqrt_Value1 := SQRT_REAL(Value1);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Value1	INT, DINT, REAL.	Variable whose square root you want to extract. 0 ≤ Value1

The following table describes the output parameters:

Parameter	Type	Comment
Sqrt_Value1	INT, DINT, REAL.	Sqrt_Value1 contains the square root of Value1. Sqrt_Value1 is of the same type as Value1. When the type is INT, Sqrt_Value1 is rounded down to the lower value, for Value1 = 15, Sqrt_Value1 = 3.

Runtime errors

When Value1 is of REAL type and negative, the result of the function contains -1.#NAN and bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) = 1.
When Value1 is of INT or DINT type and negative, the result of the function contains the negative value Value1 and bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) = 1.

85

TAN: Tangent

Description

Function description

The `TAN` function calculates the tangent of an angle.
The function call can also be carried out by `TAN_REAL`.
The additional parameters `EN` and `ENO` can be configured.

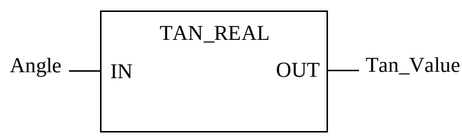
Formula

The formula is as follows:

$$Tan\_Value = Tan(Angle)$$

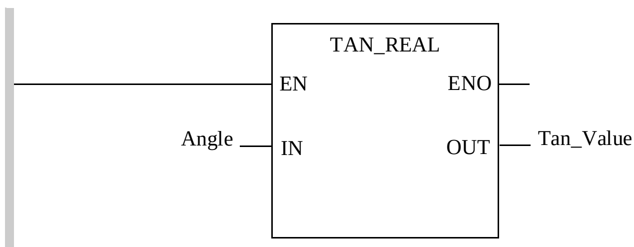
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Angle
TAN_REAL
ST Tan_Value
```

Representation in ST

Representation:

```
Tan_Value:= TAN_REAL(Angle);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Angle	REAL	Angle expressed in radians. $-2^{63} \leq$ Angle $\leq +2^{63}$

The following table describes the output parameters:

Parameter	Type	Comment
Tan_Value	REAL	Tangent of Angle -1.#INF < Tan_Value < +1.#INF

Runtime errors

When the absolute value of Angle is greater than 2^{63}, the system bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) changes to 1 and the system word **%SW17** (See *Description of system words %SW12 to %SW18, p. 451*) indicates the type of fault.

Statistical

VIII

Introduction

Overview

This section describes the elementary functions and elementary function blocks of the `Statistical` group.

What's in this Part?

This part contains the following chapters:

AVE: Averaging

86

Description

Function description

The procedure calculates the average of weighted input values and gives the result at the output.
Two successive inputs (K_Xn) represent one pair of values. The first K_Xn input corresponds to K1, the next to X1, the one after that to K2, etc.
The number of K_Xn inputs can be increased to 32 by vertically modifying the size of the block frame. This corresponds to a maximum of 16 value pairs.
The number of inputs must be even.
The data types of all input and output values must be identical.
EN and ENO can be configured as additional parameters.

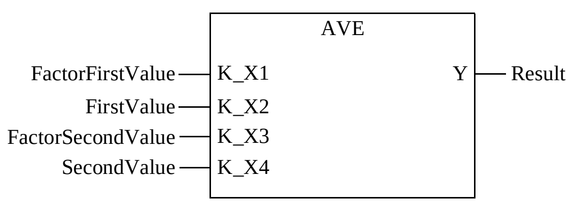
Formula

Block formula:

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

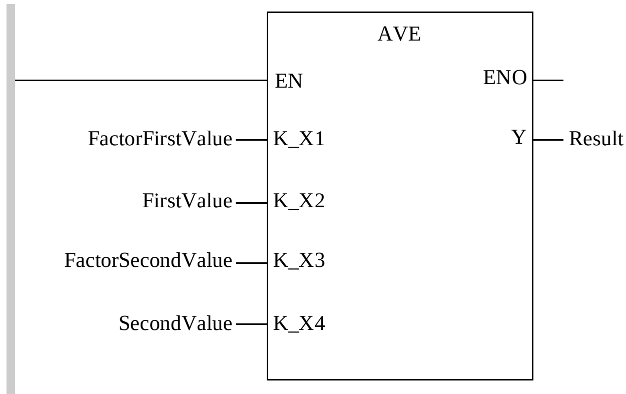
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD FactorFirstValue
AVE FirstValue, FactorSecondValue, SecondValue
ST Result
```

Representation in ST

Representation:

```
Result := AVE (FactorFirstValue, FirstValue,
          FactorSecondValue, SecondValue) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
FactorFirstValue	INT, DINT, UINT, UDINT, REAL	Factor (K1) for first value
FirstValue	INT, DINT, UINT, UDINT, REAL	First value (X1)
FactorSecondValue	INT, DINT, UINT, UDINT, REAL	Factor (K2) for second value
FactorSecondValue	INT, DINT, UINT, UDINT, REAL	Second value (X2)
:		
FactorHalfnValue	INT, DINT, UINT, UDINT, REAL	Factor for value $\frac{n}{2}$ of (K_X(n-1))
HalfnValue	INT, DINT, UINT, UDINT, REAL	Value $\frac{n}{2}$ of (K_X(n) n = max 32

Description of output parameters:

Parameter	Data type	Meaning
`Result`	`INT`, `DINT`, `UINT`, `UDINT`, `REAL`	Average value

Runtime error

The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1, if

- the value range of the output is exceeded (all available data types)
  or
- an invalid division by 0 is executed (all available data types)
  or
- an unauthorized floating point number is set at an input parameter of data type `REAL`. In this case, the status is also placed in %SW17 (See *Description of system words %SW12 to %SW18, p. 451*).

Note: For a list of all block error codes and values, see *Statistical, p. 443*.

LIMIT: Limit

87

Description

Function description

This function transfers the unchanged input value (`Input`) to the output if the input value is not less than the minimum value (`LowerLimit`) and does not exceed the maximum value (`UpperLimit`). If the input value (`Input`) is less than the minimum value (`LowerLimit`), the minimum value will be transferred to the output. If the input value (`Input`) exceeds the maximum value (`UpperLimit`), the maximum value will be transferred to the output.
The data types of all input values and output values must be identical.
`EN` and `ENO` can be configured as additional parameters.

Formula

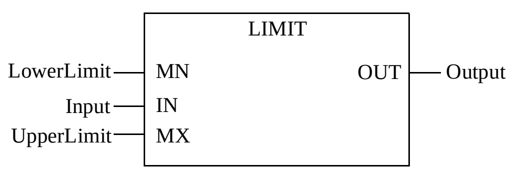
`OUT` = `IN`, if (`IN ≥ MN) & (IN ≤ MX`)

`OUT` = `MN`, if (`IN < MN`)

`OUT` = `MX`, if (`IN > MX`)

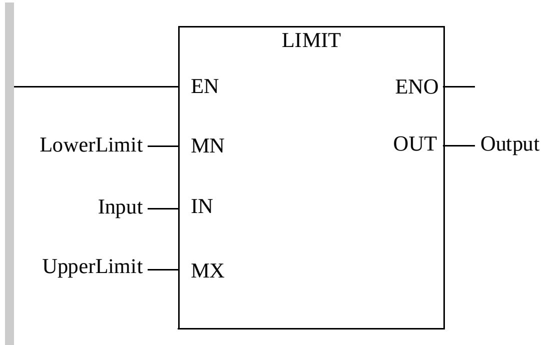
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD UpperLimit
LIMIT Input, LowerLimit
ST Output
```

Representation in ST

Representation:

```
Output := LIMIT (UpperLimit, Input, LowerLimit) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
LowerLimit	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME	lower limit
Input	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME	Input
UpperLimit	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME	upper limit

Description of the output parameter:

Parameter	Data type	Meaning
Output	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME	Output

Runtime error

If there is an unauthorized floating point number at the input, an error message is returned.

LIMIT_IND: Limit with indicator

88

Description

Function description

This procedure transfers the unchanged input value (`Input`) to the (`Output`), if the input value is not less than the minimum value (`LimitMinimum`) and does not exceed the maximum value (`LimitMaximum`). If the input value (`Input`) is less than the minimum value (`LimitMinimum`), the minimum value will be transferred to the output. If the input value (`Input`) exceeds the maximum value (`LimitMaximum`), the maximum value will be transferred to the output.
Additionally, a indication is given if the minimum or maximum value is violated. If the value at the (`Input`) input is less than the value at the (`LimitMinimum`) input, the (`MinimumViolation`) output becomes "1". If the value at the (`Input`) input is more than the value at the (`LimitMaximum`) input, the (`MaximumViolation`) output becomes "1".
The data types of the (`LimitMinimum`, `Input`, `LimitMaximum`) input values and the (`Output`) output value must be identical.
`EN` and `ENO` can be configured as additional parameters.

Formula

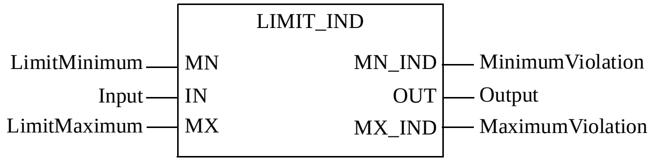
Block formula:
`OUT` = `IN`, if (`IN` ≤ `MX`) & `IN` ≥ `MN`
`OUT` = `MN`, if (`IN` < `MN`)
`OUT` = `MX`, if (`IN` > `MX`)

`MN_IND` = 0, if `IN` ≥ `MN`
`MN_IND` = 1, if `IN` < `MN`

`MX_IND` = 0, if `IN` ≤ `MX`
`MX_IND` = 1, if `IN` > `MX`

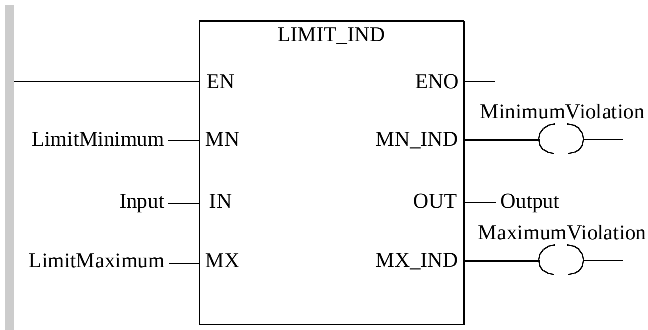
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD LimitMinimum
LIMIT_IND Input, LimitMaximum, MinimumViolation,
          Output, MaximumViolation
```

Representation in ST

Representation:

```
LIMIT_IND (LimitMinimum, Input, LimitMaximum,
           MinimumViolation, Output, MaximumViolation);
```

Parameter description

Description of the input parameters:

Parameter	**Data type**	**Meaning**
LimitMinimum	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME	Limit of minimum value
Input	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME	Input
LimitMaximum	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME	Limit of maximum value

Description of the output parameter:

Parameter	**Data type**	**Meaning**
MinimumViolation	BOOL	Display of minimum value violation
Output	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME	Output
MaximumViolation	BOOL	Display of maximum value violation

MAX: Maximum value function

89

Description

Function description

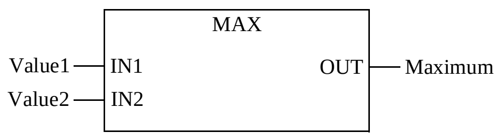
The function assigns the largest input value to the output.
The data types of all input values and output values must be identical.
The number of inputs can be increased.
`EN` and `ENO` can be configured as additional parameters.

Formula

`OUT = MAX {IN1, IN2, .., INn}`

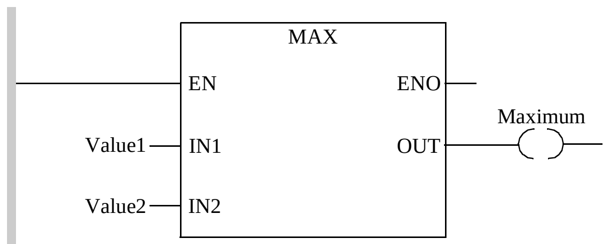
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Value1
MAX Value2
ST Maximum
```

Representation in ST

Representation:

```
Maximum := MAX (Value1, Value2) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
Value1	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME	1. Input value
Value2	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME	2. Input value
Valuen	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME	n. Input value n = max 32

Description of output parameters:

Parameter	Data type	Meaning
Maximum	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME	Maximum value

Runtime error

If an unauthorized floating point number is created for an input parameter of data type REAL, the system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1 and the status is placed in %SW17 (See *Description of system words %SW12 to %SW18, p. 451*).

Note: For a list of all the block error messages and values, see *Common Floating Point Errors, p. 444*.

MIN: Minimum value function

90

Description

Function description

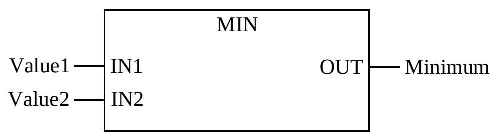
The function assigns the smallest input value to the output.
The data types of all input values and output values must be identical.
The number of inputs can be increased.
`EN` and `ENO` can be configured as additional parameters.

Formula

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

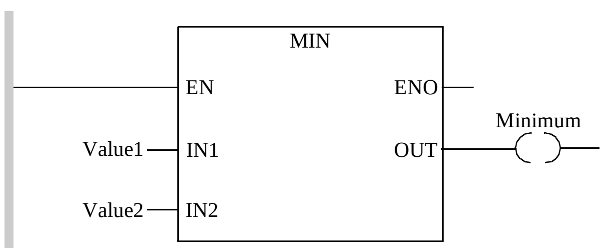
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Value1
MIN Value2
ST Minimum
```

Representation in ST

Representation:

```
Minimum := MIN (Value1, Value2) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
Value1	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME	1. Input value
Value2	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME	2. Input value
Valuen	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME	n. Input value n = max 32

Description of output parameters:

Parameter	Data type	Meaning
Minimum	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME	Minimum value

Runtime error

If an unauthorized floating point number is created for an input parameter of data type REAL, the system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1 and the status is placed in %SW17 (See *Description of system words %SW12 to %SW18, p. 451*).

Note: For a list of all the block error messages and values, see *Common Floating Point Errors, p. 444*.

MUX: Multiplexer

91

Description

Function description

This function transfers the respective input to the output depending on the value at the `K` input.
The number of inputs can be increased.
`EN` and `ENO` can be configured as additional parameters.

Example

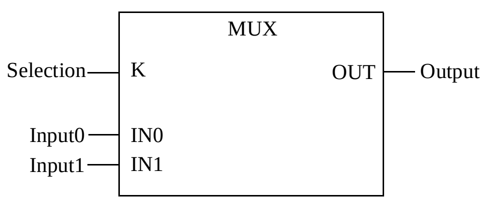
`K` = 0: Input `IN0` is transferred to the output
`K` = 1: Input `IN1` is transferred to the output
`K` = 5: Input `IN5` is transferred to the output
`K`= n: Input `INn` is transferred to the output

Data types

The data types at the inputs `Input0` to `Inputn` and at the output must be identical.

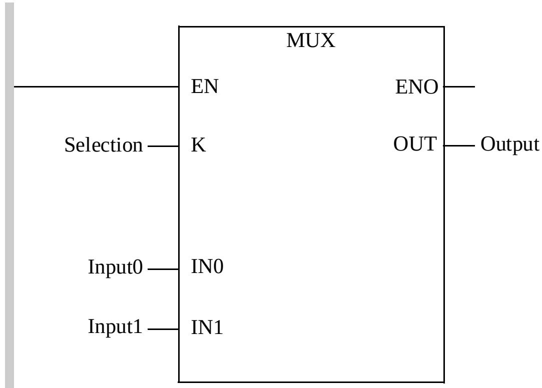
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Selection
MUX Input0, Input1
ST Output
```

Representation in ST

Representation:

```
Output := MUX (Selection, Input0, Input1) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
K	INT, DINT, UINT, UDINT	Selection input K = 0...30
IN0	ANY	1. Input
IN1	ANY	2. Input
IN2	ANY	3. Input
INn	ANY	n+1. input, n = max. 30

Description of output parameters:

Parameter	Data type	Meaning
OUT	ANY	Output

Runtime error

An error message is returned if the value range of the Kinput (selector) is exceeded.

Note: For a list of all block error codes and values, see *Statistical, p. 443*.

SEL: Binary selection

92

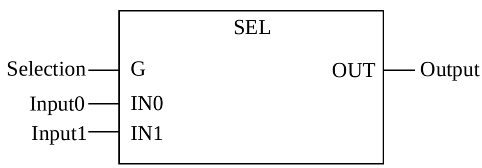
Description

Function description

The function is used for binary selection between two input values.
Depending on the state of the `Selection` input, either the `Input0` input or `Input1` input is transferred to the `Output` output.
`Selection` = 0 -> `Output` = `Input0`
`Selection` = 1 -> `Output` = `Input1`
The data types of the `Input0` and `Input1` input values and the `Output` output values must be identical.
`EN` and `ENO` can be configured as additional parameters.

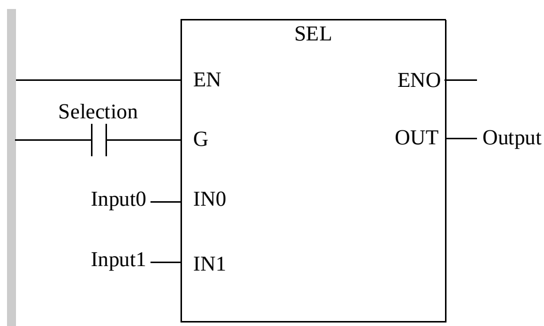
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Selection
SEL Input0, Input1
ST Output
```

Representation in ST

Representation:

```
Output := SEL (Selection, Input0, Input1) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
`Selection`	`BOOL`	Selection input
`Input0`	`ANY`	Input 0
`Input1`	`ANY`	Input 1

Description of the output parameter:

Parameter	Data type	Meaning
`Output`	`ANY`	Output

Strings

IX

Introduction

Overview

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

What's in this Part?

This part contains the following chapters:

CONCAT_STR: Concatenation of two character strings

93

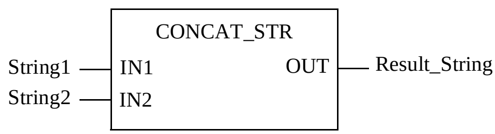
Description

Function description

The CONCAT_STR function concatenates two character strings.
The additional parameters EN and ENO can be configured.

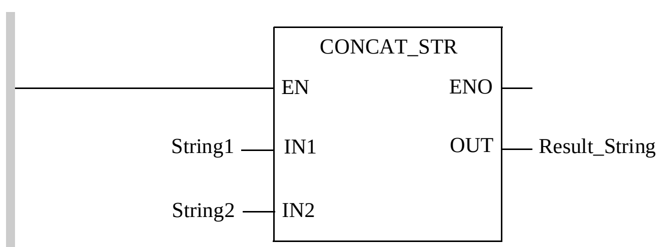
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD String1
CONCAT_STR String2
ST Result_String
```

Representation in ST

Representation:

```
Result_String:= CONCAT_STR(String1, String2);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`String1`	`STRING`	First character string to concatenate. **Example**: `String1` contains "SWITCH TO"
`String2`	`STRING`	Second character string to concatenate. **Example**: `String2` contains "RUN"

The following table describes the output parameters:

Parameter	Type	Comment
`Result_String`	`STRING`	Resulting string is equal to the content of the two strings `String1` and `String2`. **Example**: for the values in the example provided in the previous table, `Result_String` contains ‘SWITCH TO RUN’

Runtime errors

If the string `Result_String` is too short to contain the result, the system bit **%S15** (See *Description of system bits %S9 to %S13, p. 447*) changes to 1 and the result is truncated. Otherwise, the string `Result_String` is completed by the characters NUL (16#00).

DELETE_INT: Deletion of a sub-string of characters

94

Description

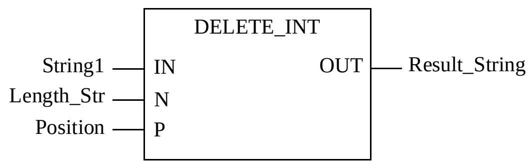
Function description

The DELETE_INT function removes a certain number of characters starting from a certain rank. The result is a character string.

The additional parameters EN and ENO can be configured.

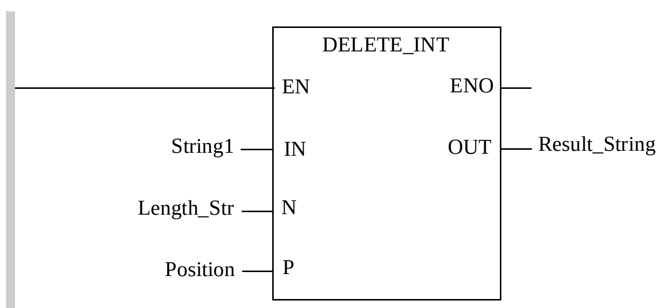
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD String1
DELETE_INT Length_Str, Position
ST Result_String
```

Representation in ST

Representation:

```
Result_String:= DELETE_INT(String1, Length_Str, Position);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
String1	STRING	Original character string from which we wish to delete certain elements. **Example**: String1 contains "SWITCH TO STOP"
Length_Str	INT	Length of string to be deleted. **Example**: Length_Str =10
Position	INT	Rank of first character of the string to be deleted. **Example**: Position =1

The following table describes the output parameters:

Parameter	Type	Comment
Result_String	STRING	Resulting string equal to content of String1 from which have been removed Length_Str characters starting from the rank Position. **Example**: for the values in the example provided in the previous table, Result_String contains 'STOP' (10 characters are deleted starting from position 1).

EQUAL_STR: Comparison of two character strings

95

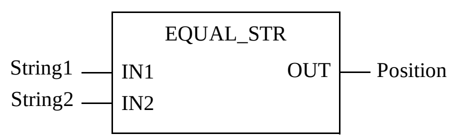
Description

Function description

The `EQUAL_STR` function compares two character strings.
The additional parameters `EN` and `ENO` can be configured.

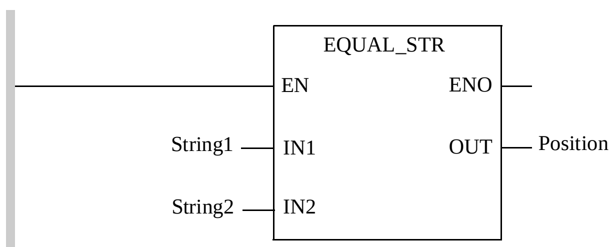
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:
```
LD String1
EQUAL_STR String2
ST Position
```

Representation in ST

Representation:
```
Position:= EQUAL_STR(String1, String2);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`String1`	`STRING`	First character string to compare. **Example**: `String1` contains "SWITCH TO STOP"
`String2`	`STRING`	Second character string to compare. **Example**: `String2` contains "SWITCH TO RUN"

The following table describes the output parameters:

Parameter	Type	Comment
`Position`	`INT`	Position of first character that differs between the two strings `String1` and `String2`. When the two strings are identical, `Position` = -1. **Example**: with the values indicated in the example in the previous table, `Position` = 11 **Note**: upper case characters are treated as different to lower case characters.

FIND_INT: Finding a sub-string of characters

96

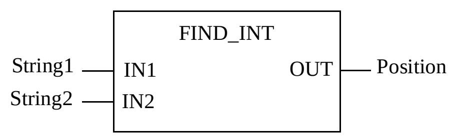
Description

Function description

The `FIND_INT` function searches for the occurrence of a character string in another string.
The additional parameters `EN` and `ENO` can be configured.

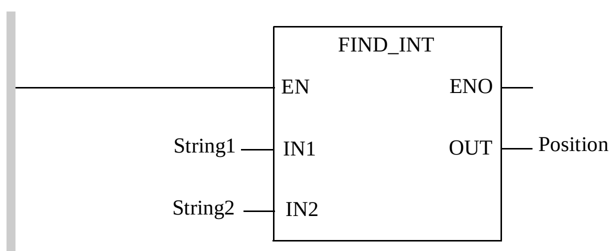
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD String1
FIND_INT String2
ST Position
```

Representation in ST

Representation:

```
Position:= FIND_INT(String1, String2);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`String1`	`STRING`	Character string in which the search is carried out. **Example**: `String1` contains "SWITCH TO STOP"
`String2`	`STRING`	Character string containing the text to find **Example**: `String2` contains ‘STOP’

The following table describes the output parameters:

Parameter	Type	Comment
`Position`	`INT`	If `String2` is contained in `String1`, `Position` contains the rank of the first character of `String2` found in `String1`. When `String2` is not contained in `String1`, `Position` = -1. **Example**: with the values indicated in the example in the previous table, `Position` = 11

INSERT_INT: Insertion of a sub-string of characters

97

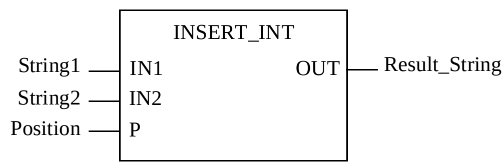
Description

Function description

The `INSERT_INT` function inserts a character string into another character string starting from a given rank. The result is a character string.
The additional parameters `EN` and `ENO` can be configured.

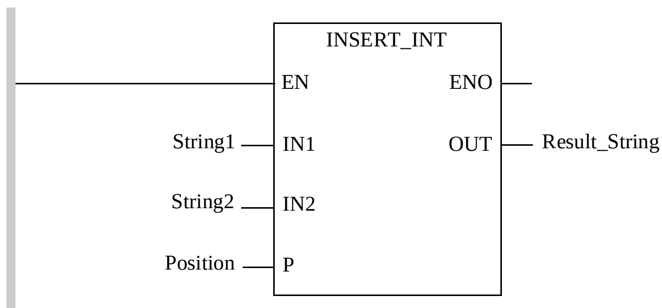
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD String1
INSERT_INT String2, Position
ST Result_String
```

Representation in ST

Representation:

```
Result_String:= DELETE_INT(String1, Length_Str, Position);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`String1`	`STRING`	Original character string to which another character string is added starting from a certain position. **Example**: `String1` contains 'START CYCLE'
`String2`	`STRING`	Character string to be inserted in `String1`. **Example**: `String2` contains ' AUTO'
`Position`	`INT`	Rank of character after which `String2` is inserted. **Example**: `Position` =5

The following table describes the output parameters:

Parameter	Type	Comment
`Result_String`	`STRING`	The string `String2` has been inserted in the string `String1` after the position `Position` to form `Result_String`. **Example**: for the values in the example provided in the previous table, `Result_String` contains 'START AUTO CYCLE'. **Note**: it is impossible to make an insertion at the start of a string with this function (use the `CONCAT_STR (See` *Function description, p. 291*) function).

Runtime errors

The bit **%S15** (See *Description of system bits %S9 to %S13, p. 447*) is set to 1 in the following cases:

- `Position` ≤ 0, `Result_String` then contains the end of string characters (16#00).
- The maximum size of the string `Result_String` is too small to insert `String2`. `Result_String` is truncated.

LEFT_INT: Extraction of characters to the left

98

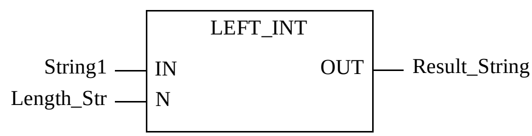
Description

Function description

The `LEFT_INT` function extracts a certain number of characters situated to the leftmost of a string. The result is a character string.
The additional parameters `EN` and `ENO` can be configured.

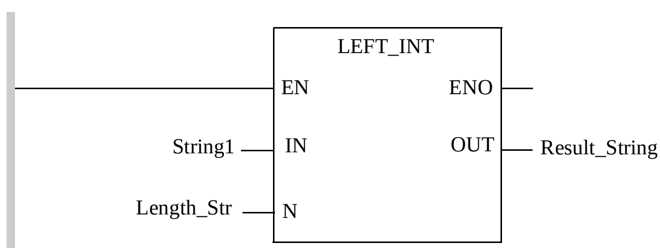
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD String1
LEFT_INT Length_Str
ST Result_String
```

Representation in ST

Representation:

```
Result_String:= LEFT_INT(String1, Length_Str);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
String1	STRING	String of characters from which we wish to extract the Length_Str leftmost characters. **Example**: String1 contains "SWITCH TO STOP"
Length_Str	INT	Number of characters to be extracted. **Example**: Length_Str =10.

The following table describes the output parameters:

Parameter	Type	Comment
Result_String	STRING	String containing the Length_Str leftmost characters of String1. **Example**: for the values in the example provided in the previous table, Result_String contains ‘SWITCH TO’ (9 leftmost characters of String1).

Runtime errors

The bit **%S15** (See *Description of system bits %S9 to %S13, p. 447*) is set to 1 in the following cases:

- Length_Str ≤ 0, Result_String then contains the end of string characters (16#00).
- The maximum size of the string Result_String is less than Length_Str, Result_String is truncated.

LEN_INT: Length of character string

99

Description

Function description

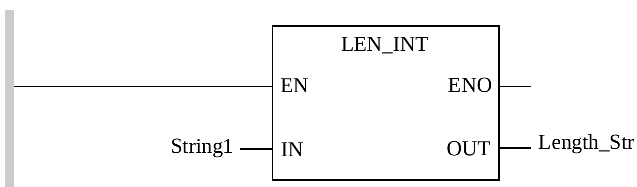
The `LEN_INT` function calculates the number of characters of a character string. The additional parameters `EN` and `ENO` can be configured.

Representation in FBD

Representation:

Representation in LD

Representation:



Representation in IL

Representation:

```
LD String1
LEN_INT
ST Length_Str
```

Representation in ST

Representation:

```
Length_Str:= LEN_INT(String1);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
String1	STRING	String of characters for which the length is to be determined. **Example**: String1 contains "SWITCH TO STOP"

The following table describes the output parameters:

Parameter	Type	Comment
Length_Str	INT	Length_Str contains the length of the character string String1. **Example**: with the values indicated in the example in the previous table, Length_Str = 14

MID_INT: Extraction of a sub-string of characters

100

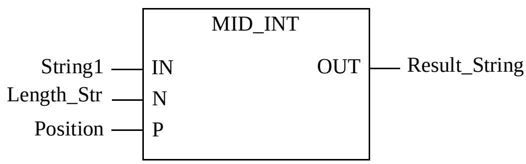
Description

Function description

The `MID_INT` function extracts a sub-string of characters starting from a certain rank. The result is a character string.
The additional parameters `EN` and `ENO` can be configured.

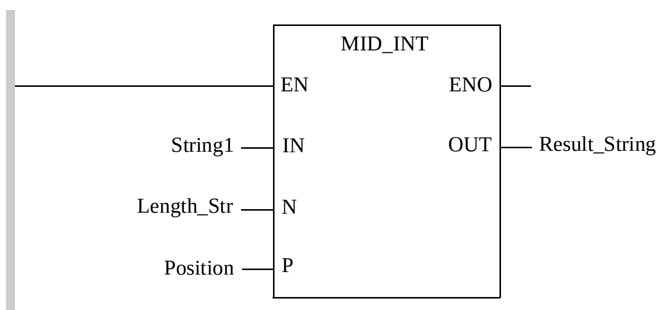
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD String1
MID_INT Length_Str, Position
ST Result_String
```

Representation in ST

Representation:

```
Result_String:= MID_INT(String1, Length_Str, Position);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
String1	STRING	Original string containing the sub-string to be extracted. **Example**: String1 contains "SWITCH TO STOP"
Length_Str	INT	Length of the sub-string to be extracted. **Example**: Length_Str =4
Position	INT	Rank of first character of the sub-string to be extracted. **Example**: Position =11

The following table describes the output parameters:

Parameter	Type	Comment
Result_String	STRING	Sub-string of String1 starting from rank Position over a length of Length_Str. **Example**: for the values in the example provided in the previous table, Result_String contains ‘STOP’.

REPLACE_INT: Replacement of a sub-string of characters

101

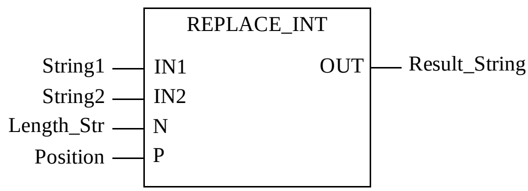
Description

Function description

The `REPLACE_INT` function replaces a character string in another character string starting from a certain rank and for a certain length. The result is a character string. The additional parameters `EN` and `ENO` can be configured.

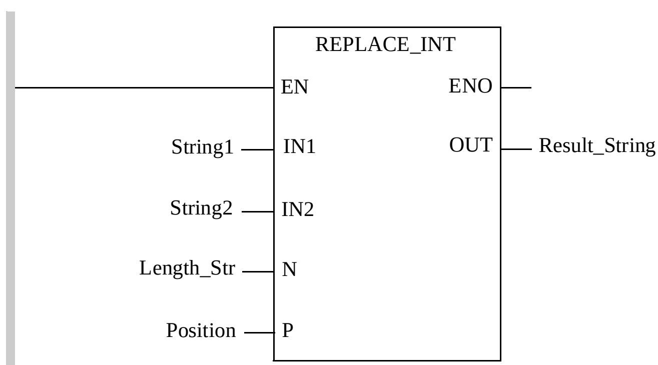
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD String1
REPLACE_INT String2, Length_Str, Position
ST Result_String
```

Representation in ST

Representation:

```
Result_String:= REPLACE_INT(String1, String2, Length_Str,
Position);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
String1	STRING	Original string of characters into which is inserted a sub-string of characters starting from Position over a length of Length_Str. **Example**: String1 contains "SWITCH TO RUN" **Note**: the length Length_Str is the length of the text to be replaced and not the length of the string String2. As it happens, the replacement string can be of a different length to the string that is replaced.
String2	STRING	Character string to be inserted in String1 to replace the existing characters. **Example**: String2 contains ‘STOP’
Length_Str	INT	Number of characters to be replaced in String1 by String2 **Example**: Length_Str =3
Position	INT	Rank of first character of the sub-string to be replaced **Example**: Position =11

The following table describes the output parameters:

Parameter	Type	Comment
Result_String	STRING	The string string2 has replaced the Length_Str characters starting from the rank Position in the string String1 to form Result_String. **Example**: for the values in the example provided in the previous table, Result_String contains ‘SWITCH TO STOP’.

Runtime errors

The bit **%S15** (See *Description of system bits %S9 to %S13, p. 447*) is set to 1 in the following cases:

- `Position` ≤ 0, `Result_String` then contains the end of string characters (16#00).
- The maximum size of the string `Result_String` is too small to insert `String2`. `Result_String` is truncated.
- `Position` is greater than or equal to the length of `String1`. `Result_String` is composed of the characters NUL (16#00).

RIGHT_INT: Extraction of a character string to the right

102

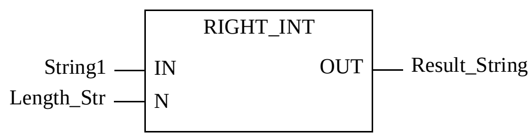
Description

Function description

The `RIGHT_INT` function extracts a certain number of characters situated to the rightmost of a string. The result is a character string.
The additional parameters `EN` and `ENO` can be configured.

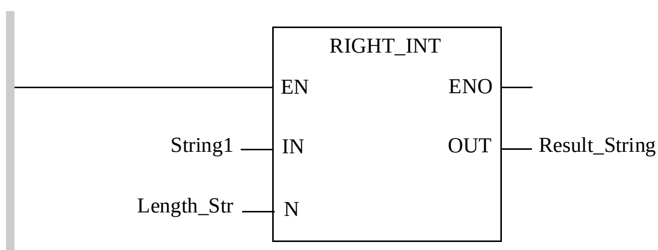
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD String1
RIGHT_INT Length_Str
ST Result_String
```

Representation in ST

Representation:
```
Result_String:= RIGHT_INT(String1, Length_Str);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
String1	STRING	String of characters from which we wish to extract the Length_Str rightmost characters. **Example**: String1 contains "SWITCH TO STOP"
Length_Str	INT	Number of characters to be extracted. **Example**: Length_Str =4

The following table describes the output parameters:

Parameter	Type	Comment
Result_String	STRING	String containing the Length_Str rightmost characters of String1. **Example**: for the values in the example provided in the previous table, Result_String contains 'STOP' (4 rightmost characters of String1).

Runtime errors

The bit **%S15** (See *Description of system bits %S9 to %S13, p. 447*) is set to 1 in the following cases:

- Length_Str ≤ 0, Result_String then contains the end of string characters (16#00).
- The maximum size of the string Result_String is less than Length_Str. Result_String is truncated.

Timer & Counter

X

Introduction

Overview

This section describes the elementary functions and elementary function blocks of the `Timer & Counter` family.

What's in this Part?

This part contains the following chapters:

CTD, CTD_***: Down counter

103

Description

Function description

The function blocks are used for downwards counting.
A "1" signal at the LD input causes the value of the PV input to be allocated to the CV output. With each transition from "0" to "1" at the CD input, the value of CV is reduced by 1.
When CV ≤ is 0, the Q output becomes "1".

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

EN and ENO can be configured as additional parameters.

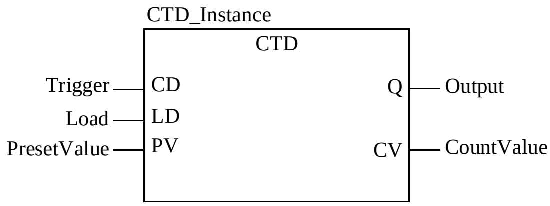
Available functions

There are two different specifications of the function block:

- CTD
  This function block specification is defined in IEC 61131-3 and only works with the INT data type.
- CTD_***
  This function block specification is an expansion that conforms to IEC 61131-3 to cover other data types. The following blocks are available:
  - CTD_INT
  - CTD_DINT
  - CTD_UINT
  - CTD_UDINT

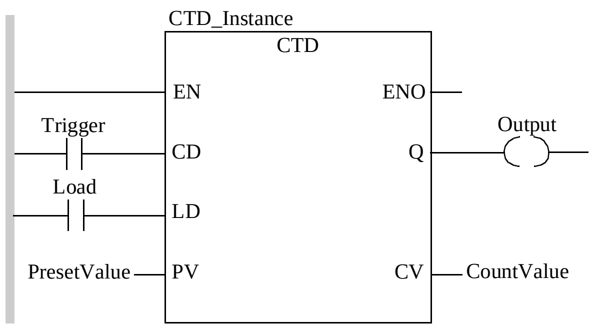
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL CTD_Instance (CD:=Trigger, LD:=Load,
    PV:=PresetValue, Q=>Output, CV=>CountValue)
```

Representation in ST

Representation:

```
CTD_Instance (CD:=Trigger, LD:=Load, PV:=PresetValue,
    Q=>Output, CV=>CountValue) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
CD	BOOL	Trigger input
LD	BOOL	Load data
PV	When CTD: INT When CTD_***: INT, DINT, UINT, UDINT	Preset value

Description of the output parameter:

Parameter	Data type	Meaning
Q	BOOL	Output
CV	When CTD: INT When CTD_***: INT, DINT, UINT, UDINT	Count value (actual value)

CTU, CTU_***: Up counter

104

Description

Function description

The function blocks are used for upwards counting.
A "1" signal at the `R` input causes the value "0" to be assigned to the `CV` output. With each transition from "0" to "1" at the `CU` input, the value of `CV` is incremented by 1. When `CV` ≥ `PV`, the `Q` output is set to "1".

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

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

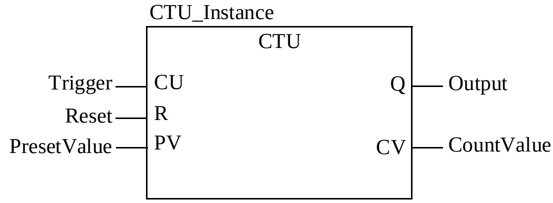
Available functions

There are two different specifications of the function block:

- `CTU`
  This function block specification is defined in IEC 61131-3 and only works with the `INT` data type.
- `CTU_***`
  This function block specification is an expansion that conforms to IEC 61131-3 to cover other data types. The following blocks are available
  - `CTU_INT`
  - `CTU_DINT`
  - `CTU_UINT`
  - `CTU_UDINT`

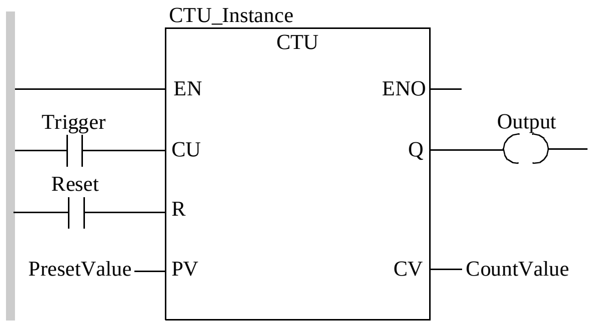
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL CTU_Instance (CU:=Trigger, R:=Reset,
    PV:=PresetValue, Q=>Output, CV=>CountValue)
```

Representation in ST

Representation:

```
CTU_Instance (CU:=Trigger, R:=Reset, PV:=PresetValue,
    Q=>Output, CV=>CountValue) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
CU	BOOL	Trigger input
R	BOOL	Reset
PV	When CTU: INT When CTU_***: INT, DINT, UINT, UDINT	Preset value

Description of the output parameter:

Parameter	Data type	Meaning
Q	BOOL	Output
CV	When CTU: INT When CTU_***: INT, DINT, UINT, UDINT	Count value (actual value)

CTUD, CTUD_***: Up/Down counter

105

Description

Function description

The function blocks are used for upwards and downwards counting.
A "1" signal at the R input causes the value "0" to be assigned to the CV output. A "1" signal at the LD input causes the value of the PV input to be allocated to the CV output. With each transition from "0" to "1" at the CU input, the value of CV is incremented by 1. With each transition from "0" to "1" at the CD input, the value of CV is reduced by 1.
If there is a simultaneous "1" signal at inputs R and LD, input R has precedence.
When CV ≥ PV, output QU is "1".
Bei CV ≤ is 0, the QD output becomes "1".

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

EN and ENO can be configured as additional parameters.

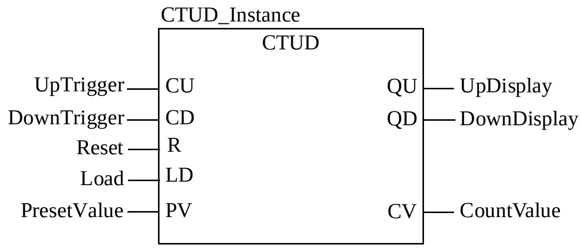
Available functions

There are two different specifications of the function block:

- CTUD
  This function block specification is defined in IEC 61131-3 and only works with the INT data type.
- CTUD_***
  This function block specification is an expansion that conforms to IEC 61131-3 to cover other data types. The following blocks are available
  - CTUD_INT
  - CTUD_DINT
  - CTUD_UINT
  - CTUD_UDINT

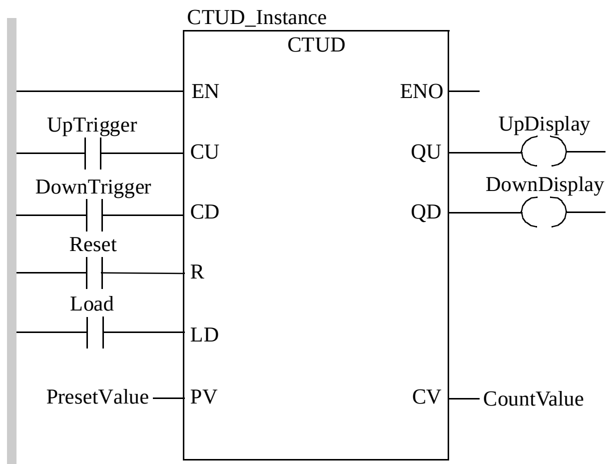
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL CTUD_Instance (CU:=UpTrigger, CD:=DownTrigger,
    R:=Reset, LD:=Load, PV:=PresetValue, QU=>UpDisplay,
    QD=>DownDisplay, CV=>CountValue)
```

Representation in ST

Representation:

```
CTUD_Instance (CU:=UpTrigger, CD:=DownTrigger,
    R:=Reset, LD:=Load, PV:=PresetValue, QU=>UpDisplay,
    QD=>DownDisplay, CV=>CountValue) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
CU	BOOL	Up counter trigger input
CD	BOOL	Down counter trigger input
R	BOOL	Reset
LD	BOOL	Load data
PV	When CTUD: INT, When CTUD_***: INT, DINT, UINT, UDINT	Preset value

Description of the output parameter:

Parameter	Data type	Meaning
QU	BOOL	Up display
QD	BOOL	Down display
CV	When CTUD: INT When CTUD_***: INT, DINT, UINT, UDINT	Count value (actual value)

TOF: Off delay

106

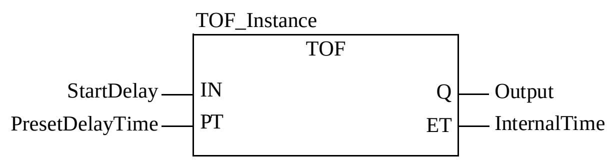
Description

Function description

The function block is used as the Off delay.
When the function block is called for the first time, the initial state of `ET` is "0".
`EN` and `ENO` can be configured as additional parameters.

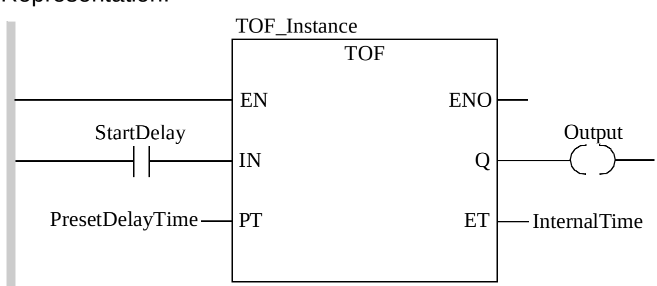
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL TOF_Instance (IN:=StartDelay, PT:=PresetDelayTime,
    Q=>Output, ET=>InternalTime)
```

Representation in ST

Representation:

```
TOF_Instance (IN:=StartDelay, PT:=PresetDelayTime,
    Q=>Output, ET=>InternalTime) ;
```

Parameter description

Description of the input parameters:

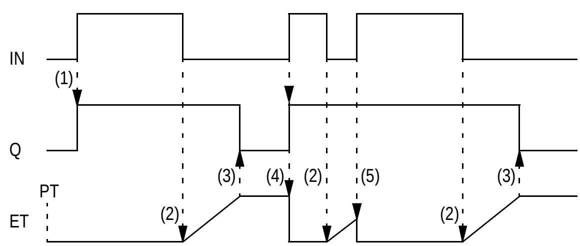
Parameter	Data type	Meaning
IN	BOOL	Start delay
PT	TIME	Preset delay time

Description of the output parameter:

Parameter	Data type	Meaning
Q	BOOL	Output
ET	TIME	Internal time

Timing diagram

Representation of the OFF delay TOF:



(1) If IN becomes "1", Q becomes "1".

(2) If IN becomes "0", the internal time (ET) is started.

(3) If the internal time reaches the value of PT, Q becomes "0".

(4) If IN becomes "1", Q becomes "1", and the internal time is stopped/reset.

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

TON: On delay

107

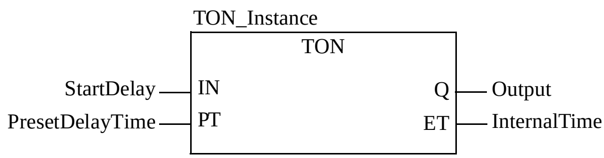
Description

Function description

The function block is used as the On delay.
When the function block is called for the first time, the initial state of `ET` is "0".
`EN` and `ENO` can be configured as additional parameters.

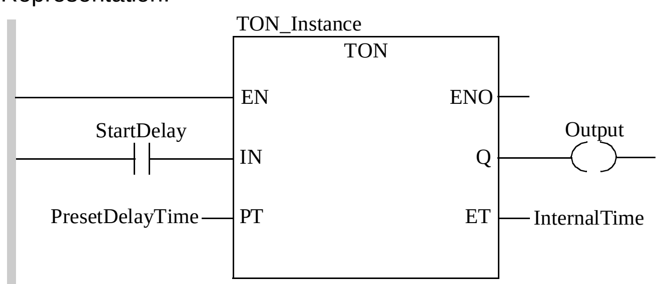
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL TON_Instance (IN:=StartDelay, PT:=PresetDelayTime,
    Q=>Output, ET=>InternalTime)
```

Representation in ST

Representation:

```
TON_Instance (IN:=StartDelay, PT:=PresetDelayTime,
    Q=>Output, ET=>InternalTime) ;
```

Parameter description

Description of the input parameters:

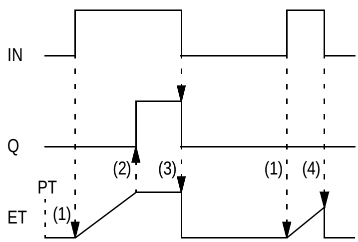
Parameter	Data type	Meaning
IN	BOOL	Start delay
PT	TIME	Preset delay time

Description of the output parameter:

Parameter	Data type	Meaning
Q	BOOL	Output
ET	TIME	Internal time

Timing diagram

Representation of the ON delay `TON`:



(1) If IN becomes "1", the internal time (ET) starts.

(2) If the internal time reaches the value of PT, Q becomes "1".

(3) If IN becomes "0", Q becomes "0" and the internal time is stopped/reset.

(4) If IN becomes "0" before the internal time has reached the value of PT, the internal time stops/resets without Q going to "1".

TP: Pulse

108

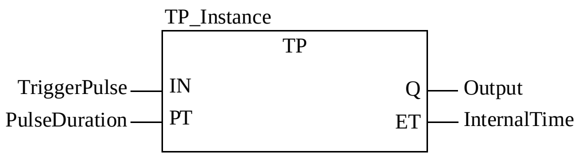
Description

Function description

The function block is used for the generation of a pulse with defined duration. When the function block is called for the first time, the initial state of `ET` is "0". `EN` and `ENO` can be configured as additional parameters.

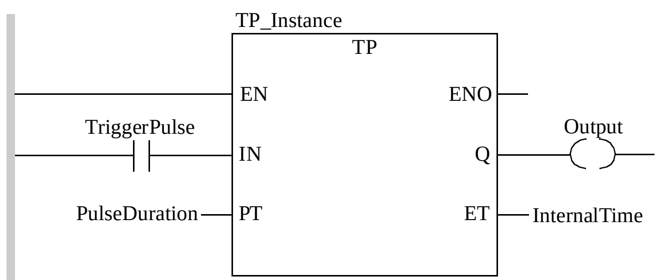
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
CAL TP_Instance (IN:=TriggerPulse, PT:=PulseDuration,
    Q=>Output, ET=>InternalTime)
```

Representation in ST

Representation:

```
TP_Instance (IN:=TriggerPulse, PT:=PulseDuration,
    Q=>Output, ET=>InternalTime) ;
```

Parameter description

Description of the input parameters:

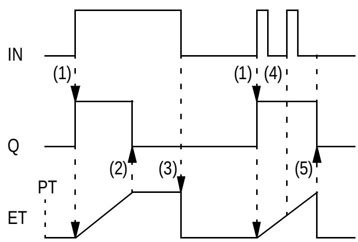
Parameter	Data type	Meaning
IN	BOOL	Trigger pulse
PT	TIME	Preset pulse duration

Description of the output parameter:

Parameter	Data type	Meaning
Q	BOOL	Output
ET	TIME	Internal time

Timing diagram

Representation of the TP pulse:



(1) If IN becomes "1", Q becomes "1" and the internal time (ET) starts.

(2) If the internal time reaches the value of PT, Q becomes "0" (independent of IN).

(3) The internal time stops/is reset if IN becomes "0".

(4) If the internal time has not reached the value of PT yet, the internal time is not affected by a clock at IN.

(5) If the internal time has reached the value of PT and IN is "0", the internal time stops/is reset and Q becomes "0".

Type to type

XI

Introduction

Overview

This section describes the elementary functions and elementary function blocks of the `Type to type` family.

What's in this Part?

This part contains the following chapters:

BCD_TO_INT: Conversion of a BCD integer into pure binary

109

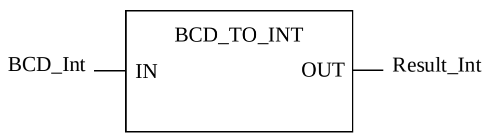
Description

Function description

The `BCD_TO_INT` function converts an integer in Binary Coded Decimal (BCD) format into a binary coded integer.
The additional parameters `EN` and `ENO` can be configured.

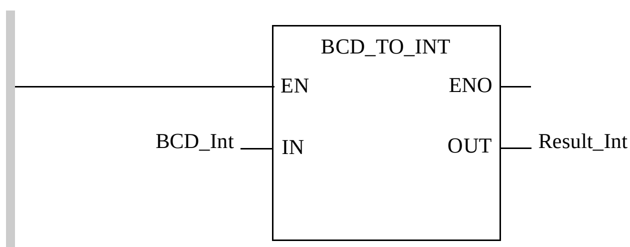
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD BCD_Int
BCD_TO_INT
ST Result_Int
```

Representation in ST

Representation:

```
Result_Int:= BCD_TO_INT(BCD_Int);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`BCD_Int`	`INT`	Integer in BCD format. **Example**: `BCD_Int` = 16#99

The following table describes the output parameters:

Parameter	Type	Comment
`Result_Int`	`INT`	`Result_Int` is a binary coded integer. **Example**: with the value provided in the example in the previous table, `Result_Int` = 99

Runtime errors

The bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) is set to 1 when the value to be converted is not a value coded in BCD. The result of the function then returns the value of the input parameter.

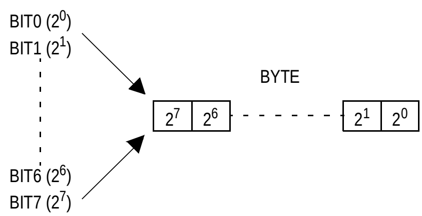
BIT_TO_BYTE: Type conversion

110

Description

Function description

The function converts 8 input values of the data type `BOOL` to an output of the `BYTE` data type.
The input values are assigned to the individual bits of the byte at the output according to the input names.



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

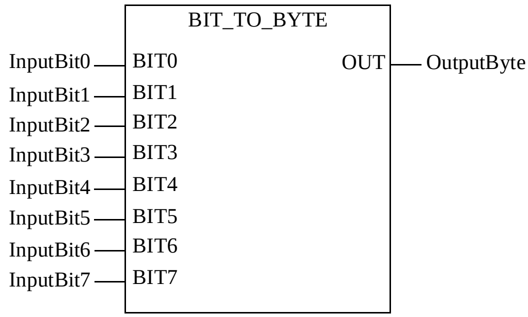
Formula

Block formula:

$$OUT = \{BIT7, BIT6, \ldots, BIT0\}$$

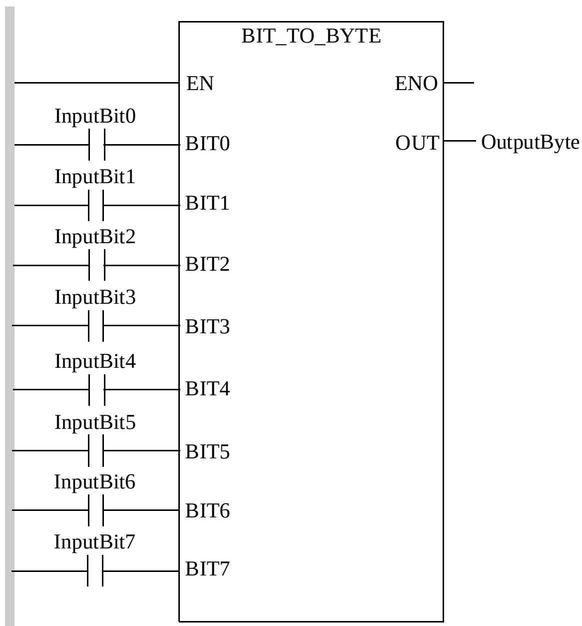
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD InputBit0
BIT_TO_BYTE InputBit1, InputBit2, InputBit3, InputBit4,
            InputBit5, InputBit6, InputBit7
ST OutputByte
```

Representation in ST

Representation:

```
OutputByte := BIT_TO_BYTE (InputBit0, InputBit1,
    InputBit2, InputBit3, InputBit4, InputBit5, InputBit6,
    InputBit7) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
InputBit0	BOOL	Input bit 0
InputBit1	BOOL	Input bit 1
:	:	:
InputBit7	BOOL	Input bit 7

Description of the output parameter:

Parameter	Data type	Meaning
OutputByte	BYTE	Output value

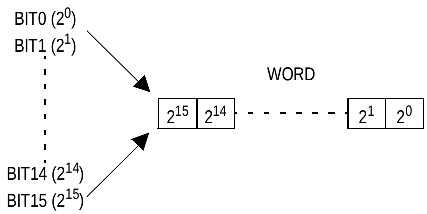
BIT_TO_WORD: Type conversion

111

Description

Function description

The function converts 16 input values of the `BOOL` data type to an output value of the `WORD` data type.
The input values are assigned to the individual bits of the word at the output according to the input names.



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

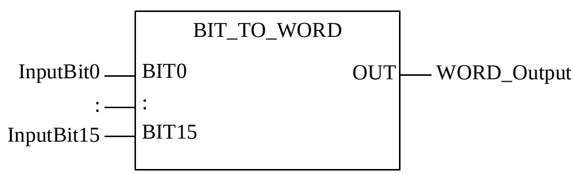
Formula

Block formula:

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

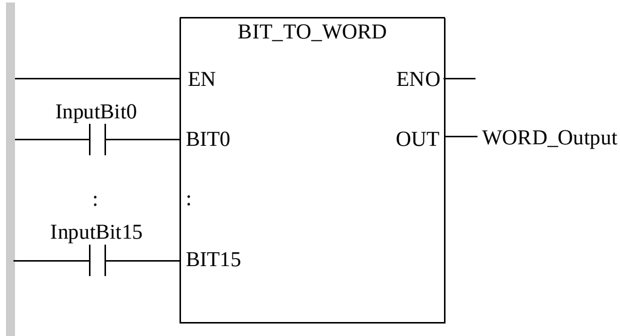
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD InputBit0
BIT_TO_WORD InputBit1, InputBit2, InputBit3, InputBit4,
            InputBit5, InputBit6, InputBit7,InputBit8,
            InputBit9, InputBit10, InputBit11, InputBit12,
            InputBit13, InputBit14, InputBit15
ST WORD_Output
```

Representation in ST

Representation:

```
WORD_Output := BIT_TO_WORD (InputBit0, InputBit1,
    InputBit2, InputBit3, InputBit4, InputBit5,
    InputBit6, InputBit7, InputBit8, InputBit9,
    InputBit10, InputBit11, InputBit12, InputBit13,
    InputBit14, InputBit15) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
InputBit0	BOOL	Input bit 0
:	:	:
InputBit15	BOOL	Input bit 15

Description of the output parameter:

Parameter	Data type	Meaning
WORD_ Output	WORD	Output value

BOOL_TO_***: Type conversion

112

Description

Function description

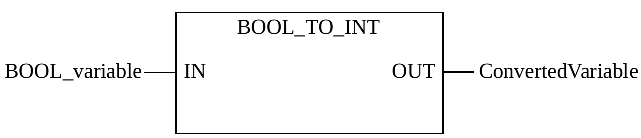
The function converts an input value of the `BOOL` data type to a `BYTE`, `WORD`, `DWORD`, `INT`, `DINT`, `UINT`, `UDINT`, `REAL` or `TIME` data type.
The input value is written in the lowest bit of the output. All other output bits are set to zero.
`EN` and `ENO` can be configured as additional parameters.
(The output `ENO` is not used for `BOOL_TO_REAL`; it always has the value "1".)

Available functions

List of available functions:
- `BOOL_TO_BYTE`
- `BOOL_TO_WORD`
- `BOOL_TO_DWORD`
- `BOOL_TO_INT`
- `BOOL_TO_DINT`
- `BOOL_TO_UINT`
- `BOOL_TO_UDINT`
- `BOOL_TO_REAL`
- `BOOL_TO_TIME`

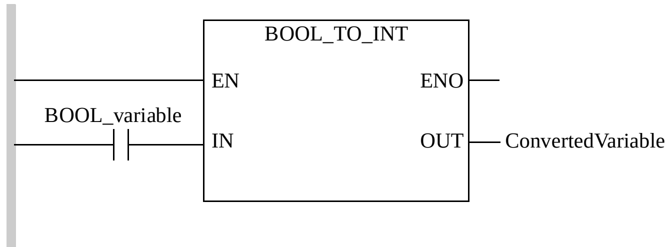
Representation in FBD

Representation of an Integer application:



Representation in LD

Representation of an Integer application:



Representation in IL

Representation of an Integer application:

```
LD BOOL_variable
BOOL_TO_INT
ST ConvertedVariable
```

Representation in ST

Representation of an Integer application:

```
ConvertedVariable := BOOL_TO_INT (BOOL_variable) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
`BOOL_variable`	`BOOL`	Input value

Description of the output parameter:

Parameter	Data type	Meaning
`ConvertedVariable`	`BYTE`, `WORD`, `DWORD`, `INT`, `DINT`, `UINT`, `UDINT`, `REAL`, `TIME`	Output value

BYTE_AS_WORD: Type conversion

113

Description

Function description

The function converts 2 input values of the `BYTE` data type to an output value of the `WORD` data type.
The input values are assigned to the word at the output according to the input names.
`EN` and `ENO` can be configured as additional parameters.

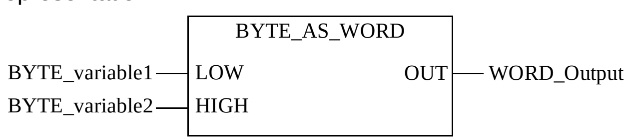
Formula

Block formula:

OUT = {HIGH, LOW}

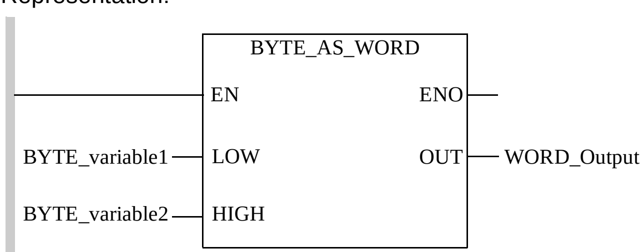
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD BYTE_variable1
BYTE_AS_WORD BYTE_variable2
ST WORD_Output
```

Representation in ST

Representation:

```
WORD_Output := BYTE_AS_WORD (BYTE_variable1,
               BYTE_variable2) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
BYTE_variable1	BYTE	least significant byte
BYTE_variable2	BYTE	most significant byte

Description of the output parameter:

Parameter	Data type	Meaning
WORD_Output	WORD	Output value

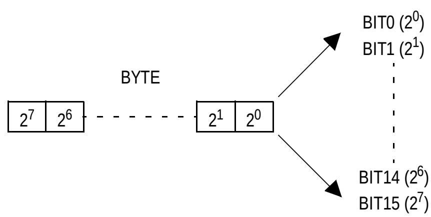
BYTE_TO_BIT: Type conversion

114

Description

Function description

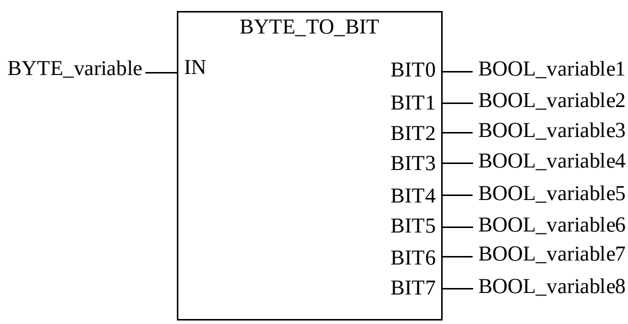
The procedure converts an input value of the `BYTE` data type to 8 output values of the `BOOL` data type.
The individual bits of the byte at the input are assigned to the outputs according to the output names.



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

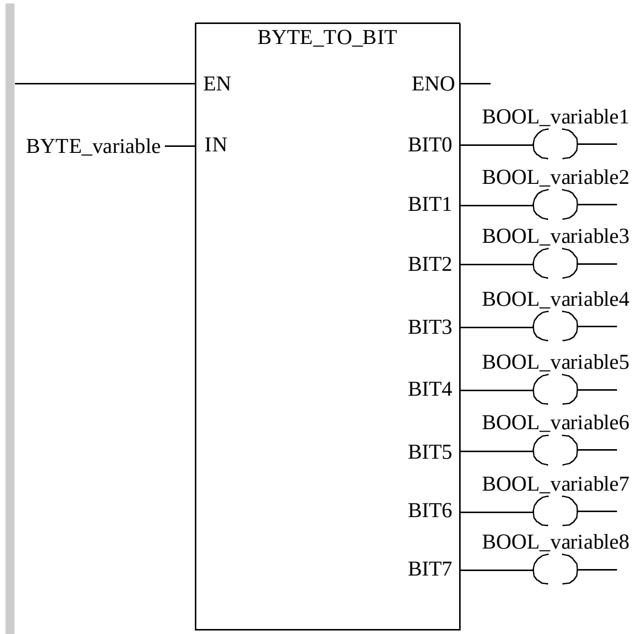
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD BYTE_variable
BYTE_TO_BIT BOOL_variable1, BOOL_variable2, BOOL_variable3,
            BOOL_variable4, BOOL_variable5, BOOL_variable6,
            BOOL_variable7, BOOL_variable8
```

Representation in ST

Representation:

```
BYTE_TO_BIT (BYTE_variable, BOOL_variable1, BOOL_variable2,
             BOOL_variable3, BOOL_variable4, BOOL_variable5,
            BOOL_variable6, BOOL_variable7, BOOL_variable8);
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
BYTE_variable	BYTE	Input

Description of the output parameter:

Parameter	Data type	Meaning
BOOL_variable1	BOOL	Output bit 0
BOOL_variable2	BOOL	Output bit 1
:	:	:
BOOL_variable8	BOOL	Output bit 7

BYTE_TO_***: Type conversion

115

Description

Function description

The function converts an input value of the `BYTE` data type to a `BOOL`, `WORD`, `DWORD`, `INT`, `DINT`, `UINT`, `UDINT`, `REAL` or `TIME` data type.

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

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

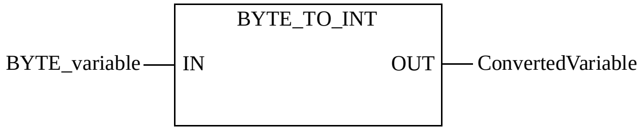
`EN` and `ENO` can be configured as additional parameters.
(The output `ENO` is not used for `BYTE_TO_REAL`; it always has the value "1".)

Available functions

List of available functions:
- `BYTE_TO_BOOL`
- `BYTE_TO_WORD`
- `BYTE_TO_DWORD`
- `BYTE_TO_INT`
- `BYTE_TO_DINT`
- `BYTE_TO_UINT`
- `BYTE_TO_UDINT`
- `BYTE_TO_REAL`
- `BYTE_TO_TIME`

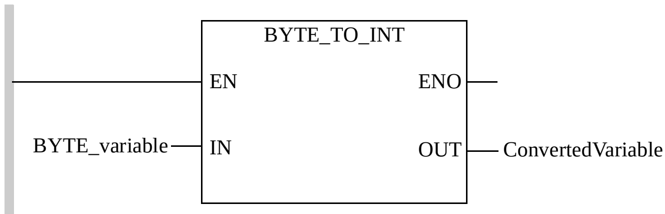
Representation in FBD

Representation of an Integer application:



Representation in LD

Representation of an Integer application:



Representation in IL

Representation of an Integer application:

```
LD BYTE_variable
BYTE_TO_INT
ST ConvertedVariable
```

Representation in ST

Representation of an Integer application:

```
ConvertedVariable := BYTE_TO_INT (BYTE_variable) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
BYTE_variable	BYTE	Input value

Description of output parameters:

Parameter	Data type	Meaning
ConvertedVariable	BOOL, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME	Output value

Runtime error

Error handling is dependent on the function:

- `BYTE_TO_REAL`
  The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1 and the status is stored in %SW17 (See *Description of system words %SW12 to %SW18, p. 451*) , if an illegal floating-point decimal is generated during the conversion process.
- all other Functions
  The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) and system word %SW17 (See *Description of system words %SW12 to %SW18, p. 451*) are not used.

DATE_TO_STRING: Conversion of a variable in DATE format into a character string

116

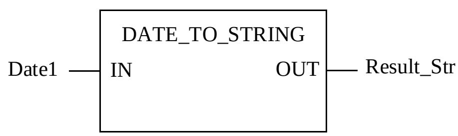
Description

Function description

The `DATE_TO_STRING` function converts a variable in DATE format into a character string.
The additional parameters `EN` and `ENO` can be configured.

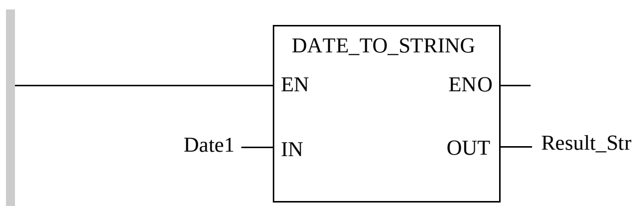
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Date1
DATE_TO_STRING
ST Result_Str
```

Representation in ST

Representation:

```
Result_Str:= DATE_TO_STRING(Date1);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Date1	DATE	Date to be converted into character string format.

The following table describes the output parameters:

Parameter	Type	Comment
Result_Str	STRING	Result_Str is a string of 10 characters which contains a date (not including hours) in the following format: YYYY-MM-DD. **Example**: ‘2000-12-27’ **Note:** if the maximum size of the string Result_Str is greater than 10, Result_Str is completed by the end of string characters (16#00).

Runtime errors

If the string Result_Str is too short to contain the date (length of less than 10 characters), the date is truncated and the bit **%S15** (See *Description of system bits %S9 to %S13, p. 447*) is set to 1.
If Date1 in not interpretable and coherent in DATE format, the system bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) is set to 1 and Result_Str =’****-**-**’

DBCD_TO_***: Conversion of a double BCD integer into binary

117

Description

Function description

The DBCD_TO_*** function converts a double integer in Binary Code Decimal (DBCD) format into a double binary coded integer.
The additional parameters EN and ENO can be configured.

Available functions

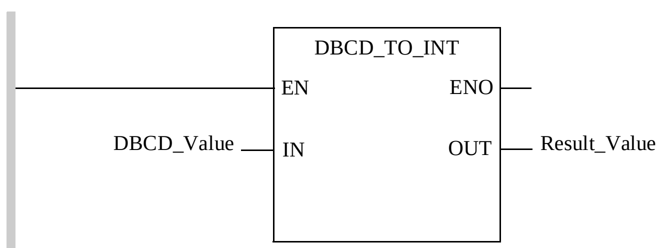
The available functions are as follows:
- DBCD_TO_INT,
- DBCD_TO_DINT.

Representation in FBD

Representation applied to an integer:

Representation in LD

Representation applied to an integer:



Representation in IL

Representation applied to an integer:

```
LD DBCD_Value
DBCD_TO_INT
ST Result_Value
```

Representation in ST

Representation applied to an integer:

```
Result_Value:= DBCD_TO_INT(DBCD_Value);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
DBCD_Value	DINT	Double integer in BCD format. **Example**: DBCD_Value = 16#32767

The following table describes the output parameters:

Parameter	Type	Comment
Result_Value	INT, DINT	Result_Value is an integer or double integer in binary code. **Example**: with the value provided in the example in the previous table, Result_Value = 32767

Runtime errors

The bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) is set to 1 when:

- the value to be converted is not a value coded in BCD. The result of the function then returns the value of the first half-byte by default.
- for the function DBCD_TO_INT, the value to be converted is greater in BCD than 32767. The result of the function is then -1.

DEG_TO_RAD : Conversion of degrees to radians

118

Description

Function description

The `DEG_TO_RAD` function converts an angle expressed in degrees into radians. The additional parameters `EN` and `ENO` can be configured.

Formula

The formula is as follows:

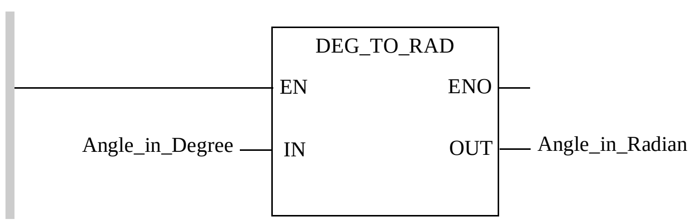
Angle_in_Radian = DEG_TO_RAD (Angle_in_Degree)

Representation in FBD

Representation:

Representation in LD

Representation:



Representation in IL

Representation:

```
LD Angle_in_Degree
DEG_TO_RAD
ST Angle_in_Radian
```

Representation in ST

Representation:

```
Angle_in_Radian:= DEG_TO_RAD(Angle_in_Degree);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Angle_in_Degree	REAL	Angle expressed in degrees. -737280.0 < Angle_in_Degree < +737280.0.

The following table describes the output parameters:

Parameter	Type	Comment
Angle_in_Radian	REAL	Value of Angle expressed in radians. $-\pi \leq$ Angle_in_Radian $\leq +\pi$.

Runtime errors

When Angle_in_Degree is situated outside the interval]-737280.0, +73780.0[, the system bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) changes to 1, the system word **%SW17** (See *Description of system words %SW12 to %SW18, p. 451*) indicates the type of fault and the result displayed is 1.#NAN.

DINT_AS_WORD: Type conversion

119

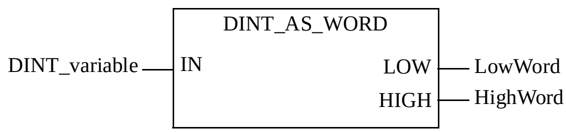
Description

Function description

The procedure converts an input value of the `DINT` data type to 2 output values of the `WORD` data type.
The individual words of the `DINT` input are assigned to the outputs according to the output names.
`EN` and `ENO` can be configured as additional parameters.

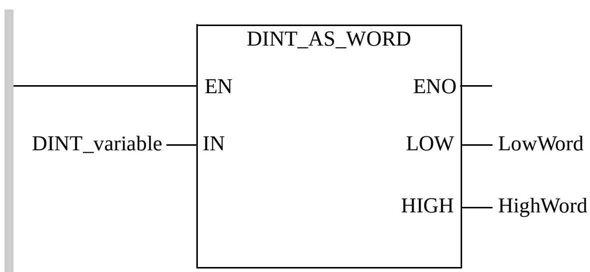
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD DINT_variable
DINT_AS_WORD LowWord, HighWord
```

Representation in ST

Representation:

```
DINT_AS_WORD (DINT_variable, LowWord, HighWord);
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
DINT_variable	DINT	Input

Description of the output parameter:

Parameter	Data type	Meaning
LowWord	WORD	least significant word
HighWord	WORD	most significant word

DINT_TO_***: Type conversion

120

Description

Function description

The function converts an input value of the `DINT` data type to a `BOOL`, `BYTE`, `WORD`, `DWORD`, `INT`, `UINT` `UDINT`, `REAL` or `TIME` output value.

> **Note:** The function converts strictly in accordance with IEC rules. Since this function has been realized as a generic function, there will also be a few illogical conversions, e.g. `DINT_TO_BOOL`.

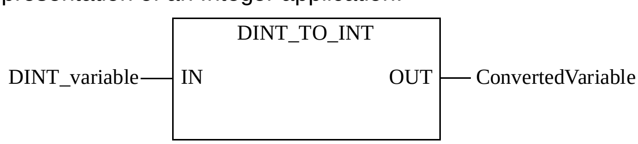
When converting the data type `DINT` to the `BOOL`, `BYTE`, `WORD`, `INT` or `UINT` data type, the least significant bits of the input value are transferred to the output. Negative input values cannot be converted into data types `UINT`, `UDINT` or `TIME`. `EN` and `ENO` can be configured as additional parameters.

Available functions

List of available functions:
- `DINT_TO_BOOL`
- `DINT_TO_BYTE`
- `DINT_TO_WORD`
- `DINT_TO_DWORD`
- `DINT_TO_INT`
- `DINT_TO_UINT`
- `DINT_TO_UDINT`
- `DINT_TO_REAL`
- `DINT_TO_TIME`

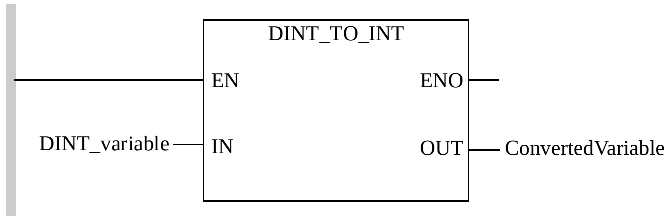
Representation in FBD

Representation of an Integer application:



Representation in LD

Representation of an Integer application:



Representation in IL

Representation of an Integer application:

```
LD DINT_variable
DINT_TO_INT
ST ConvertedVariable
```

Representation in ST

Representation of an Integer application:

```
ConvertedVariable := DINT_TO_INT (DINT_variable) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
DINT_variable	DINT,	Input value

Description of output parameters:

Parameter	Data type	Meaning
ConvertedVariable	BOOL, BYTE, WORD, DWORD, INT, UINT, UDINT, REAL, TIME	Output value

Runtime error

The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1, if

- the value range on the output is exceeded (numeric data types)
- a negative input value is to be converted into an `UDINT`-, `UINT` or `TIME` output value.
- an unauthorized floating point number is created during the conversion into the `REAL` data type. In this case, the status is also placed in %SW17 (See *Description of system words %SW12 to %SW18, p. 451*).

The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) and system word %SW17 (See *Description of system words %SW12 to %SW18, p. 451*) are **not** used when data types are converted:

- `BOOL`
- `BYTE`
- `WORD`
- `DWORD`

DINT_TO_DBCD: Conversion of a double binary coded integer into a double Binary Coded Decimal integer

121

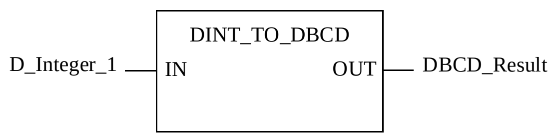
Description

Function description

The DINT_TO_DBCD function carries out the conversion of a double binary coded integer into an integer in Double Binary Coded Decimal (DBCD) format.
The additional parameters EN and ENO can be configured.

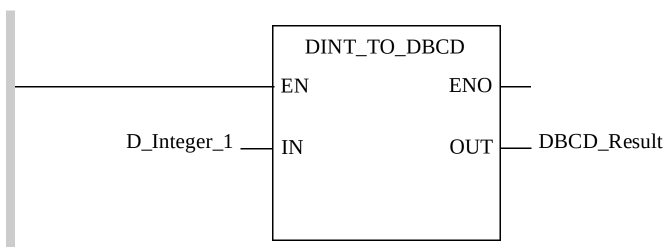
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD D_Integer_1
DINT_TO_BCD
ST DBCD_Result
```

Representation in ST

Representation:

```
DBCD_Result:= DINT_TO_BCD(D_Integer_1);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`D_Integer_1`	`DINT`	Double binary coded integer between 0 and 99999999. **Example**: `D_Integer_1` = 888888

The following table describes the output parameters:

Parameter	Type	Comment
`DBCD_Result`	`DINT`	`DBCD_Result` is a double integer in BCD format. **Example**: with the value provided in the example in the previous table, `DBCD_Result` = 16#00888888

Runtime errors

The bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) is set to 1 when the value to be converted is not a value between 0 and 99999999. The result of the function then returns the value of the input parameter.

DT_TO_STRING: Conversion of a variable in DT format into a character string

122

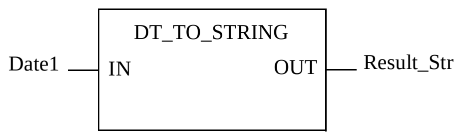
Description

Function description

The `DT_TO_STRING` function converts a variable in DT format into a character string.
The additional parameters `EN` and `ENO` can be configured.

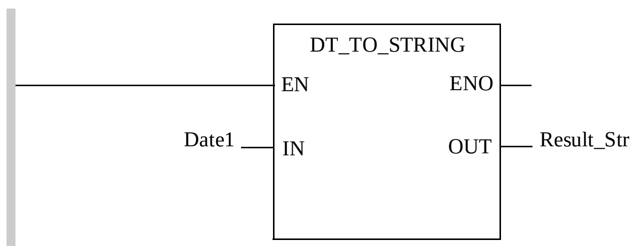
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Date1
DT_TO_STRING
ST Result_Str
```

Representation in ST

Representation:

```
Result_Str:= DT_TO_STRING(Date1);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Date1	DT	Date to be converted into character string format.

The following table describes the output parameters:

Parameter	Type	Comment
Result_Str	STRING	Result_Str is a string of 19 characters which contains a date (including hours) in the following format: YYYY-MM-DD-HH:MM:SS. **Example**: '2000-12-27-23:15:50' **Note:** if the maximum size of the string Result_Str is greater than 19, Result_Str is completed by the end of string characters (16#00).

Runtime errors

If the string Result_Str is too short to contain the date (length of less than 19 characters), the date is truncated and the bit **%S15** (See *Description of system bits %S9 to %S13, p. 447*) is set to 1.
If Date1 in not interpretable and coherent in format DT, the system bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) is set to 1 and Result_Str ='****-**-**-**:**:**'.

DWORD_TO_***: Type conversion

123

Description

Function description

The function converts an input value of the DWORD data type to a BOOL, BYTE, WORD, INT, DINT, UINT, UDINT, REAL or TIME data type.

> **Note:** The function converts strictly in accordance with IEC rules. Since this function has been realized as a generic function, there will also be a few illogical conversions, e.g. DWORD_TO_BOOL.

When converting the data type DWORD to the BOOL, BYTE, WORD, INT or UINT data type, the least significant bits of the input value are transferred to the output.
EN and ENO can be configured as additional parameters.
(The output ENO is not used for DWORD_TO_REAL; it always has the value "1".)

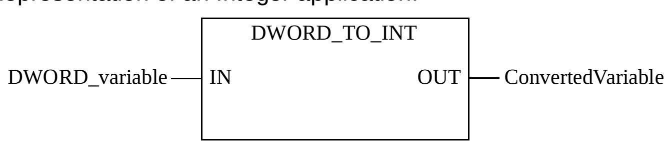
Available functions

List of available functions:

- DWORD_TO_BOOL
- DWORD_TO_BYTE
- DWORD_TO_WORD
- DWORD_TO_INT
- DWORD_TO_DINT
- DWORD_TO_UINT
- DWORD_TO_UDINT
- DWORD_TO_REAL
- DWORD_TO_TIME

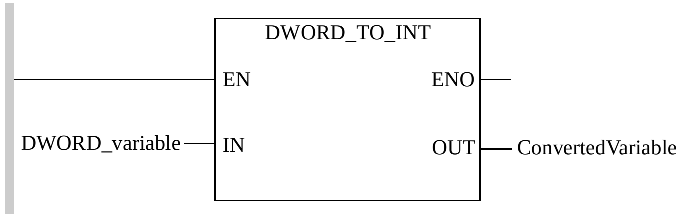
Representation in FBD

Representation of an Integer application:



Representation in LD

Representation of an Integer application:



Representation in IL

Representation of an Integer application:

```
LD DWORD_variable
DWORD_TO_INT
ST ConvertedVariable
```

Representation in ST

Representation of an Integer application:

```
ConvertedVariable := DWORD_TO_INT (DWORD_variable) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
DWORD_variable	DWORD	Input value

Description of output parameters:

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

Runtime error

Error handling is dependent on the function:

- DWORD_TO_REAL
  The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1 and the status is stored in %SW17 (See *Description of system words %SW12 to %SW18, p. 451*) , if an illegal floating-point decimal is generated during the conversion process.
- all other Functions
  The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) and system word %SW17 (See *Description of system words %SW12 to %SW18, p. 451*) are not used.

GRAY_TO_INT: Conversion of an integer in Gray code into a binary coded integer

124

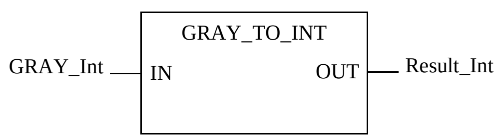
Description

Function description

The `GRAY_TO_INT` function converts an integer expressed in GRAY code into a binary coded integer.
The additional parameters `EN` and `ENO` can be configured.

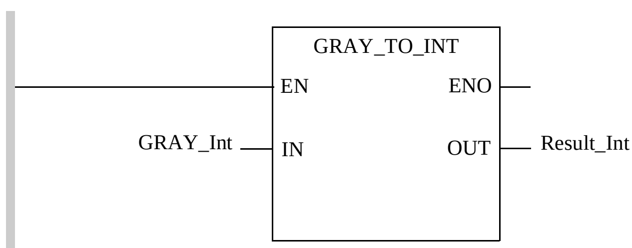
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD GRAY_Int
GRAY_TO_INT
ST Result_Int
```

Representation in ST

Representation:

```
Result_Int:= GRAY_TO_INT(GRAY_Int);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
GRAY_Int	INT	Integer expressed in GRAY code.

The following table describes the output parameters:

Parameter	Type	Comment
Result_Int	INT	Result_Int is a binary coded integer.

INT_AS_DINT: Concatenation of two integers to form a double integer

125

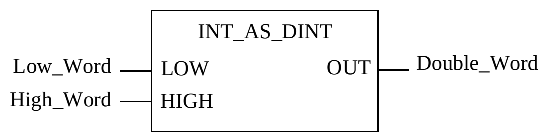
Description

Function description

The `INT_AS_DINT` function concatenates two integers to form a double integer. The additional parameters `EN` and `ENO` can be configured.

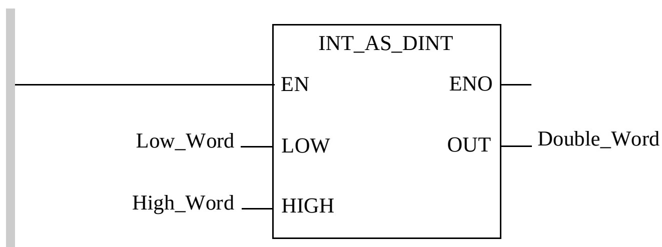
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Low_Word
INT_AS_DINT High_Word
ST Double_Word
```

Representation in ST

Representation:

```
Double_Word:= INT_AS_DINT(Low_Word, High_Word);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Low_Word	INT	Integer which must become the least significant word of a double integer Double_Word. **Example**: Low_Word contains 16#5678.
High_Word	INT	Integer which must become the most significant word of a double integer Double_Word. **Example**: High_Word contains 16#1234.

The following table describes the output parameters:

Parameter	Type	Comment
Double_Word	DINT	Double integer composed of two integers Low_Word for the least significant and High_Word for the most significant. **Example**: for the values in the example provided in the previous table, Double_Word contains 16#12345678.

INT_TO_***: Type conversion

126

Description

Function description

The function converts an input value of the INT data type to a BOOL, BYTE, WORD, DWORD, DINT, UINT, UDINT, REAL or TIME output value.

Note: The function converts strictly in accordance with IEC rules. Since this function has been realized as a generic function, there will also be a few illogical conversions, e.g. INT_TO_BOOL.

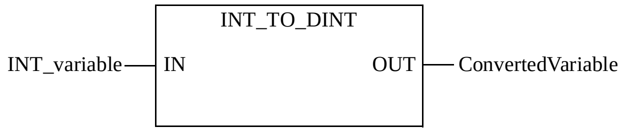
Negative input values cannot be converted into data types UINT, UDINT or TIME.
When converting an input value from the data type INT into data type WORD, the bit pattern from the input is transferred to the output without being modified.
When converting an input value of data type INT into the data types BOOL or BYTE, the least significant bits of the input are transferred to the output.
EN and ENO can be configured as additional parameters.

Available functions

List of available functions:
- INT_TO_BOOL
- INT_TO_BYTE
- INT_TO_WORD
- INT_TO_DWORD
- INT_TO_DINT
- INT_TO_UINT
- INT_TO_UDINT
- INT_TO_REAL
- INT_TO_TIME

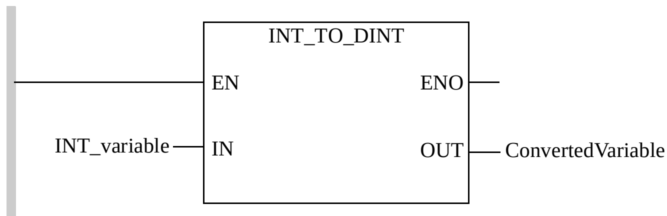
Representation in FBD

Representation of a double integer application:



Representation in LD

Representation of a double integer application:



Representation in IL

Representation of a double integer application:

```
LD INT_variable
INT_TO_DINT
ST ConvertedVariable
```

Representation in ST

Representation of a double integer application:

```
ConvertedVariable := INT_TO_DINT (INT_variable) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
INT_variable	INT	Input value

Description of output parameters:

Parameter	Data type	Meaning
ConvertedVariable	BOOL, BYTE, DWORD, WORD, DINT, UINT, UDINT, REAL, TIME	Output value

Runtime error

The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1, if

- the value range on the output is exceeded (numeric data types)
- a negative input value is to be converted into an `UDINT`-, `UINT` or `TIME` output value.
- an unauthorized floating point number is created during the conversion into the `REAL` data type. In this case, the status is also placed in %SW17 (See *Description of system words %SW12 to %SW18, p. 451*).

The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) and system word %SW17 (See *Description of system words %SW12 to %SW18, p. 451*) are **not** used when data types are converted:

- `BOOL`
- `BYTE`
- `WORD`
- `DWORD`

INT_TO_BCD: Conversion of a binary coded integer into a Binary Coded Decimal integer

127

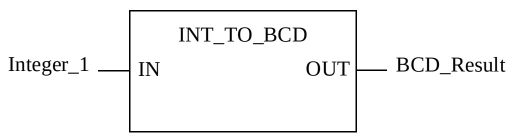
Description

Function description

The `INT_TO_BCD` function carries out the conversion of a binary coded integer into an integer in Binary Coded Decimal (`BCD`) format.
The additional parameters `EN` and `ENO` can be configured.

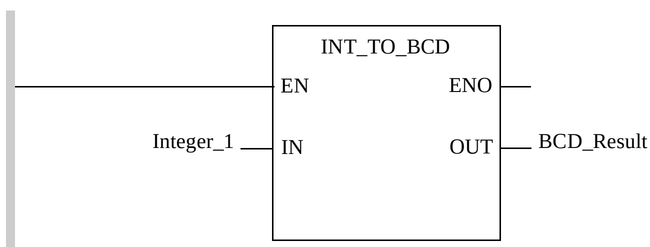
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Integer_1
INT_TO_BCD
ST BCD_Result
```

Representation in ST

Representation:

```
BCD_Result := INT_TO_BCD(Integer_1);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`Integer_1`	`INT`	Binary coded integer between 0 and 9999. **Example**: `Integer_1` = 99

The following table describes the output parameters:

Parameter	Type	Comment
`BCD_Result`	`INT`	`BCD_Result` is a BCD integer. **Example**: with the value provided in the example in the previous table, `BCD_Result` = 16#99

Runtime errors

The bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) is set to 1 when the value to be converted is not a value between 0 and 9999. The result of the function then returns the value of the input parameter.

INT_TO_DBCD: Conversion of a binary coded integer into a double Binary Coded Decimal integer

128

Description

Function description

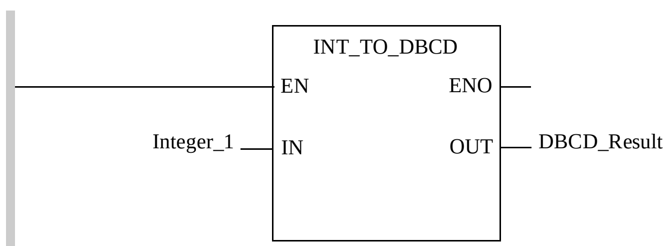
The `INT_TO_DBCD` function carries out the conversion of a binary coded integer into an integer in Double Binary Coded Decimal (`DBCD`) format. This function is useful when converting numbers with BCD coding greater than 32768.
The additional parameters `EN` and `ENO` can be configured.

Representation in FBD

Representation:

Representation in LD

Representation:



Representation in IL

Representation:

```
LD Integer_1
INT_TO_BCD
ST DBCD_Result
```

Representation in ST

Representation:
`DBCD_Result:= INT_TO_BCD(Integer_1);`

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`Integer_1`	`INT`	Binary coded integer between 0 and 32768. **Example**: `Integer_1` = 30000

The following table describes the output parameters:

Parameter	Type	Comment
`DBCD_Result`	`DINT`	`DBCD_Result` is a double integer in BCD format. **Example**: with the value provided in the example in the previous table, `DBCD_Result` = 16#0030000

Runtime errors

The bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) is set to 1 when the value to be converted is not a value between 0 and 99999999 or when the value to be converted is negative. The result of the function then returns the value of the input parameter.

RAD_TO_DEG: Conversion of radians to degrees

129

Description

Function description

The `RAD_TO_DEG` function converts an angle expressed in radians into degrees. The additional parameters `EN` and `ENO` can be configured.

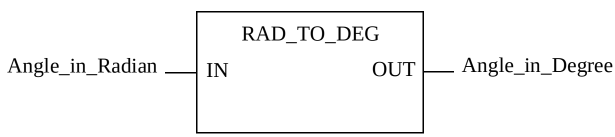
Formula

The formula is as follows:

RAD_TO_DEG (Angle_in_Radian) = Angle_in_Degree

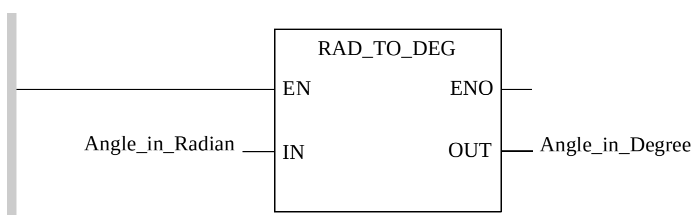
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Angle_in_Radian
RAD_TO_DEG
ST Angle_in_Degree
```

Representation in ST

Representation:

```
Angle_in_Degree:= RAD_TO_DEG(Angle_in_Radian);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`Angle_in_Radian`	`REAL`	Value of Angle expressed in radians. $-4096\pi \leq$ `Angle_in_Radian` $\leq +4096\pi$

The following table describes the output parameters:

Parameter	Type	Comment
`Angle_in_Degree`	`REAL`	Angle expressed in degrees. $-360 <$ `Angle_in_Degree` $< +360$.

Runtime errors

When `Angle_in_Degree` is situated outside the interval $]-4096\,\pi, 4096\,\pi[$, the system bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) changes to 1 and the system bit **%SW17** (See *Description of system words %SW12 to %SW18, p. 451*) indicates the type of fault.

REAL_AS_WORD: Type conversion

130

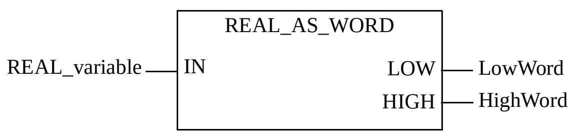
Description

Function description

The procedure converts an input value of the `REAL` data type to 2 output values of the `WORD` data type.
The individual words of the `REAL` input are assigned to the outputs according to the output names.
`EN` and `ENO` can be configured as additional parameters.

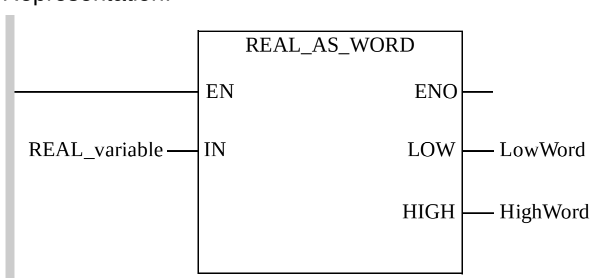
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD REAL_variable
REAL_AS_WORD LowWord, HighWord
```

Representation in ST

Representation:
`REAL_AS_WORD (REAL_variable, LowWord, HighWord);`

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
REAL_variable	REAL	Input

Description of output parameters:

Parameter	Data type	Meaning
LowWord	WORD	least significant word
HighWord	WORD	most significant word

Runtime error

The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1, if an unauthorized floating point number is set at the input.

REAL_TO_***: Type conversion

131

Description

Function description

The function converts an input value of the REAL data type to a BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT data type or into the TIME data type.

Note: The function converts strictly in accordance with IEC rules. Since this function has been realized as a generic function, there will also be a few illogical conversions, e.g. REAL_TO_BOOL.

When converting to BOOL, BYTE, WORD, the least significant bits of the input value are transferred to the output. A runtime error message is not given and ENO remains 1.

When converting to INT, DINT, UINT, UDINT and TIME, the IEC 559 rules for rounding are applied.

EN and ENO can be configured as additional parameters.
(The output ENO is not used for REAL_TO_BOOL, REAL_TO_BYTE, REAL_TO_WORD and REAL_TO_DWORD; it always has the value "1".)

Available functions

List of available functions:
- REAL_TO_BOOL
- REAL_TO_BYTE
- REAL_TO_WORD
- REAL_TO_DWORD
- REAL_TO_INT
- REAL_TO_DINT
- REAL_TO_UINT
- REAL_TO_UDINT
- REAL_TO_TIME

Example

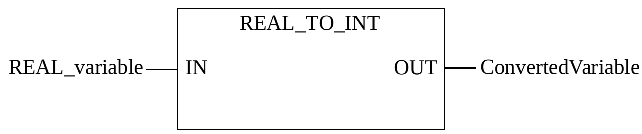
The following example shows how the IEC 559 rounding is applied.
1,4 -> 1
1,5 -> 2
2,5 -> 2
3,5 -> 4
4,5 -> 4
4,6 -> 5

Negative input values

Negative input values cannot be converted into data types `UINT`, `UDINT` or `TIME`.

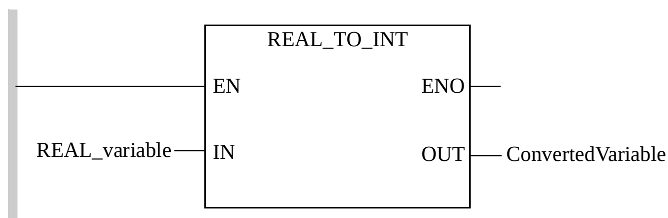
Representation in FBD

Representation of an Integer application:



Representation in LD

Representation of an Integer application:



Representation in IL

Representation of an Integer application:

```
LD REAL_variable
REAL_TO_INT
ST ConvertedVariable
```

Representation in ST

Representation of an Integer application:

```
ConvertedVariable := REAL_TO_INT (REAL_variable) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
REAL_variable	REAL	Input value

Description of output parameters:

Parameter	Data type	Meaning
ConvertedVariable	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, TIME	Output value

Runtime error

The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1, if

- an unauthorized floating point number is set at the input
- the value range on the output is exceeded (numeric data types)
- a negative input value is to be converted into an UDINT-, UINT or TIME output value.
- an unauthorized floating point number is created during the conversion into the REAL data type. In this case, the status is also placed in %SW17 (See *Description of system words %SW12 to %SW18, p. 451*).

The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) and system word %SW17 (See *Description of system words %SW12 to %SW18, p. 451*) are **not** used when data types are converted:

- BOOL
- BYTE
- WORD
- DWORD

REAL_TRUNC_***: Type conversion

132

Description

Function description

The function converts (by truncating towards zero) a `REAL` data type input value to a output value of the `INT`, `DINT`, `UINT` or `UDINT` data type.
`EN` and `ENO` can be configured as additional parameters.

Available functions

List of available functions:
- `REAL_TRUNC_INT`
- `REAL_TRUNC_DINT`
- `REAL_TRUNC_UINT`
- `REAL_TRUNC_UDINT`

Example

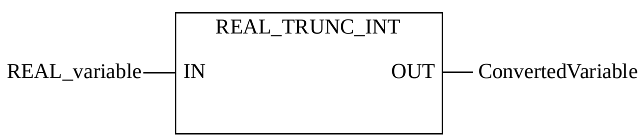
The following example shows the converting procedure.
1,6 -> 1
-1,6 -> -1
1,4 -> 1
-1,4 -> -1

Data type

Negative input values cannot be converted into data types `UDINT` or `UINT`.

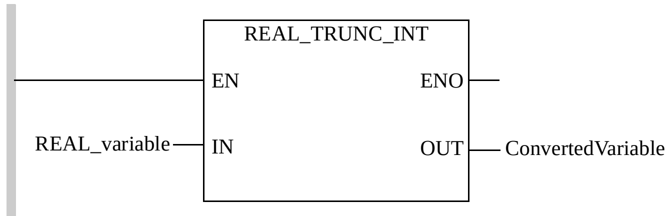
Representation in FBD

Representation of an Integer application:



Representation in LD

Representation of an Integer application:



Representation in IL

Representation of an Integer application:

```
LD REAL_variable
REAL_TRUNC_INT
ST ConvertedVariable
```

Representation in ST

Representation of an Integer application:

```
ConvertedVariable := REAL_TRUNC_INT (REAL_variable) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
REAL_variable	REAL	Input value

Description of the output parameter:

Parameter	Data type	Meaning
ConvertedVariable	INT, DINT, UINT, UDINT	Output value

Runtime error

The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1 and the status is stored in %SW17 (See *Description of system words %SW12 to %SW18, p. 451*) if

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

STRING_TO_*** : Conversion of a character string to a number of the INT, DINT or REAL type

133

Description

Description of the function

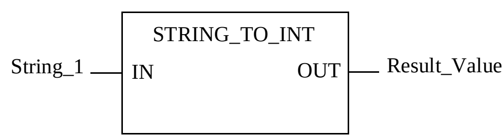
The `STRING_TO_***` function converts a character string into a one- or two-digit integer or into a real number.
This function is IEC 1131.
The additional parameters `EN` and `ENO` can be configured.

Available functions

The available functions are as follows:
- `STRING_TO_INT`,
- `STRING_TO_DINT`,
- `STRING_TO_REAL`.

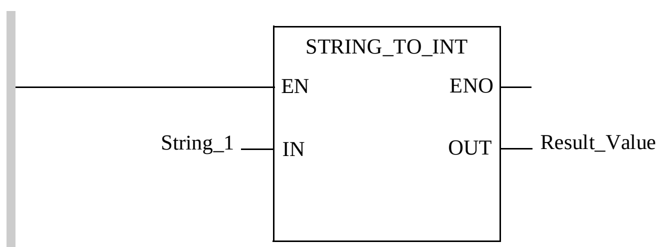
Representation en FBD

Representation applied to an integer:



Representation in LD

Representation applied to an integer:



Representation in IL

Representation applied to an integer:

```
LD String_1
STRING_TO_INT
ST Result_Value
```

Representation in ST

Representation applied to an integer:

```
Result_Value := STRING_TO_INT(String_1);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`String_1`	`STRING`	Character string **Example** : `String_1` = '-32500'

The following table describes the output parameters:

Parameter	Type	Comment
`Result_Value`	`INT`, `DINT`, `REAL`	`Result_Value` is an integer, a two-digit integer or a real number according to the function used. This result is the conversion of the character string `String_1` in accordance with the recommendations of standard IEC 1131. **Example** : with the value of the example in the above table, `Result_Value` = -32500.

Execution errors

The **%S18** (See *Description of system bits %S15 to %S21, p. 448*) bit is positioned at 1 when the content of the string to be converted is positioned outside of the boundaries of the type chosen (`INT`, `DINT` or `REAL`) or when one of the characters in the string is incorrect.

TYPE_AS_WORD: Type conversion

134

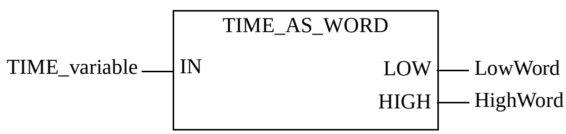
Description

Function description

The procedure converts an input value of the `TIME` data type to 2 output values of the `WORD` data type.
The individual words of the `TIME` input are assigned to the outputs according to the output names.
`EN` and `ENO` can be configured as additional parameters.

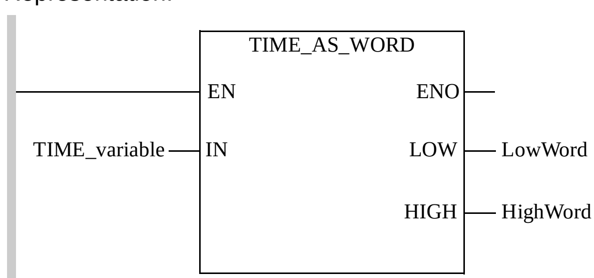
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD TIME_variable
TIME_AS_WORD LowWord, HighWord
```

Representation in ST

Representation:
`TIME_AS_WORD (REAL_variable, LowWord, HighWord);`

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
`TIME_variable`	`TIME`	Input

Description of output parameters:

Parameter	Data type	Meaning
`LowWord`	`WORD`	least significant word
`HighWord`	`WORD`	most significant word

TIME_TO_***: Type conversion

135

Description

Function description

The function converts an input value of the TIME data type to a BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT or REAL data type.

Note: The function converts strictly in accordance with IEC rules. Since this function has been realized as a generic function, there will also be a few illogical conversions, e.g. TIME_TO_BOOL.

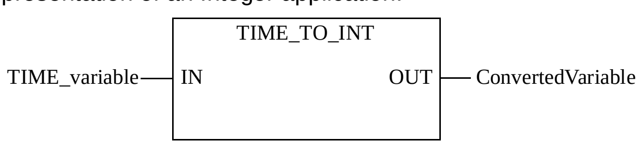
While converting an input value of data type TIME into an output value of data type BOOL, BYTE, WORD, INT or UINT, the least significant bits, respectively, are transferred from the input to the output.
EN and ENO can be configured as additional parameters.

Available functions

List of available functions:
- TIME_TO_BOOL
- TIME_TO_BYTE
- TIME_TO_WORD
- TIME_TO_DWORD
- TIME_TO_INT
- TIME_TO_DINT
- TIME_TO_UINT
- TIME_TO_UDINT
- TIME_TO_REAL

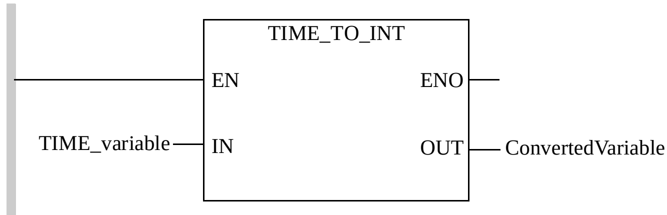
Representation in FBD

Representation of an Integer application:



Representation in LD

Representation of an Integer application:



Representation in IL

Representation of an Integer application:

```
LD TIME_variable
TIME_TO_INT
ST ConvertedVariable
```

Representation in ST

Representation of an Integer application:

```
ConvertedVariable := TIME_TO_INT (TIME_variable) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
`TIME_variable`	`TIME`	Input value

Description of the output parameter:

Parameter	Data type	Meaning
`ConvertedVariable`	`BOOL`, `BYTE`, `WORD`, `DWORD`, `INT`, `DINT`, `UINT`, `UDINT`, `REAL`	Output value

Runtime error

The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1, if

- the value range is exceeded at the output during the execution of the function.

TIME_TO_STRING: Conversion of a variable in TIME format into a character string

136

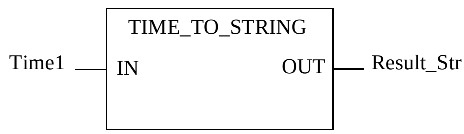
Description

Function description

The `TIME_TO_STRING` function converts a variable in TIME format into a character string.
The additional parameters `EN` and `ENO` can be configured.

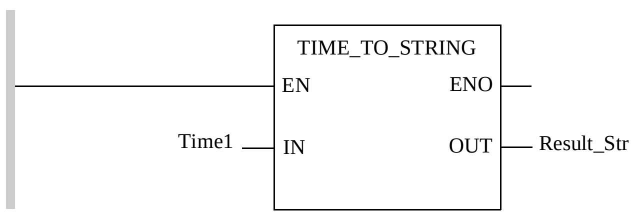
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Time1
TIME_TO_STRING
ST Result_Str
```

Representation in ST

Representation:

```
Result_Str:= TIME_TO_STRING(Time1);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`Time1`	`TIME`	Duration to be converted into character string format.

The following table describes the output parameters:

Parameter	Type	Comment
`Result_Str`	`STRING`	`Result_Str` is a string of 15 characters which contains a duration in the following format: `HHHHHH:MM:SS.D`. **Example**: ‘119304:38:49.5’ **Note**: if the maximum size of the string `Result_Str` is greater than 15, `Result_Str` is completed by the end of string characters (16#00).

Runtime errors

If the string `Result_Str` is too short to contain the converted value (less than 15 characters in length), it is truncated and the bit **%S15** (See *Description of system bits %S9 to %S13, p. 447*) is set to 1.

TOD_TO_STRING: Conversion of a variable in TOD format into a character string

137

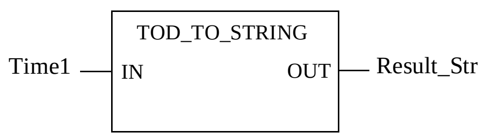
Description

Function description

The `TOD_TO_STRING` function converts a variable in TOD format into a character string.
The additional parameters `EN` and `ENO` can be configured.

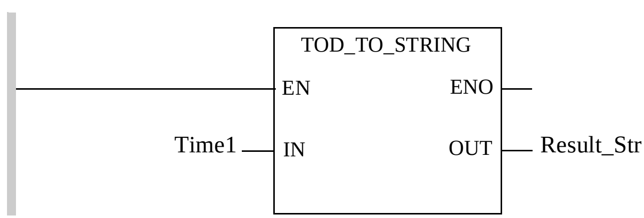
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD Time1
TOD_TO_STRING
ST Result_Str
```

Representation in ST

Representation:

```
Result_Str:= =TOD_TO_STRING(Time1);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
`Time1`	`TOD`	Time of day to be converted into character string format.

The following table describes the output parameters:

Parameter	Type	Comment
`Result_Str`	`STRING`	`Result_Str` is a string of 8 characters which contains a time of day in the following format: `HH:MM:SS`. **Example**: ‘04:38:49’ **Note**: if the maximum size of the string `Result_Str` is greater than 8, `Result_Str` is completed by the end of string characters (16#00).

Runtime errors

If the string `Result_Str` is too short to contain the converted value (less than 8 characters in length), it is truncated and the bit **%S15** (See *Description of system bits %S9 to %S13, p. 447*) is set to 1.

UDINT_AS_WORD: Type conversion

138

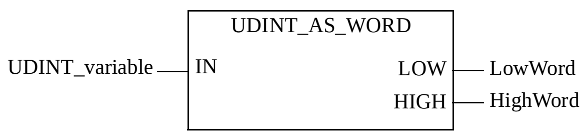
Description

Function description

The procedure converts an input value of the `UDINT` data type to 2 output values of the `WORD` data type.
The individual words of the `UDINT` input are assigned to the outputs according to the output names.
`EN` and `ENO` can be configured as additional parameters.

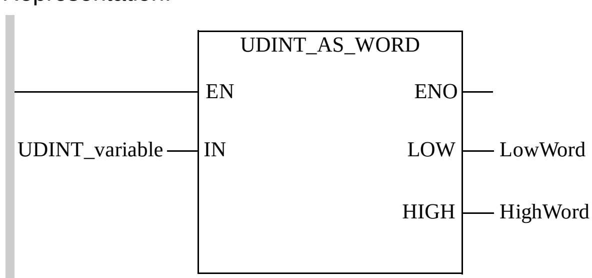
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD UDINT_variable
UDINT_AS_WORD LowWord, HighWord
```

Representation in ST

Representation:

```
UDINT_AS_WORD (UDINT_variable, LowWord, HighWord);
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
`UDINT_variable`	`UDINT`	Input

Description of output parameters:

Parameter	Data type	Meaning
`LowWord`	`WORD`	least significant word
`HighWord`	`WORD`	most significant word

UDINT_TO_***: Type conversion

139

Description

Function description

The function converts an input value of the `UDINT` data type to an output value of the `BOOL`, `BYTE`, `WORD`, `DWORD`, `INT`, `DINT`, `UINT`, `REAL` or `TIME` data type.

> **Note:** The function converts strictly in accordance with IEC rules. Since this function has been realized as a generic function, there will also be a few illogical conversions, e.g. `UDINT_TO_BOOL`.

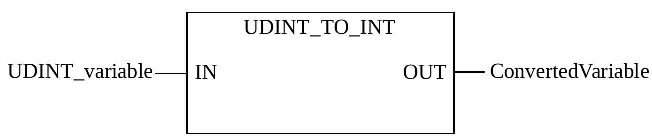
When converting the data type `DINT` to the `BOOL`, `BYTE`, `WORD`, `INT` or `UINT` data type, the least significant bits of the input value are transferred to the output.
`EN` and `ENO` can be configured as additional parameters.

Available functions

List of available functions:
- `UDINT_TO_BOOL`
- `UDINT_TO_BYTE`
- `UDINT_TO_WORD`
- `UDINT_TO_DWORD`
- `UDINT_TO_INT`
- `UDINT_TO_DINT`
- `UDINT_TO_UINT`
- `UDINT_TO_REAL`
- `UDINT_TO_TIME`

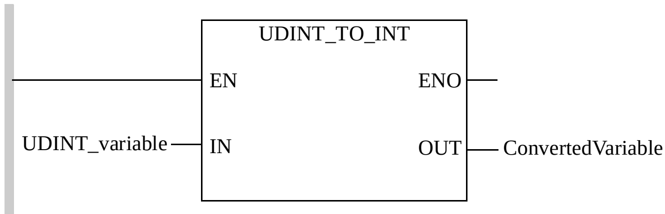
Representation in FBD

Representation of an Integer application:



Representation in LD

Representation of an Integer application:



Representation in IL

Representation of an Integer application:

```
LD UDINT_variable
UDINT_TO_INT
ST ConvertedVariable
```

Representation in ST

Representation of an Integer application:

```
ConvertedVariable := UDINT_TO_INT (UDINT_variable) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
`UDINT_variable`	`UDINT`	Input value

Description of output parameters:

Parameter	Data type	Meaning
`ConvertedVariable`	`BOOL`, `BYTE`, `WORD`, `DWORD`, `INT`, `DINT`, `UINT`, `REAL`, `TIME`	Output value

Runtime error

The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1, if

- the value range on the output is exceeded (numeric data types)
- a negative input value is to be converted into an `UDINT`-, `UINT` or `TIME` output value.
- an unauthorized floating point number is created during the conversion into the `REAL` data type. In this case, the status is also placed in %SW17 (See *Description of system words %SW12 to %SW18, p. 451*).

The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) and system word %SW17 (See *Description of system words %SW12 to %SW18, p. 451*) are **not** used when data types are converted:

- `BOOL`
- `BYTE`
- `WORD`
- `DWORD`

UINT_TO_***: Type conversion

140

Description

Function description

The function converts an input value of the UINT data type to an output value of the BOOL, BYTE, WORD, DWORD, INT, DINT, UDINT, REAL or TIME.data type.

> **Note:** The function converts strictly in accordance with IEC rules. Since this function has been realized as a generic function, there will also be a few illogical conversions, e.g. UINT_TO_BOOL.

When converting an input value from the data type UINT into data type WORD, the bit pattern from the input is transferred to the output without being modified.
When converting an input value of data type UINT into the data types BOOL or BYTE, the least significant bits of the input are transferred to the output.
EN and ENO can be configured as additional parameters.

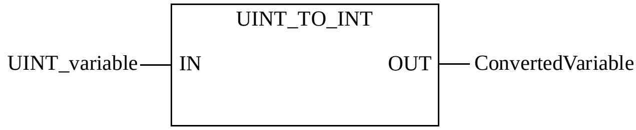
Available functions

List of available functions:

- UINT_TO_BOOL
- UINT_TO_BYTE
- UINT_TO_WORD
- UINT_TO_DWORD
- UINT_TO_INT
- UINT_TO_DINT
- UINT_TO_UDINT
- UINT_TO_REAL
- UINT_TO_TIME

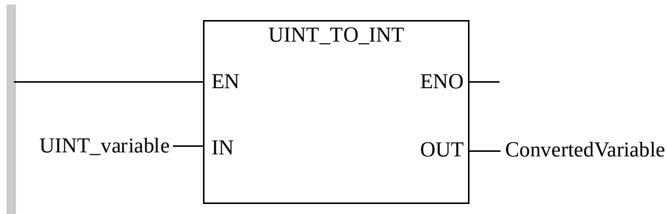
Representation in FBD

Representation of an Integer application:



Representation in LD

Representation of an Integer application:



Representation in IL

Representation of an Integer application:

```
LD UINT_variable
UINT_TO_INT
ST ConvertedVariable
```

Representation in ST

Representation of an Integer application:

```
ConvertedVariable := UINT_TO_INT (UINT_variable) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
UINT_variable	UINT	Input value

Description of output parameters:

Parameter	Data type	Meaning
ConvertedVariable	BOOL, BYTE, WORD, DWORD, INT, DINT, UDINT, REAL, TIME	Output value

Runtime error

The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1, if

- the value range on the output is exceeded (numeric data types)
- a negative input value is to be converted into an `UDINT`-, `UINT` or `TIME` output value.
- an unauthorized floating point number is created during the conversion into the `REAL` data type. In this case, the status is also placed in %SW17 (See *Description of system words %SW12 to %SW18, p. 451*).

The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) and system word %SW17 (See *Description of system words %SW12 to %SW18, p. 451*) are **not** used when data types are converted:

- `BOOL`
- `BYTE`
- `WORD`
- `DWORD`

WORD_AS_BYTE: Type conversion

141

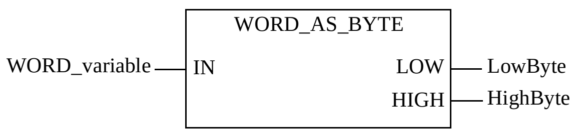
Description

Function description

The procedure converts an input value of the `WORD` data type to 2 output values of the `BYTE` data type.
The individual bytes of the word at the input are assigned to the outputs according to the output names.
`EN` and `ENO` can be configured as additional parameters.

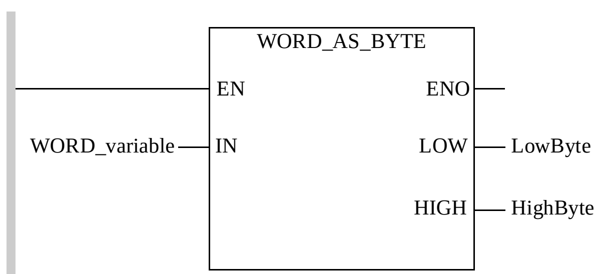
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD WORD_variable
WORD_AS_BYTE LowByte, HighByte
```

Representation in ST

Representation:

```
WORD_AS_BYTE (WORD_variable, LowByte, HighByte);
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
WORD_ variable	WORD	Input

Description of the output parameter:

Parameter	Data type	Meaning
LowByte	BYTE	least significant byte
HighByte	BYTE	most significant byte

WORD_AS_DINT: Type conversion

142

Description

Function description

The function converts 2 input values of the `WORD` data type to an output of the `DINT` data type.
The input values are assigned to the word at the output according to the input names.
`EN` and `ENO` can be configured as additional parameters.

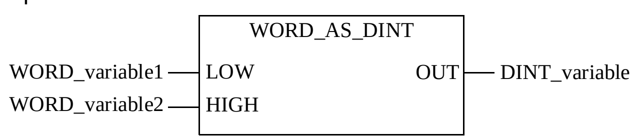
Formula

Block formula:

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

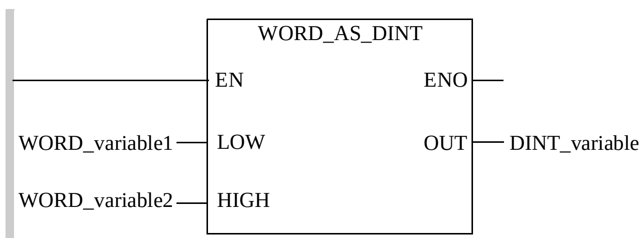
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD WORD_variable1
WORD_AS_DINT WORD_variable2
ST DINT_variable
```

Representation in ST

Representation:

```
DINT_variable := WORD_AS_DINT (WORD_variable1,
                 WORD_variable2) ;
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
WORD_ variable1	WORD	least significant word
WORD_ variable2	WORD	most significant word

Description of the output parameter:

Parameter	Data type	Meaning
DINT_ variable	DINT	Output value

WORD_AS_REAL: Type conversion

143

Description

Function description

The procedure converts an input value of the 2`WORD data type to` output values of the `REAL` data type.
The input values are assigned to the word at the output according to the input names.
`EN` and `ENO` can be configured as additional parameters.

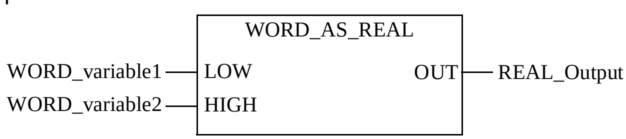
Formula

Block formula:

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

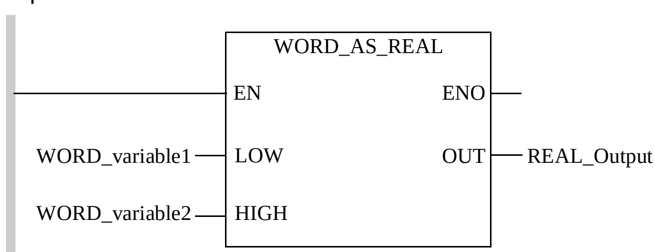
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD WORD_variable1
WORD_AS_REAL WORD_variable2, REAL_Output
```

Representation in ST

Representation:

```
WORD_AS_REAL (WORD_variable1, WORD_variable2,
               REAL_Output) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
WORD_variable1	WORD	least significant byte
WORD_variable2	WORD	most significant byte

Description of output parameters:

Parameter	Data type	Meaning
REAL_Output	REAL	Output value

Runtime error

The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1, if

- an unauthorized floating-point number is set at the input
- an unauthorized floating-point number is created during the conversion into the REAL data type. In this case, the status is also placed in %SW17 (See *Description of system words %SW12 to %SW18, p. 451*).

WORD_AS_TIME: Type conversion

144

Description

Function description

The function converts 2 input values of the `WORD` data type to an output value of the `TIME` data type.
The input values are assigned to the word at the output according to the input names.
`EN` and `ENO` can be configured as additional parameters.

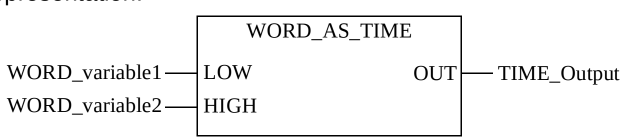
Formula

Block formula:

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

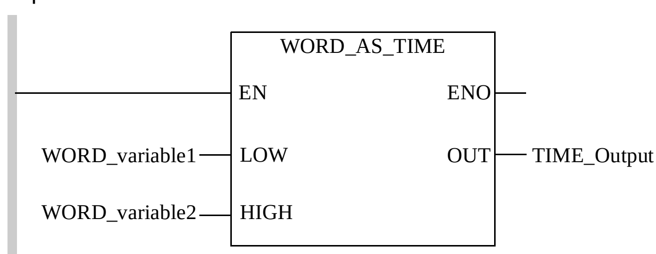
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD WORD_variable1
WORD_AS_TIME WORD_variable2
ST TIME_Output
```

Representation in ST

Representation:

```
TIME_Output := WORD_AS_TIME (WORD_variable1,
               WORD_variable2) ;
```

Parameter description

Description of input parameters:

Parameter	**Data type**	**Meaning**
WORD_variable1	WORD	least significant byte
WORD_variable2	WORD	most significant byte

Description of output parameters:

Parameter	**Data type**	**Meaning**
TIME_Output	TIME	Output value

WORD_AS_UDINT: Type conversion

145

Description

Function description

The function converts 2 input values of the `WORD` data type to an output value of the `UDINT` data type.
The input values are assigned to the word at the output according to the input names.
`EN` and `ENO` can be configured as additional parameters.

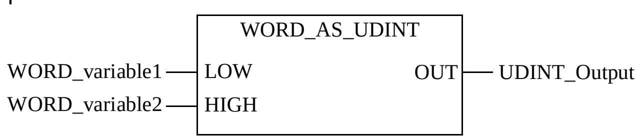
Formula

Block formula:

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

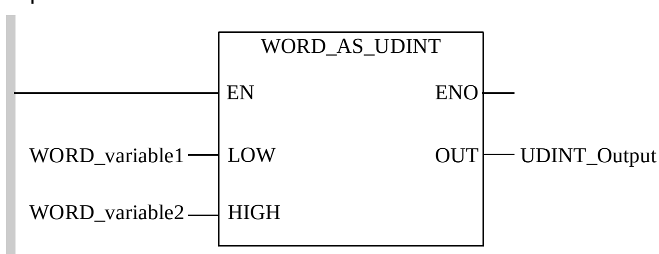
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD WORD_variable1
WORD_AS_UDINT WORD_variable2
ST UDINT_Output
```

Representation in ST

Representation:

```
UDINT_Output := WORD_AS_UDINT (WORD_variable1,
               WORD_variable2) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
WORD_variable1	WORD	least significant byte
WORD_variable2	WORD	most significant byte

Description of output parameters:

Parameter	Data type	Meaning
UDINT_Output	UDINT	Output value

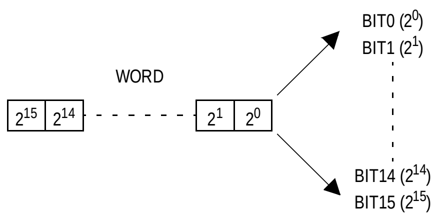
WORD_TO_BIT: Type conversion

146

Description

Function description

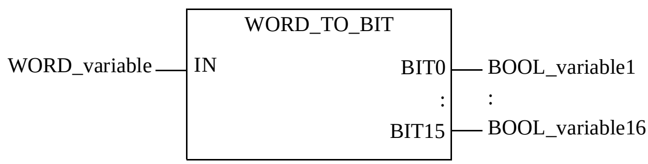
The procedure converts an input value of the `WORD` data type to 16 output values of the `BOOL` data type.
The individual bits of the word at the input are assigned to the outputs according to the output names.



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

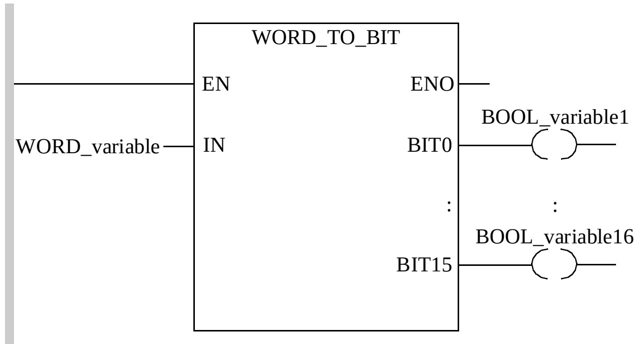
Representation in FBD

Representation:



Representation in LD

Representation:



Representation in IL

Representation:

```
LD WORD_variable
WORD_TO_BIT BOOL_variable1, BOOL_variable2,
             BOOL_variable3, BOOL_variable4, BOOL_variable5,
     BOOL_variable6, BOOL_variable7, BOOL_variable8,
     BOOL_variable9, BOOL_variable10, BOOL_variable11,
     BOOL_variable12, BOOL_variable13, BOOL_variable14,
     BOOL_variable15, BOOL_variable16
```

Representation in ST

Representation:

```
WORD_TO_BIT (WORD_variable, BOOL_variable1
    BOOL_variable2, BOOL_variable3, BOOL_variable4,
    BOOL_variable5, BOOL_variable6, BOOL_variable7,
    BOOL_variable8, BOOL_variable9, BOOL_variable10,
    BOOL_variable11, BOOL_variable12, BOOL_variable13,
    BOOL_variable14, BOOL_variable15, BOOL_variable16);
```

Parameter description

Description of the input parameters:

Parameter	Data type	Meaning
WORD_ variable	WORD	Input

Description of the output parameter:

Parameter	Data type	Meaning
BOOL_ variable1	BOOL	Output BIT0
:	:	:
BOOL_ variable16	BOOL	Output BIT15

WORD_TO_***: Type conversion

147

Description

Function description

The function converts an input value of the WORD data type to a BOOL, BYTE, DWORD, INT, DINT, UINT, UDINT, REAL or TIME data type.

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

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

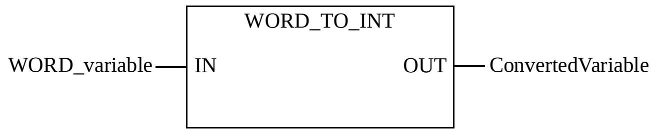
EN and ENO can be configured as additional parameters.
(The output ENO is not used for WORD_TO_REAL; it always has the value "1".)

Available functions

List of available functions:
- WORD_TO_BOOL
- WORD_TO_BYTE
- WORD_TO_DWORD
- WORD_TO_INT
- WORD_TO_DINT
- WORD_TO_UINT
- WORD_TO_UDINT
- WORD_TO_REAL
- WORD_TO_TIME

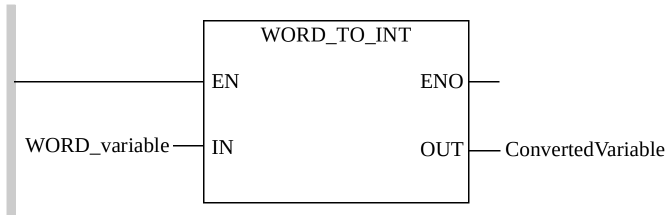
Representation in FBD

Representation of an Integer application:



Representation in LD

Representation of an Integer application:



Representation in IL

Representation of an Integer application:

```
LD WORD_variable
WORD_TO_INT
ST ConvertedVariable
```

Representation in ST

Representation of an Integer application:

```
ConvertedVariable := WORD_TO_INT (WORD_variable) ;
```

Parameter description

Description of input parameters:

Parameter	Data type	Meaning
WORD_ variable	WORD	Input value

Description of output parameters:

Parameter	Data type	Meaning
Converted Variable	BOOL, BYTE, DWORD, INT, DINT, UINT, UDINT, REAL, TIME	Output value

Runtime error

The system bit %S18 (See *Description of system bits %S15 to %S21, p. 448*) is set to 1 and the status is stored in %SW17 (See *Description of system words %SW12 to %SW18, p. 451*) if

- an unauthorized floating-point number is created during the conversion into the `REAL` data type.

***_TO_STRING: Conversion of a variable into a character string

148

Description

Function description

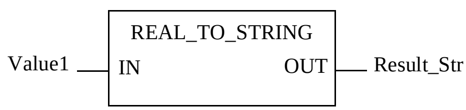
The `***_TO_STRING` function converts an INT, DINT or REAL variable into a character string.
The additional parameters `EN` and `ENO` can be configured.

Available functions

The available functions are as follows:
- `INT_TO_STRING`,
- `DINT_TO_STRING`,
- `REAL_TO_STRING`.

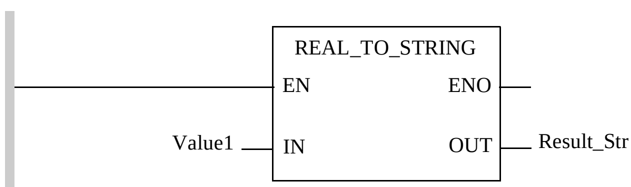
Representation in FBD

Representation applied to a real:



Representation in LD

Representation applied to a real:



Representation in IL

Representation applied to a real:

```
LD Value1
REAL_TO_STRING
ST Result_Str
```

Representation in ST

Representation applied to a real:

```
Result_Str:= REAL_TO_STRING(Value1);
```

Description of parameters

The following table describes the input parameters:

Parameter	Type	Comment
Value1	INT, DINT, REAL	Variable to be converted into character string format.

The following table describes the output parameters:

Parameter	Type	Comment
Result_Str	STRING	Result_Str is a character string whose length depends on the type of Value1: ● 5 figures plus the sign making 6 characters for one INT (example: '-00045'), ● 10 figures plus the sign, making 11 characters for one DINT (example: '-0000678911'), ● 14 characters for a REAL (example: '-3.1234560e+25').

Runtime errors

If, during the conversion of a value of REAL type, Value1 is not between -3.402824e+38 and -1.175494e-38 or +1.175494e-38 and +3.402824e+38, the bit **%S18** (See *Description of system bits %S15 to %S21, p. 448*) is set to 1 and the contents of the string Result_Str is not significant.

Appendices

Introduction

Overview

This section contains the appendicies.

What's in this Appendix?

The appendix contains the following chapters:

EFB Error Codes and Values

A

Overview

Introduction

The following tables show the error codes and error values created for the EFBs of the Base Library.

What's in this Chapter?

This chapter contains the following topics:

Tables of Error Codes for the Base Library

Introduction

The following tables show the error codes and error values created for the EFBs of the Base Library.

Date & Time

Table of error codes and errors values created for EFBs of the `Date & Time` family.

EFB name	Error code	ENO state in case of error	Error value in Dec	Error value in Hex	Error description
DIVTIME	E_DIVIDE_BY_ZERO	F	-30176	0x8A20	Divide by zero
DIVTIME	E_NEGATIVE_INPUT_ FOR_TIME_ OPERATION	F	-30177	0x8A1F	A negative value cannot be converted to data type `TIME`
DIVTIME	E_ARITHMETIC_ ERROR	F	-30170	0x8A26	Arithmetic error
DIVTIME	E_ERR_ARITHMETIC	F	-30003	0x8ACD	Arithmetic overflow (%S18 set)
DIVTIME	FP_ERROR	F	-	-	See table *Common Floating Point Errors, p. 444*
MULTIME	E_ERR_ARITHMETIC	F	-30003	0x8ACD	Arithmetic overflow (%S18 set)
MULTIME	E_ARITHMETIC_ ERROR_MUL_OV	F	-30172	0x8A24	Arithmetic error / Muliplication overflow
MULTIME	E_ARITHMETIC_ ERROR_ADD_OV	F	-30173	0x8A23	Arithmetic error / Addition overflow
MULTIME	E_ARITHMETIC_ ERROR_BIG_PAR	F	-30171	0x8A25	Arithmetic error / Parameter exceeds range
MULTIME	E_NEGATIVE_INPUT_ FOR_TIME_ OPERATION	F	-30177	0x8A1F	A negative value cannot be converted to data type `TIME`
MULTIME	FP_ERROR	F	-	-	See table *Common Floating Point Errors, p. 444*

Statistical

Table of error codes and errors values created for EFBs of the `Statistical` family.

EFB name	Error code	ENO state in case of error	Error value in Dec	Error value in Hex	Error description
AVE	E_INPUT_VALUE_ OUT_OF_RANGE	F	-30183	0x8A19	Input value is out of range
AVE	E_DIVIDE_BY_ZERO	F	-30176	0x8A20	Divide by zero
AVE	FP_ERROR	F	-	-	See table *Common Floating Point Errors, p. 444*
AVE	E_ARITHMETIC_ ERROR	F	-30170	0x8A26	Arithmetic error
AVE	E_FP_STATUS_ FAILED	F	-30150	0x8A3A	Illegal floating point operation
AVE	E_ARITHMETIC_ ERROR_MUL_OV	F	-30172	0x8A24	Arithmetic error / Muliplication overflow
AVE	E_ARITHMETIC_ ERROR_ADD_OV	F	-30173	0x8A23	Arithmetic error / Addition overflow
AVE	E_ARITHMETIC_ ERROR_BIG_PAR	F	-30171	0x8A25	Arithmetic error / Parameter exceeds range
AVE	E_ARITHMETIC_ ERROR_UNSIGN_OV	F	-30174	0x8A22	Arithmetic error / Unsigned overflow
MAX	FP_ERROR	F	-	-	See table *Common Floating Point Errors, p. 444*
MIN	FP_ERROR	F	-	-	See table *Common Floating Point Errors, p. 444*
MUX	E_SELECTOR_OUT_ OF_RANGE	F	-30175	0x8A21	Selector is out of range

Common Floating Point Errors

Introduction

The following table shows the commen error codes and error values created for floating point errors.

Common Floating Point Errors

Table of common floating point errors

Error codes	Error value in Dec	Error value in Hex	Error description
FP_ERROR	-30150	0x8A3A	Base value (not apearing as an error value)
E_FP_STATUS_FAILED_IE	-30151	0x8A39	Illegal floating point operation
E_FP_STATUS_FAILED_DE	-30152	0x8A38	Operand is denormalized - not a valid REAL number
E_FP_STATUS_FAILED_ZE	-30154	0x8A36	Illegal divide by zero
E_FP_STATUS_FAILED_ZE_IE	-30155	0x8A35	Illegal floating point operation / Divide by zero
E_FP_STATUS_FAILED_OE	-30158	0x8A32	Floating point overflow
E_FP_STATUS_FAILED_OE_IE	-30159	0x8A31	Illegal floating point operation / Overflow
E_FP_STATUS_FAILED_OE_ZE	-30162	0x8A2E	Floating point overflow / Divide by zero
E_FP_STATUS_FAILED_OE_ZE_IE	-30163	0x8A2D	Illegal floating point operation / Overflow / Divide by zero
E_FP_NOT_COMPARABLE	-30166	0x8A2A	Internal error

System objects

B

At a Glance

Subject of this Chapter

This chapter describes the system bits and words of Unity Pro language.

Note: The symbols, associated with each bit object or system word, mentioned in the descriptive tables of these objects, are not implemented as standard in the software, but can be entered using the data editor.
They are proposed in order to ensure the homogeneity of their names in the different applications.

What's in this Chapter?

This chapter contains the following topics:

System bit introduction

General

The Premium, Atrium and Quantum PLCs use %Si system bits which indicate the state of the PLC, or they can be used to control how it operates.
These bits can be tested in the user program to detect any functional development requiring a set processing procedure.
Some of these bits must be reset to their initial or normal state by the program. However, the system bits that are reset to their initial or normal state by the system must not be reset by the program or by the terminal.

Description of system bits %S9 to %S13

Detailed description

Description of system bits %S9 to %S13:

Bit Symbol	Function	Description	Initial state	Quantum	Premium Atrium
%S9 `OUTDIS`	Outputs set to the fallback position on all buses	Normally at 0, this bit is set to 1 by the program or the terminal: ● set to 1: sets the bit to 0 or maintains the current value depending on the chosen configuration (X bus, Fipio, AS-i, etc.), ● set to 0: outputs are updated normally. **Note:** The system bit acts directly on the physical outputs and not on the image bits of the outputs.	0	NO	YES
%S10 `IOERR`	Input/output fault	Normally at 1, this is set to 0 when an I/O fault on an in-rack module or device on Fipio is detected (e.g. non-compliant configuration, exchange fault, hardware fault, etc.). The %S10 bit is reset to 1 by the system as soon as the fault disappears.	1	YES	YES
%S11 `WDG`	Watchdog overflow	Normally at 0, this is set to 1 by the system as soon as the task execution time becomes greater than the maximum execution time (i.e. the watchdog) declared in the task properties.	0	YES	YES
%S12 `PLCRUNNING`	PLC in RUN	This bit is set to 1 by the system when the PLC is in RUN. It is set to 0 by the system as soon as the PLC is no longer in RUN (STOP, INIT, etc.).	0	YES	YES
%S13 `1RSTSCANRUN`	First cycle after switching to RUN	Normally set to 0, this is set to 1 by the system during the first cycle of the master task after the PLC is set to RUN.	-	YES	YES

Description of system bits %S15 to %S21

Detailed description

Description of system bits %S15 to %S21:

Bit **Symbol**	**Function**	**Description**	**Initial state**	**Quantum**	**Premium Atrium**
%S15 STRINGERROR	Character string fault	Normally set to 0, this is set to 1 when the destination zone for a character string transfer is not of sufficient size to receive this character string. The application stops in error state if the %S78 bit has been to set to 1. This bit must be reset to 0 by the application.	0	YES	YES
%S16 IOERRTSK	Task input/ output fault	Normally set to 1, this is set to 0 by the system when a fault occurs on an in-rack I/O module or a Fipio device configured in the task. This bit must be reset to 1 by the user.	1	YES	YES
%S17 CARRY	Rotate shift output	Normally at 0. During a rotate shift operation, this takes the state of the outgoing bit.	0	YES	YES

Bit **Symbol**	Function	Description	Initial state	Quantum	Premium Atrium
%S18 OVERFLOW	Overflow or arithmetic error	Normally set to 0, this is set to 1 in the event of a capacity overflow if there is: • a result greater than + 32 767 or less than - 32 768, in single length, • result greater than + 65 535, in unsigned integer, • a result greater than + 2 147 483 647 or less than - 2 147 483 648, in double length, • result greater than +4 294 967 296, in double length or unsigned integer, • real values outside limits, • division by 0, • the root of a negative number, • forcing to a non-existent step on a drum. • stacking up of an already full register, emptying of an already empty register. It must be tested by the user program after each operation where there is a risk of overflow, then reset to 0 by the user if there is indeed an overflow. When the %S18 bit switches to 1, the application stops in error state if the %S78 bit has been to set to 1.	0	YES	YES
%S19 OVERRUN	Task period overrun (periodical scanning)	Normally set to 0, this bit is set to 1 by the system in the event of a time period overrun (i.e. task execution time is greater than the period defined by the user in the configuration or programmed into the %SW word associated with the task). The user must reset this bit to 0. Each task manages its own %S19 bit.	0	YES	YES

Bit Symbol	Function	Description	Initial state	Quantum	Premium Atrium
%S20 `INDEXOVF`	Index overflow	Normally set to 0, this is set to 1 when the address of the indexed object becomes less than 0 or exceeds the number of objects declared in the configuration. In this case, it is as if the index were equal to 0. It must be tested by the user program after each operation where there is a risk of overflow, then reset to 0 if there is indeed an overflow. When the %S20 bit switches to 1, the application stops in error state if the %S78 bit has been to set to 1.	0	YES	YES
%S21 `1RSTTASKRUN`	First task cycle	Tested in a task (Mast, Fast, Aux0, Aux1, Aux2 Aux3), the bit %S21 indicates the first cycle of this task. %S21 is set to 1 at the start of the cycle and reset to zero at the end of the cycle. **Notes**: the bit %S21 does not have the same meaning in PL7 as in Unity Pro.	0	YES	YES

CAUTION

%S16 for Quantum PLCs

On Quantum, communication errors from modules (NOM, NOE, NWM, CRA, CRP) and MMS modules are not reported on bits %S10 and %S16.
It is entirely your responsibility to ensure that these system bits are used correctly

Failure to follow this precaution can result in injury or equipment damage.

Description of system words %SW12 to %SW18

Detailed description

Description of system words %SW12 to %SW18:

Word **Symbol**	**Function**	**Description**	**Initial state**	**Quantum**	**Premium Atrium**
%SW12 UTWPORTADDR	Uni-Telway terminal port address	Uni_Telway address of terminal port (in slave mode) as defined in the configuration and loaded into this word on cold start. **Note:** The modification of the value of this word is not taken into account by the system	-	NO	YES
%SW13 XWAYNETWADDR	Main address of the station	Indicates the following for the main network (Fipway or Ethway): ● the station number (least significant byte) from 0 to 127, ● the network number (most significant byte) from 0 to 63, (value of the micro-switches on the PCMCIA card).	254 (16#00FE)	NO	YES
%SW14 OSCOMMVERS	Commercial version of PLC processor	This word contains the commercial version of the PLC processor. **Example:** 16#0135 version: 01 issue number: 35	-	YES	YES
%SW15 OSCOMMPATCH	PLC processor patch version	This word contains the commercial version of the PLC processor patch. It is coded onto the least significant byte of the word. Coding: 0 = no patch, 1 = A, 2 = B... **Example:** 16#0003 corresponds to patch C.	-	YES	YES
%SW16 OSINTVERS	Firmware version of PLC processor	This word contains the Firmware version of the PLC processor. **Example:** 16#0143 version: 01 issue number: 43	-	YES	YES

Word `Symbol`	Function	Description	Initial state	Quantum	Premium Atrium
%SW17 `FLOATSTAT`	Error status on floating operation	On detection of an error in a floating arithmetic operation, bit %SW18 is set to 1 and %SW17 error status is updated according to the following coding: • %SW17.0 = Invalid operation / result is not a number • %SW17.1 = Non-standardized operand / result is acceptable • %SW17.2 = Division by 0 / result is infinity • %SW17.3 = Overflow / result is infinity • %SW17.4 = Underflow / result is 0 • %SW17.5 to 15 = not used This word is reset to 0 by the system on cold start, and also by the program for re-usage purposes.	0	YES	YES
%SD18 `100MSCOUNTER`	Absolute time counter	This double word is used to calculate duration. It is incremented every 1/10th of a second by the system (even when PLC is in STOP, it is no longer incremented if the PLC is powered down). It can be read and written by the user program or by the terminal.	0	YES	YES

Glossary

!

%I According to the IEC standard, `%I` indicates a discrete input-type language object.

%IW According to the IEC standard, `%IW` indicates an analog input -type language object.

%KW According to the IEC standard, `%KW` indicates a constant word-type language object.

%M According to the IEC standard, `%M` indicates a memory bit-type language object.

%MW According to the IEC standard, `%MW` indicates a memory word-type language object.

%Q According to the IEC standard, `%Q` indicates a discrete output-type language object.

%QW According to the IEC standard, `%QW` indicates an analog output-type language object.

A

ADDR_TYPE This predefined type is used as output for ADDR function. This type is ARRAY[0..5] OF Int. You can find it in the libset, in the same family than the EFs which use it.

ANL_IN `ANL_IN` is the abbreviation of Analog Input data type and is used when processing analog values. The `%IW` adresses for the configured analog input module, which were specified in the I/O component list, are automatically assigned data types and should therefore only be occupied with Unlocated Variables.

ANL_OUT

`ANL_OUT` is the abbreviation of Analog Output data type and is used when processing analog values. The `%MW` adresses for the configured analog input module, which were specified in the I/O component list, are automatically assigned data types and should therefore only be occupied with Unlocated Variables.

ANY

There is a hierarchy between the different types of data. In the DFB, it is sometimes possible to declare which variables can contain several types of values. Here, we use `ANY_xxx` types.
The following diagram shows the hierarchically-ordered structure:

- **ANY**
  - **ANY_ELEMENTARY**
    - **ANY_MAGNITUDE_OR_BIT**
      - **ANY_MAGNITUDE**
        - ANY_NUM
          - **ANY_REAL**
            - REAL
          - **ANY_INT**
            - DINT, INT, UDINT, UINT
        - TIME
      - **ANY_BIT**
        - DWORD, WORD, BYTE, BOOL
    - **ANY_STRING**
      - STRING
    - **ANY_DATE**
      - DATE_AND_TIME, DATE, TIME_OF_DAY
    - EBOOL
  - **ANY_DERIVED**
    - **ANY_ARRAY**
      - **ANY_ARRAY_ANY_EDT**
        - **ANY_ARRAY_ANY_MAGNITUDE**
          - **ANY_ARRAY_ANY_NUM**
            - **ANY_ARRAY_ANY_REAL**
              - **ANY_ARRAY_REAL**
            - **ANY_ARRAY_ANY_INT**
              - **ANY_ARRAY_DINT**
              - **ANY_ARRAY_INT**
              - **ANY_ARRAY_UDINT**
              - **ANNY_ARRAY_UINT**
          - **ANY_ARRAY_TIME**
        - **ANY_ARRAY_ANY_BIT**
          - **ANY_ARRAY_DWORD**
          - **ANY_ARRAY_WORD**
          - **ANY_ARRAY_BYTE**
          - **ANY_ARRAY_BOOL**
        - **ANY_ARRAY_ANY_STRING**
          - **ANY_ARRAY_STRING**
        - **ANY_ARRAY_ANY_DATE**
          - **ANY_ARRAY_DATE_AND_TIME**
          - **ANY_ARRAY_DATE**
          - **ANY_ARRAY_TIME_OF_DAY**
        - ANY_ARRAY_EBOOL
      - **ANY_ARRAY_ANY_DDT**
    - **ANY_STRUCTURE**
      - **ANY_DDT**
      - **ANY_IODDT**
      - **ANY_FFB**
        - **ANY_EFB**
        - **ANY_DFB**

ARRAY

An ARRAY is a table of elements of the same type.
The syntax is as follows: ARRAY [<terminals>] OF <Type>
Example:
ARRAY [1..2] OF BOOL is a one-dimensional table made up of two BOOL-type elements.
ARRAY [1..10, 1..20] OF INT is a two-dimensional table made up of 10x20 INT-type elements.

B

Base 10 literals

A literal value in base 10 is used to represent a decimal integer value. This value can be preceded by the signs "+" and "-". If the character "_" is employed in this literal value, it is not significant.
Example:
-12, 0, 123_456, +986

Base 16 Literals

An literal value in base 16 is used to represent an integer in hexadecimal. The base is determined by the number "16" and the sign "#". The signs "+" and "-" are not allowed. For greater clarity when reading, you can use the sign "_" between bits.
Example:
16#F_F or 16#FF (in decimal 255)
16#F_F or 16#FF (in decimal 224)

Base 2 Literals

A literal value in base 2 is used to represent a binary integer. The base is determined by the number "2" and the sign "#". The signs "+" and "-" are not allowed. For greater clarity when reading, you can use the sign "_" between bits.
Example:
2#1111_1111 or 2#11111111 (in decimal 255)
2#1110_0000 or 2#11100000 (in decimal 224)

Base 8 Literals

A literal value in base 8 is used to represent an octal integer. The base is determined by the number "8" and the sign "#". The signs "+" and "-" are not allowed. For greater clarity when reading, you can use the sign "_" between bits.
Example:
8#3_77 or 8#377 (in decimal 255)
8#34_0 or 8#340 (in decimal 224)

BCD

BCD is the abbreviation of Binary Coded Decimal format
BCD is used to represent decimal numbers between 0 and 9 using a group of four bits (half-byte).
In this format, the four bits used to code the decimal numbers have a range of unused combinations.
Example of BCD coding:

- the number `2450`
- is coded: `0010 0100 0101 0000`

BOOL

`BOOL` is the abbreviation of Boolean type. This is the elementary data item in computing. A `BOOL` type variable has a value of either: 0 (`FALSE`) or 1 (`TRUE`).
A `BOOL` type word extract bit, for example: `%MW10.4`.

BYTE

When 8 bits are put together, this is callad a `BYTE`. A `BYTE` is either entered in binary, or in base 8.
The `BYTE` type is coded in an 8 bit format, which, in hexadecimal, ranges from `16#00` to `16#FF`

D

DATE

The `DATE` type coded in BCD in 32 bit format contains the following information:

- the year coded in a 16-bit field,
- the month coded in an 8-bit field,
- the day coded in an 8-bit field.

The `DATE` type is entered as follows: **D#**<Year>**-**<Month>**-**<Day>
This table shows the lower/upper limits in each field:

Field	Limits	Comment
Year	[1990,2099]	Year
Month	[01,12]	The left 0 is always displayed, but can be omitted at the time of entry
Day	[01,31]	For the months 01\03\05\07\08\10\12
	[01,30]	For the months 04\06\09\11
	[01,29]	For the month 02 (leap years)
	[01,28]	For the month 02 (non leap years)

DATE_AND_ TIME

see `DT`

DBCD

Representation of a Double BCD-format double integer.
The Binary Coded Decimal (BCD) format is used to represent decimal numbers between 0 and 9 using a group of four bits.
In this format, the four bits used to code the decimal numbers have a range of unused combinations.
Example of DBCD coding:
- the number `78993016`
- is coded: `0111 1000 1001 1001 0011 0000 0001 0110`

DDT

DDT is the abbreviation of Derived Data Type.
A derived data type is a set of elements of the same type (`ARRAY`) or of various types (structure)

DFB

DFB is the abbrevation of Derived Function Block.
DFB types are function blocks that can be programmed by the user ST, IL, LD or FBD.
By using DFB types in an application, it is possible to:
- simplify the design and input of the program,
- increase the legibility of the program,
- facilitate the debugging of the program,
- reduce the volume of the generated code.

DINT

`DINT` is the abbrevation of Double Integer format (coded on 32 bits).
The lower and upper limits are as follows: -(2 to the power of 31) to (2 to the power of 31) - 1.
Example:
`-2147483648`, `2147483647`, `16#FFFFFFFF`.

DT

`DT` is the abbreviation of Date and Time.
The `DT` type coded in BCD in 64 bit format contains the following information:

- The year coded in a 16-bit field,
- the month coded in an 8-bit field,
- the day coded in an 8-bit field,
- the hour coded in a 8-bit field,
- the minutes coded in an 8-bit field,
- the seconds coded in an 8-bit field.

Note: The 8 least significant bits are unused.

The `DT` type is entered as follows:
DT#<Year>**-**<Month>**-**<Day>**-**<Hour>**:**<Minutes>**:**<Seconds>
This table shows the lower/upper limits in each field:

Field	Limits	Comment
Year	[1990,2099]	Year
Month	[01,12]	The left 0 is always displayed, but can be omitted at the time of entry
Day	[01,31]	For the months 01\03\05\07\08\10\12
	[01,30]	For the months 04\06\09\11
	[01,29]	For the month 02 (leap years)
	[01,28]	For the month 02 (non leap years)
Hour	[00,23]	The left 0 is always displayed, but can be omitted at the time of entry
Minute	[00,59]	The left 0 is always displayed, but can be omitted at the time of entry
Second	[00,59]	The left 0 is always displayed, but can be omitted at the time of entry

DWORD

`DWORD` is the abbreviation of Double Word.
The `DWORD` type is coded in 32 bit format.
This table shows the lower/upper limits of the bases which can be used:

Base	Lower limit	Upper limit
Hexadecimal	16#0	16#FFFFFFFF
Octal	8#0	8#37777777777
Binary	2#0	2#11111111111111111111111111111111

Representation examples:

Data content	Representation in one of the bases
00000000000010101101110011011110	16#ADCDE
00000000000000010000000000000000	8#200000
00000000000010101011110011011110	2#10101011110011011110

E

EBOOL

`EBOOL` is the abbrevation of Extended Boolean type. It can be used to manage rising or falling edges, as well as forcing.
An `EBOOL` type variable takes up one byte of memory.

EF

Is the abbreviation of Elementary Function.
This is a block which is used in a program, and which performs a predefined software function.
A function has no internal status information. Multiple invocations of the same function using the same input parameters always supply the same output values.
Details of the graphic form of the function invocation can be found in the "[Functional block (instance)] ". In contrast to the invocation of the function blocks, function invocations only have a single unnamed output, whose name is the same as the function. In FBD each invocation is denoted by a unique [number] via the graphic block, this number is automatically generated and can not be altered.
You position and set up these functions in your program in order to carry out your application.
You can also develop other functions using the SDKC development kit.

EFB

Is the abbreviation for Elementary Function Block.
This is a block which is used in a program, and which performs a predefined software function.
EFBs have internal statuses and parameters. Even where the inputs are identical, the output values may be different. For example, a counter has an output which indicates that the preselection value has been reached. This output is set to 1 when the current value is equal to the preselection value.

Elementary Function

see EF

EN

EN means **EN**able, this is an optional block input. When EN is activated, an ENO output is automatically drafted.
If EN = 0, the block is not activated, its internal program is not executed and ENO ist set to 0.
If EN = 1, the internal program of the block is executed, and ENO is set to 1 by the system. If an error occurs, ENO is set to 0.

ENO

ENO means **E**rror **NO**tification, this is the output associated to the optional input EN.
If ENO is set to 0 (caused by EN=0 or in case of an execution error),
- the outputs of function blocks remain in the status they were in for the last correct executed scanning cycle and
- the output(s) of functions and procedures are set to "0".

F

FBD

FBD is the abbreviation of Function Block Diagram.
FBD is a graphic programming language that operates as a logic diagram. In addition to the simple logic blocks (AND, OR, etc.), each function or function block of the program is represented using this graphic form. For each block, the inputs are located to the left and the outputs to the right. The outputs of the blocks can be linked to the inputs of other blocks to form complex expressions.

FFB

Collective term for EF (Elementary Function), EFB (Elementary Function Block) and DFB (Derived Function block)

Function

see EF

Function Block Diagram

see FBD

G

GRAY

Gray or "reflected binary" code is used to code a numerical value being developed into a chain of binary configurations that can be differentiated by the change in status of one and only one bit.
This code can be used, for example, to avoid the following random event: in pure binary, the change of the value 0111 to 1000 can produce random numbers between 0 and 1000, as the bits do not change value altogether simultaneously.
Equivalence between decimal, BCD and Gray:

Decimal	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001
Gray	0000	0001	0011	0010	0110	0111	0101	0100	1100	1101

I

IEC 61131-3

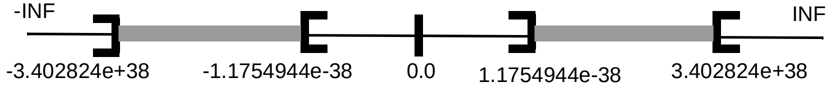
International standard: Programmable Logic Controls
Part 3: Programming languages.

IL

IL is the abbreviation of Instruction List.
This language is a series of basic instructions.
This language is very close to the assembly language used to program processors.
Each instruction is composed of an instruction code and an operand.

INF

Used to indicate that a number overruns the allowed limits.
For a number of Integers, the value ranges (shown in gray) are as follows:



When a calculation result is:

- less than -3.402824e+38, the symbol `-INF` (for -infinite) is displayed,
- greater than +3.402824e+38, the symbol `INF` (for +infinite) is displayed.

INT

INT is the abbreviation of single integer format (coded on 16 bits).
The lower and upper limits are as follows: -(2 to the power of 15) to (2 to the power of 15) - 1.
Example:
-32768, 32767, 2#1111110001001001, 16#9FA4.

Integer Literals

Integer literal are used to enter integer values in the decimal system. The values can have a preceding sign (+/-). Individual underlines (_) between numbers are not significant.
Example:
-12, 0, 123_456, +986

IODDT

IODDT is the abbreviation of Input/Output Derived Data Type.
The term IODDT designates a structured data type representing a module or a channel of a PLC module. Each application expert module possesses its own IODDTs.

K

Keyword

A keyword is a unique combination of characters used as a syntactical programming language element (See annex B definition of the IEC standard 61131-3. All the key words used in Unity Pro and of this standard are listed in annex C of the IEC standard 61131-3. These keywords cannot be used as identifiers in your program (names of variables, sections, DFB types, etc.)).

L

LD

LD is the abbreviation of Ladder Diagram.
LD is a programming language, representing the instructions to be carried out in the form of graphic diagrams very close to a schematic electrical diagram (contacts, coils, etc.).

Located variables

A located variable is a variable for which it is possible to know its position in the PLC memory. For example, the variable Water_pressure, is associated with%MW102. Water_pressure is said to be localized.

M

Multiple Token Operating mode of an SFC. In multitoken mode, the SFC may possess several active steps at the same time.

N

Naming conventions (Identifier) An identifier is a sequence of letters, numbers and underlines beginning with a letter or underline (e.g. name of a function block type, an instance, a variable or a section). Letters from national character sets (e.g: ö,ü, é, õ) can be used except in project and DFB names. Underlines are significant in identifiers; e.g. `A_BCD` and `AB_CD` are interpreted as different identifiers. Multiple leading underlines and consecutive underlines are invalid.
Identifiers cannot contain spaces. Not case sensitive; e.g. `ABCD` and `abcd` are interpreted as the same identifier.
According to IEC 61131-3 leading digits are not allowed in identifiers. Nevertheless, you can use them if you activate in dialog **Tools** → **Project settings** in tab **Language extensions** the ceck box **Leading digits**.
Identifiers cannot be keywords.

NAN Used to indicate that a result of an operation is not a number (NAN = Not A Number). Example: calculating the square root of a negative number.

> **Note:** The IEC 559 standard defines two classes of NAN: quiet `NAN` (`QNAN`) and signaling `NaN` (`SNaN`) `QNAN` is a `NAN` with the most significant fraction bit set and a `SNAN` is a `NAN` with the most significant fraction bit clear (Bit number 22). `QNAN`s are allowed to propagate through most arithmetic operations without signaling an exception. `SNAN` generally signal an invalid-operation exception whenever they appear as operands in arithmetic operations (See %SW17 and %S18).

Network

There are two meanings for Network.

- In LD:
  A network is a set of interconnected graphic elements. The scope of a network is local to the program organization unit (section) in which the network is located.
- With communication expert modules:
  A network is a group of stations which communicate among one another. The term network is also used to define a group of interconnected graphic elements. This group forms then a part of a program which may be composed of a group of networks.

P

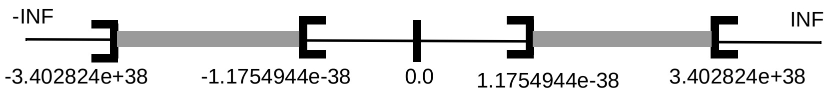
Procedure

Procedures are functions view technically. The only difference to elementary functions is that procedures can take up more than one output and they support data type `VAR_IN_OUT`. To the eye, procedures are no different than elementary functions.
Procedures are a supplement to IEC 61131-3.

R

REAL

Real type is a coded type in 32 bits.
The ranges of possible values are illustrated in gray in the following diagram:



When a calculation result is:

- between -1.175494e-38 and 1.175494e-38 it is considerd as a `DEN`,
- less than -3.402824e+38, the symbol `-INF` (for - infinite) is displayed,
- greater than +3.402824e+38, the symbol `INF` (for +infinite) is displayed,
- undefined (square root of a negative number), the symbol `NAN` or `NAN` is displayed.

> **Note:** The IEC 559 standard defines two classes of NAN: quiet `NAN` (`QNAN`) and signaling `NaN` (`SNaN`) `QNAN` is a `NAN` with the most significant fraction bit set and a `SNAN` is a `NAN` with the most significant fraction bit clear (Bit number 22). `QNAN`s are allowed to propagate through most arithmetic operations without signaling an exception. `SNAN` generally signal an invalid-operation exception whenever they appear as operands in arithmetic operations (See %SW17 and %S18).

> **Note:** when an operand is a `DEN` (Denormalized number) the result is not significant.

Real Literals

An literal real value is a number expressed in one or more decimals.
Example:
`-12.0`, `0.0`, `+0.456`, `3.14159_26`

Real Literals with Exponent

An Literal decimal value can be expressed using standard scientific notation. The representation is as follows: mantissa + exponential.
Example:
`-1.34E-12` or `-1.34e-12`
`1.0E+6` or `1.0e+6`
`1.234E6` or `1.234e6`

S

SFC SFC is the abbreviation of Sequential Function Chart.
SFC enables the operation of a sequential automation device to be represented graphically and in a structured manner. This graphic description of the sequential behavior of an automation device, and the various situations which result from it, is performed using simple graphic symbols.

Single Token Operating mode of an SFC chart for which only a single step can be active at any one time.

ST ST is the abbreviation of Structured Text language.
Structured Text language is an elaborated language close to computer programming languages. It enables you to structure series of instructions.

STRING A variable of the type `STRING` is an ASCII standard character string. A character string has a maximum length of 65534 characters.

T

TIME The type `TIME` expresses a duration in milliseconds. Coded in 32 bits, this type makes it possible to obtain periods from 0 to 2^{32}-1 milliseconds.
The units of type `TIME` are the following: the days (`d`), the hours (`h`), the minutes (`m`), the seconds (`s`) and the milliseconds (`ms`). A literal value of the type `TIME` is represented by a combination of previous types preceded by `T#`, `t#`, `TIME#` or `time#`.
Examples: `T#25h15m`, `t#14.7S`, `TIME#5d10h23m45s3ms`

Time literals The units of type `TIME` are the following: the days (`d`), the hours (`h`), the minutes (`m`), the seconds (`s`) and the milliseconds (`ms`). A literal value of the type `TIME` is represented by a combination of previous types preceded by `T#`, `t#`, `TIME#` or `time#`.
Examples: `T#25h15m`, `t#14.7S`, `TIME#5d10h23m45s3ms`

TIME_OF_DAY see `TOD`

TOD

`TOD` is the abbreviation of Time of Day.
The `TOD` type coded in BCD in 32 bit format contains the following information:

- the hour coded in a 8-bit field,
- the minutes coded in an 8-bit field,
- the seconds coded in an 8-bit field.

Note: The 8 least significant bits are unused.

The Time of Day type is entered as follows: **TOD#**<Hour>**:**<Minutes>**:**<Seconds>
This table shows the lower/upper limits in each field:

Field	Limits	Comment
Hour	[00,23]	The left 0 is always displayed, but can be omitted at the time of entry
Minute	[00,59]	The left 0 is always displayed, but can be omitted at the time of entry
Second	[00,59]	The left 0 is always displayed, but can be omitted at the time of entry

Example: `TOD#23:59:45`.

Token

An active step of an SFC is known as a token.

TOPO_ADDR_TYPE

This predefined type is used as output for READ_TOPO_ADDR function. This type is an ARRAY[0..4] OF Int. You can find it in the libset, in the same family than the EFs which use it.

U

UDINT

`UDINT` is the abbreviation of Unsigned Double Integer format (coded on 32 bits) unsigned. The lower and upper limits are as follows: 0 to (2 to the power of 32) - 1.
Example:
`0`, `4294967295`, `2#11111111111111111111111111111111`, `8#37777777777`, `16#FFFFFFFF`.

UINT

`UINT` is the abbreviation of Unsigned integer format (coded on 16 bits). The lower and upper limits are as follows: 0 to (2 to the power of 16) - 1.
Example:
`0`, `65535`, `2#1111111111111111`, `8#177777`, `16#FFFF`.

Unlocated variable

An unlocated variable is a variable for which it is impossible to know its position in the PLC memory. A variable which have no address assigned is said to be unlocated.

V

Variable

Memory entity of the type `BOOL`, `WORD`, `DWORD`, etc., whose contents can be modified by the program during execution.

W

WORD

The `WORD` type is coded in 16 bit format and is used to carry out processing on bit strings.
This table shows the lower/upper limits of the bases which can be used:

Base	Lower limit	Upper limit
Hexadecimal	16#0	16#FFFF
Octal	8#0	8#177777
Binary	2#0	2#1111111111111111

Representation examples

Data content	Representation in one of the bases
0000000011010011	16#D3
1010101010101010	8#125252
0000000011010011	2#11010011

Index

B

C

D

E

F

G

I

L

M

N

O

P

R

S

T

U

W

X