Unity Pro
Standard
Block Library

September 2004
# Table of Contents

About the Book .................................................. 15

**Part I** General information ................................. 17
   Introduction .................................................. 17

Chapter 1 Block types and their applications ................ 19
   Introduction .................................................. 19
   Block types .................................................. 20
   FFB Structure ................................................. 21
   EN and ENO ................................................... 24

Chapter 2 Availability of the blocks on different hardware platforms . . . 27
   Availability of the block on the various hardware platforms .......... 27

**Part II** Arrays .................................................. 39
   Introduction .................................................. 39

Chapter 3 ADD_***_***: Addition of a number to elements of a table or addition of two tables ...................... 41

Chapter 4 AND_***_***: Logical AND between tables and variables . . . 45

Chapter 5 COPY_***_***: Copy on tables ........................ 49

Chapter 6 DIV_***_***: Division of tables ...................... 53

Chapter 7 EQUAL_***: Comparison of two tables ................ 57

Chapter 8 FIND_EQ_***: First element of a table equal to a given value .................................................. 61
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Function</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 9</td>
<td>FIND_EQP_***</td>
<td>First element of a table equal to a value starting from a given rank</td>
<td>63</td>
</tr>
<tr>
<td>Chapter 10</td>
<td>FIND_GT_***</td>
<td>First element of a table greater than a given value</td>
<td>67</td>
</tr>
<tr>
<td>Chapter 11</td>
<td>FIND_LT_***</td>
<td>First element of a table less than a given value</td>
<td>69</td>
</tr>
<tr>
<td>Chapter 12</td>
<td>LENGTH_***</td>
<td>Length of a table</td>
<td>73</td>
</tr>
<tr>
<td>Chapter 13</td>
<td>MAX_***</td>
<td>Maximum value of table elements</td>
<td>75</td>
</tr>
<tr>
<td>Chapter 14</td>
<td>MIN_***</td>
<td>Minimum value of table elements</td>
<td>77</td>
</tr>
<tr>
<td>Chapter 15</td>
<td>MOD_<em><strong>_</strong></em></td>
<td>Remainder of division of tables</td>
<td>79</td>
</tr>
<tr>
<td>Chapter 16</td>
<td>MOVE_<em><strong>_</strong></em></td>
<td>Assignment to tables</td>
<td>83</td>
</tr>
<tr>
<td>Chapter 17</td>
<td>MOVE_<em><strong>_</strong></em></td>
<td>Table conversion</td>
<td>85</td>
</tr>
<tr>
<td>Chapter 18</td>
<td>MUL_<em><strong>_</strong></em></td>
<td>Multiplication of tables</td>
<td>87</td>
</tr>
<tr>
<td>Chapter 19</td>
<td>NOT_***</td>
<td>Logical negation of tables</td>
<td>91</td>
</tr>
<tr>
<td>Chapter 20</td>
<td>OCCUR_***</td>
<td>Occurrence of a value in a table</td>
<td>93</td>
</tr>
<tr>
<td>Chapter 21</td>
<td>OR_<em><strong>_</strong></em></td>
<td>Logical OR between tables and variables</td>
<td>95</td>
</tr>
<tr>
<td>Chapter 22</td>
<td>ROL_***</td>
<td>Rotate shift to left</td>
<td>99</td>
</tr>
<tr>
<td>Chapter 23</td>
<td>ROR_***</td>
<td>Rotate shift to right</td>
<td>101</td>
</tr>
<tr>
<td>Chapter 24</td>
<td>SORT_***</td>
<td>Ascending or descending sort</td>
<td>103</td>
</tr>
</tbody>
</table>
Chapter 25  SUB_***_***: Subtraction from tables .......................... 105
Chapter 26  SUM_***: Sum of table elements ............................... 109
Chapter 27  SWAP_***: Permutation of the bytes of a table ............. 111
Chapter 28  XOR_***_***: Exclusive OR between tables ................. 113

Part III  CLC_INT .................................................................. 117
          Introduction .................................................................. 117
Chapter 29  Introduction to integer regulation functions ............... 119
          At a Glance ................................................................... 119
          General Introduction ...................................................... 120
          Principal of the regulation loop ...................................... 121
          Development methodology for a regulation application ....... 122
          Programming a regulation function .................................. 123
          Behavior of functions in operating modes ....................... 124
Chapter 30  PID_INT: PID controller ........................................... 125
            Description ................................................................. 125
            Function description ................................................... 126
            Description of Derived Data ........................................ 130
Chapter 31  PWM_INT: Pulse width modulation of a numerical value . 133
Chapter 32  SERVO_INT: Servo drive function ............................... 137
Part IV  Comparison .............................................................. 143
        Introduction .......................................................... 143
Chapter 33  EQ: Equal to ..................................................... 145
Chapter 34  GE: Greater than or equal to ............................... 147
Chapter 35  GT: Greater than ................................................ 151
Chapter 36  LE: Less than or equal to .................................... 155
Chapter 37  LT: Less than ..................................................... 159
Chapter 38  NE: Not equal to ................................................ 163

Part V  Date & Time ............................................................. 165
        Introduction ............................................................. 165
Chapter 39  ADD_***_TIME: Addition of a duration to a date ....... 167
Chapter 40  DIVTIME: Division ............................................. 169
Chapter 41  MULTIME: Multiplication .................................... 171
Chapter 42  SUB_***_***: Calculates the time difference between
two dates or times .......................................................... 173
Chapter 43  SUB_***_TIME: Subtraction of a duration from a date .... 175
Part VI Logic .................................................................................. 177
Introduction .................................................................................. 177
Chapter 44 AND: AND function .................................................... 179
Chapter 45 F_TRIG: Falling edge detection .................................... 181
Chapter 46 FE: Detection of Falling Edge ....................................... 183
Chapter 47 NOT: Negation ............................................................... 185
Chapter 48 OR: OR function ............................................................. 187
Chapter 49 R_TRIG: Rising edge detection ...................................... 189
Chapter 50 RE: Detection of Rising Edge ........................................ 191
Chapter 51 RESET: Setting of a bit to 0 ........................................... 193
Chapter 52 ROL: Rotate left ............................................................. 195
Chapter 53 ROR: Rotate right .......................................................... 197
Chapter 54 RS: Bistable function block, reset dominant ................. 199
Chapter 55 SET: Setting of a bit to 1 ............................................... 201
Chapter 56 SHL: Shift left ............................................................... 203
Chapter 57 SHR: Shift right ............................................................. 205
Chapter 58 SR: Bistable function block, set dominant ................. 207
| Chapter 59 | TRIGGER: Detection of all edges | 209 |
| Chapter 60 | XOR: Exclusive OR function | 211 |
| Part VII | Mathematics | 213 |
| Introduction | 213 |
| Chapter 61 | ABS: Absolute value computation | 215 |
| Chapter 62 | ACOS: Arc cosine | 217 |
| Chapter 63 | ADD: Addition | 219 |
| Chapter 64 | ADD_TIME: Addition | 221 |
| Chapter 65 | ASIN: Arc sine | 223 |
| Chapter 66 | ATAN: Arc tangent | 225 |
| Chapter 67 | COS: Cosine | 227 |
| Chapter 68 | DEC: Decrementation of a variable | 229 |
| Chapter 69 | DIV: Division | 231 |
| Chapter 70 | DIVMOD: Division and Modulo | 233 |
| Chapter 71 | EXP: Natural exponential | 235 |
| Chapter 72 | EXPT_REAL_**: Exponentiation of one value by another value | 237 |
| Chapter 73 | INC: Incrementation of a variable | 239 |
Chapter 90  MIN: Minimum value function  281
Chapter 91  MUX: Multiplexer  283
Chapter 92  SEL: Binary selection  287

Part IX  Strings  289
        Introduction  289
Chapter 93  CONCAT_STR: Concatenation of two character strings  291
Chapter 94  DELETE_INT: Deletion of a sub-string of characters  293
Chapter 95  EQUAL_STR: Comparison of two character strings  295
Chapter 96  FIND_INT: Finding a sub-string of characters  297
Chapter 97  INSERT_INT: Insertion of a sub-string of characters  299
Chapter 98  LEFT_INT: Extraction of characters to the left  303
Chapter 99  LEN_INT: Length of character string  305
Chapter 100  MID_INT: Extraction of a sub-string of characters  307
Chapter 101  REPLACE_INT: Replacement of a sub-string of characters  309
Chapter 102  RIGHT_INT: Extraction of a character string to the right  313
Part X  Timer & Counter ................................................. 315
  Introduction ......................................................... 315
Chapter 103  CTD, CTD_**: Down counter ....................... 317
Chapter 104  CTU, CTU_**: Up counter .......................... 321
Chapter 105  CTUD, CTUD_**: Up/Down counter ......... 325
Chapter 106  TOF: Off delay ....................................... 329
Chapter 107  TON: On delay ....................................... 331
Chapter 108  TP: Pulse ............................................. 333

Part XI  Type to type .................................................. 335
  Introduction ......................................................... 335
Chapter 109  BCD_TO_INT: Conversion of a BCD integer
               into pure binary ....................................... 339
Chapter 110  BIT_TO_BYTE: Type conversion .................... 341
Chapter 111  BIT_TO WORD: Type conversion ..................... 345
Chapter 112  BOOL_TO_***: Type conversion ...................... 347
Chapter 113  BYTE_AS_WORD: Type conversion .............. 349
Chapter 114  BYTE_TO_BIT: Type conversion ................. 351
Chapter 115  BYTE_TO_***: Type conversion ................. 355
Chapter 116  DATE_TO_STRING: Conversion of a variable
               in DATE format into a character string ............ 359
Chapter 117 DBCD_TO_***: Conversion of a double BCD integer into binary ........................................ 361

Chapter 118 DEG_TO_RAD: Conversion of degrees to radians ........ 363

Chapter 119 DINT_AS_WORD: Type conversion .......................... 365

Chapter 120 DINT_TO_***: Type conversion ......................... 367

Chapter 121 DINT_TO_DBCD: Conversion of a double binary coded integer into a double Binary Coded Decimal integer .... 371

Chapter 122 DT_TO_STRING: Conversion of a variable in DT format into a character string ....................................................... 373

Chapter 123 DWORD_TO_***: Type conversion .......................... 375

Chapter 124 GRAY_TO_INT: Conversion of an integer in Gray code into a binary coded integer .......................... 377

Chapter 125 INT_AS_DINT: Concatenation of two integers to form a double integer .................................................. 379

Chapter 126 INT_TO_***: Type conversion .............................. 381

Chapter 127 INT_TO_BCD: Conversion of a binary coded integer into a Binary Coded Decimal integer .................. 385

Chapter 128 INT_TO_DBCD: Conversion of a binary coded integer into a double Binary Coded Decimal integer ........ 387

Chapter 129 RAD_TO_DEG: Conversion of radians to degrees .... 389
Chapter 130  REAL_AS_WORD: Type conversion ............................... 391
Chapter 131  REAL_TO_***: Type conversion ............................... 393
Chapter 132  REAL_TRUNC_***: Type conversion .............................. 397
Chapter 133  to a number of the INT, DINT or REAL type ............... 399
Chapter 134  TYPE_AS_WORD: Type conversion ............................... 401
Chapter 135  TIME_TO_***: Type conversion ............................... 403
Chapter 136  TIME_TO_STRING: Conversion of a variable
in TIME format into a character string ............................... 405
Chapter 137  TOD_TO_STRING: Conversion of a variable
in TOD format into a character string ............................... 407
Chapter 138  UDINT_AS_WORD: Type conversion ............................... 409
Chapter 139  UDINT_TO_***: Type conversion ............................... 411
Chapter 140  UINT_TO_***: Type conversion ............................... 415
Chapter 141  WORD_AS_BYTE: Type conversion ............................... 419
Chapter 142  WORD_AS_DINT: Type conversion ............................... 421
Chapter 143  WORD_AS_REAL: Type conversion ............................... 423
Chapter 144  WORD_AS_TIME: Type conversion ............................... 425
Chapter 145  WORD_AS_UDINT: Type conversion ............................... 427
# About the Book

## At a Glance

<table>
<thead>
<tr>
<th>Document Scope</th>
<th>This document describes the functions and function blocks of the Standard library. This document is valid for Unity Pro Version 2.0.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validity Note</td>
<td>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.</td>
</tr>
</tbody>
</table>
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.

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of Schneider Electric.

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

Introduction

Overview

This section contains general information about the Standard library.

What's in this Part?

This part contains the following chapters:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Chapter Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Block types and their applications</td>
<td>19</td>
</tr>
<tr>
<td>2</td>
<td>Availability of the blocks on different hardware platforms</td>
<td>27</td>
</tr>
</tbody>
</table>
Introduction

Overview
This chapter describes the different block types and their applications.

What's in this Chapter?
This chapter contains the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block types</td>
<td>20</td>
</tr>
<tr>
<td>FFB Structure</td>
<td>21</td>
</tr>
<tr>
<td>EN and ENO</td>
<td>24</td>
</tr>
</tbody>
</table>
Block types and their applications

**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:

```
MY_TON (EN:=ENABLE, IN:=EXAMP, PT:=TIME1, ENO=>ERROR, Q=>OUT, ET=>TIME2);
```

Call of a function block in the ST programming language:

```
MY_TON (EN:=ENABLE, IN:=EXAMP, PT:=TIME1, ENO=>ERROR, Q=>OUT, ET=>TIME2);
```

**Operation**
The operation determines which function is to be executed with the FFB, e.g. shift register, conversion operations.
## Block types and their applications

<table>
<thead>
<tr>
<th><strong>Operand</strong></th>
<th>The operand specifies what the operation is to be executed with. With FFBs, this consists of formal and actual parameters.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Formal/actual parameters</strong></td>
<td>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.</td>
</tr>
</tbody>
</table>

### 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 a 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:**

```
MY_EXAMP1

+-------+-------+
| Input1| IN1    | Output1 |
| Input2| IN2    | Output2 |
| Comb_IN_OUT | IO1  | Comb_IN_OUT |
```

**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 variables.**
- 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:
    
    ![FunctionBlock_1](image1)
    
    ![FunctionBlock_2](image2)
    
    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:
    
    ![FunctionBlock_1](image3)
    
    ![FunctionBlock_2](image4)
    
    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:

<table>
<thead>
<tr>
<th>Function/Procedure_1</th>
<th>Function/Procedure_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
<td>EN</td>
</tr>
<tr>
<td>ENO</td>
<td>ENO</td>
</tr>
<tr>
<td>IN1</td>
<td>IN1</td>
</tr>
<tr>
<td>OUT</td>
<td>OUT</td>
</tr>
<tr>
<td>IN2</td>
<td>IN2</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Function/Procedure_1</th>
<th>Function/Procedure_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
<td>EN</td>
</tr>
<tr>
<td>ENO</td>
<td>ENO</td>
</tr>
<tr>
<td>IN1</td>
<td>IN1</td>
</tr>
<tr>
<td>OUT</td>
<td>OUT</td>
</tr>
<tr>
<td>OUT1</td>
<td>IN2</td>
</tr>
</tbody>
</table>

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.
Block types and their applications

**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

### 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:**

<table>
<thead>
<tr>
<th>Block name</th>
<th>Block type</th>
<th>IEC 61131-3</th>
<th>Premium</th>
<th>Quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD_<em><strong>_</strong></em></td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>AND_<em><strong>_</strong></em></td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>COPY_<em><strong>_</strong></em></td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>DIV_<em><strong>_</strong></em></td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>EQUAL_***</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>FIND_EQ_***</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>FIND_EQP_***</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>FIND_GT_***</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>FIND_LT_***</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>LENGHT_***</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>MAX_***</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>MIN_***</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>MOD_<em><strong>_</strong></em></td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>MOVE_<em><strong>_</strong></em> (direct assignment)</td>
<td>Procedure</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>MOVE_<em><strong>_</strong></em> (conversion)</td>
<td>Procedure</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>MUL_<em><strong>_</strong></em></td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>NOT_***</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>OCCUR_***</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>OR_<em><strong>_</strong></em></td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>ROL_***</td>
<td>Procedure</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>ROR_***</td>
<td>Procedure</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SORT_***</td>
<td>Procedure</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SUB_<em><strong>_</strong></em></td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SUM_***</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SWAP_***</td>
<td>Procedure</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>XOR_<em><strong>_</strong></em></td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

**Legend:**

- Yes
- No
### CLC_INT

**Availability of the blocks:**

<table>
<thead>
<tr>
<th>Block name</th>
<th>Block type</th>
<th>defined in IEC 61131-3</th>
<th>Premium</th>
<th>Quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>PID_INT</td>
<td>Procedure</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>PWM_INT</td>
<td>Procedure</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SERVO_INT</td>
<td>Procedure</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

**Legend:**

+  Yes
-  No

---

### Comparison

**Availability of the blocks:**

<table>
<thead>
<tr>
<th>Block name</th>
<th>Block type</th>
<th>defined in IEC 61131-3</th>
<th>Premium</th>
<th>Quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQ</td>
<td>EF</td>
<td>+</td>
<td>+*</td>
<td>+</td>
</tr>
<tr>
<td>GE</td>
<td>EF</td>
<td>+</td>
<td>+*</td>
<td>+</td>
</tr>
<tr>
<td>GT</td>
<td>EF</td>
<td>+</td>
<td>+*</td>
<td>+</td>
</tr>
<tr>
<td>LE</td>
<td>EF</td>
<td>+</td>
<td>+*</td>
<td>+</td>
</tr>
<tr>
<td>LT</td>
<td>EF</td>
<td>+</td>
<td>+*</td>
<td>+</td>
</tr>
<tr>
<td>NE</td>
<td>EF</td>
<td>+</td>
<td>+*</td>
<td>+</td>
</tr>
</tbody>
</table>

**Legend:**

+  Yes
-  No

**Premium:** +* The data types UINT and UDINT are only available on Premium TSX P 57 5**.
## Date & Time

### Availability of the blocks:

<table>
<thead>
<tr>
<th>Block name</th>
<th>Block type</th>
<th>defined in IEC 61131-3</th>
<th>Premium</th>
<th>Quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD_***_TIME</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>DIVTIME</td>
<td>EF</td>
<td>+</td>
<td>*</td>
<td>+</td>
</tr>
<tr>
<td>MULTIME</td>
<td>EF</td>
<td>+</td>
<td>*</td>
<td>+</td>
</tr>
<tr>
<td>SUB_<em><strong>_</strong></em></td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SUB_***_TIME</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

**Legend:**

- +: Yes
- -: No

**Premium: +**

The data types UINT and UDINT are only available on Premium TSX P 57 5+. 

- Premium: No
## Logic

### Availability of the blocks:

<table>
<thead>
<tr>
<th>Block name</th>
<th>Block type</th>
<th>defined in IEC 61131-3</th>
<th>Premium</th>
<th>Quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>F_TRIG</td>
<td>EFB</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>FE</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>NOT</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>OR</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>R_TRIG</td>
<td>EFB</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>RE</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>RESET</td>
<td>Procedure</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>ROL</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>ROR</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>RS</td>
<td>EFB</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SET</td>
<td>Procedure</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SHL</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SHR</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SR</td>
<td>EFB</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>TRIGGER</td>
<td>EFB</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>XOR</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

**Legend:**
- + Yes
- - No
## Mathematics

### Availability of the blocks:

<table>
<thead>
<tr>
<th>Block name</th>
<th>Block type</th>
<th>defined in IEC 61131-3</th>
<th>Premium</th>
<th>Quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>ACOS</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>ADD</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>ADD_TIME</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>ASIN</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>ATAN</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>COS</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>DEC</td>
<td>Procedure</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>DIV</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>DIVMOD</td>
<td>Procedure</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>EXP</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>EXPT_REAL</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>INC</td>
<td>Procedure</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>LN</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>LOG</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>MOD</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>MOVE</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>MUL</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>NEG</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SIGN</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SIN</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SQRT</td>
<td>EF</td>
<td>DINT: - INT: - REAL: +</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SUB</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SUB_TIME</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>TAN</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

**Legend:**

- **+Y** e s
### Availability of the block

 statistical block type defined in block type defined in block type defined in IEC 61131-3 Premium Quantum

<table>
<thead>
<tr>
<th>Block name</th>
<th>Block type</th>
<th>defined in IEC 61131-3</th>
<th>Premium</th>
<th>Quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Premium: +</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The data types <strong>UINT</strong> and <strong>UDINT</strong> are only available on Premium TSX P 57 5**.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Statistical

Availability of the blocks:

<table>
<thead>
<tr>
<th>Block name</th>
<th>Block type</th>
<th>defined in IEC 61131-3</th>
<th>Premium</th>
<th>Quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVE</td>
<td>EF</td>
<td></td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>LIMIT</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>LIMIT_IND</td>
<td>Procedure</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>MAX</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>MIN</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>MUX</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SEL</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

Legend:

+ Yes

Premium: + The data types **UINT** and **UDINT** are only available on Premium TSX P 57 5**.

- No
### Strings

#### Availability of the blocks:

<table>
<thead>
<tr>
<th>Block name</th>
<th>Block type</th>
<th>defined in IEC 61131-3</th>
<th>Premium</th>
<th>Quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONCAT_STR</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>DELETE_INT</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>EQUAL_STR</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>FIND_INT</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>INSERT_INT</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>LEFT_INT</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>LEN_INT</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>MID_INT</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>REPLACE_INT</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>RIGHT_INT</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

**Legend:**

<table>
<thead>
<tr>
<th>+</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>No</td>
</tr>
</tbody>
</table>
### Availability of the blocks:

<table>
<thead>
<tr>
<th>Block name</th>
<th>Block type</th>
<th>defined in IEC 61131-3</th>
<th>Premium</th>
<th>Quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTD</td>
<td>EFB</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>CTD_***</td>
<td>EFB</td>
<td>-</td>
<td>*</td>
<td>+</td>
</tr>
<tr>
<td>CTU</td>
<td>EFB</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>CTU_***</td>
<td>EFB</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>CTUD</td>
<td>EFB</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>CTUD_***</td>
<td>EFB</td>
<td>-</td>
<td>*</td>
<td>+</td>
</tr>
<tr>
<td>TOF</td>
<td>EFB</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>TON</td>
<td>EFB</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>TP</td>
<td>EFB</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

**Legend:**

+ Y e s

- No

**Premium: +**

The data types UINT and UDINT are only available on Premium TSX P 57 5**.
### Availability of the blocks:

<table>
<thead>
<tr>
<th>Block name</th>
<th>Block type</th>
<th>defined in IEC 61131-3</th>
<th>Premium</th>
<th>Quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCD_TO_INT</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>BIT_TO_BYTE</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>BIT_TO_WORD</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>BOOL_TO_***</td>
<td>EF</td>
<td>+</td>
<td>±</td>
<td>+</td>
</tr>
<tr>
<td>BYTE_AS_WORD</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>BYTE_TO_BIT</td>
<td>Procedure</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>BYTE_TO_***</td>
<td>EF</td>
<td>+</td>
<td>±</td>
<td>+</td>
</tr>
<tr>
<td>DATE_TO_STRING</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>DBCD_TO_***</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>DEG_TO_RAD</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>DINT_AS_WORD</td>
<td>Procedure</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>DINT_TO_***</td>
<td>EF</td>
<td>+</td>
<td>±</td>
<td>+</td>
</tr>
<tr>
<td>DINT_TO_DBCD</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>DINT_TO_STRING</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>DT_TO_STRING</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>DWORD_TO_***</td>
<td>EF</td>
<td>+</td>
<td>±</td>
<td>+</td>
</tr>
<tr>
<td>GRAY_TO_INT</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>INT_AS_DINT</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>INT_TO_***</td>
<td>EF</td>
<td>+</td>
<td>±</td>
<td>+</td>
</tr>
<tr>
<td>INT_TO_BCD</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>INT_TO_DBCD</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>INT_TO_STRING</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>RAD_TO_DEG</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>REAL_AS_WORD</td>
<td>Procedure</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>REAL_TO_***</td>
<td>EF</td>
<td>+</td>
<td>±</td>
<td>+</td>
</tr>
<tr>
<td>REAL_TO_STRING</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>REAL_TRUNC_***</td>
<td>EF</td>
<td>+</td>
<td>±</td>
<td>+</td>
</tr>
<tr>
<td>STRING_TO_***</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>TIME_AS_WORD</td>
<td>Procedure</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>TIME_TO_***</td>
<td>EF</td>
<td>+</td>
<td>±</td>
<td>+</td>
</tr>
<tr>
<td>TIME_TO_STRING</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>
## Availability of the block

<table>
<thead>
<tr>
<th>Block name</th>
<th>Block type</th>
<th>defined in IEC 61131-3</th>
<th>Premium</th>
<th>Quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOD_TO_STRING</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>UDINT_AS_WORD</td>
<td>Procedure</td>
<td>-</td>
<td>+*</td>
<td>+</td>
</tr>
<tr>
<td>UDINT_TO_***</td>
<td>EF</td>
<td>+</td>
<td>+*</td>
<td>+</td>
</tr>
<tr>
<td>UINT_TO_***</td>
<td>EF</td>
<td>+</td>
<td>+*</td>
<td>+</td>
</tr>
<tr>
<td>WORD_AS_BYTE</td>
<td>Procedure</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>WORD_AS_DINT</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>WORD_AS_REAL</td>
<td>Procedure</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>WORD_AS_TIME</td>
<td>EF</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>WORD_AS_UDINT</td>
<td>EF</td>
<td>-</td>
<td>+*</td>
<td>+</td>
</tr>
<tr>
<td>WORD_TO_BIT</td>
<td>Procedure</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>WORD_TO_***</td>
<td>EF</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

Legend:

- + Yes
- - No

Premium: + The data types UINT and UDINT are only available on Premium TSX P 57 5**.
Availability of the block
Introduction

Overview

This section describes the elementary functions and elementary function blocks of the \textit{Arrays} family.
This part contains the following chapters:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Chapter Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>ADD_<em><strong>_</strong></em>: Addition of a number to elements of a table or addition of two tables</td>
<td>41</td>
</tr>
<tr>
<td>4</td>
<td>AND_<em><strong>_</strong></em>: Logical AND between tables and variables</td>
<td>45</td>
</tr>
<tr>
<td>5</td>
<td>COPY_<em><strong>_</strong></em>: Copy on tables</td>
<td>49</td>
</tr>
<tr>
<td>6</td>
<td>DIV_<em><strong>_</strong></em>: Division of tables</td>
<td>53</td>
</tr>
<tr>
<td>7</td>
<td>EQUAL_***: Comparison of two tables</td>
<td>57</td>
</tr>
<tr>
<td>8</td>
<td>FIND_EQ_***: First element of a table equal to a given value</td>
<td>61</td>
</tr>
<tr>
<td>9</td>
<td>FIND_EQP_***: First element of a table equal to a value starting from a given rank</td>
<td>63</td>
</tr>
<tr>
<td>10</td>
<td>FIND_GT_***: First element of a table greater than a given value</td>
<td>67</td>
</tr>
<tr>
<td>11</td>
<td>FIND_LT_***: First element of a table less than a given value</td>
<td>69</td>
</tr>
<tr>
<td>12</td>
<td>LENGTH_***: Length of a table</td>
<td>73</td>
</tr>
<tr>
<td>13</td>
<td>MAX_***: Maximum value of table elements</td>
<td>75</td>
</tr>
<tr>
<td>14</td>
<td>MIN_***: Minimum value of table elements</td>
<td>77</td>
</tr>
<tr>
<td>15</td>
<td>MOD_<em><strong>_</strong></em>: Remainder of division of tables</td>
<td>79</td>
</tr>
<tr>
<td>16</td>
<td>MOVE_<em><strong>_</strong></em>: Assignment to tables</td>
<td>83</td>
</tr>
<tr>
<td>17</td>
<td>MOVE_<em><strong>_</strong></em>: Table conversion</td>
<td>85</td>
</tr>
<tr>
<td>18</td>
<td>MUL_<em><strong>_</strong></em>: Multiplication of tables</td>
<td>87</td>
</tr>
<tr>
<td>19</td>
<td>NOT_***: Logical negation of tables</td>
<td>91</td>
</tr>
<tr>
<td>20</td>
<td>OCCUR_***: Occurrence of a value in a table</td>
<td>93</td>
</tr>
<tr>
<td>21</td>
<td>OR_<em><strong>_</strong></em>: Logical OR between tables and variables</td>
<td>95</td>
</tr>
<tr>
<td>22</td>
<td>ROL_***: Rotate shift to left</td>
<td>99</td>
</tr>
<tr>
<td>23</td>
<td>ROR_***: Rotate shift to right</td>
<td>101</td>
</tr>
<tr>
<td>24</td>
<td>SORT_***: Ascending or descending sort</td>
<td>103</td>
</tr>
<tr>
<td>25</td>
<td>SUB_<em><strong>_</strong></em>: Subtraction from tables</td>
<td>105</td>
</tr>
<tr>
<td>26</td>
<td>SUM_***: Sum of table elements</td>
<td>109</td>
</tr>
<tr>
<td>27</td>
<td>SWAP_***: Permutation of the bytes of a table</td>
<td>111</td>
</tr>
<tr>
<td>28</td>
<td>XOR_<em><strong>_</strong></em>: Exclusive OR between tables</td>
<td>113</td>
</tr>
</tbody>
</table>
**ADD_***_***: Addition of a number to elements of a table or addition of two tables**

### 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:

```
<table>
<thead>
<tr>
<th>Input_IN1</th>
<th>ADD_ARINT_INT</th>
<th>IN1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input_IN2</td>
<td></td>
<td>IN2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Array1</td>
</tr>
</tbody>
</table>
```
Representation applied to the sum of an integer and an integer table:

**Representation in LD**

```
ADD_ARINT_INT
EN  ENO
Input_IN1  IN1  OUT  Array1
Input_IN2  IN2
```

**Representation in IL**

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

**Representation in ST**

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

**Description of parameters**

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input_IN1</td>
<td>ARRAY [n..m] OF INT, ARRAY [n..m] OF DINT</td>
<td>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.</td>
</tr>
<tr>
<td>Input_IN2</td>
<td>INT, DINT, ARRAY [n..m] OF INT, ARRAY [n..m] OF DINT</td>
<td>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.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
</table>
| 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**

**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_ARBOOL` (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_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).

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

**FBD**

```
AND_ARWORD_WORD
IN1
IN2
OUT
```

**LD**

```
AND_ARWORD_WORD
EN
IN1
IN2
OUT
```

**IL**

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

**ST**

```
Array1:= AND_ARWORD_WORD(Input_IN1,Input_IN2);
```
The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input_IN1</td>
<td>ARRAY [n..m] OF EBOOL, WORD, ARRAY [n..m] OF WORD, DWORD, ARRAY [n..m] OF DWORD, ARRAY [n..m] OF INT, Array [n..m] OF DINT</td>
<td>n and m maximum and minimum limits.</td>
</tr>
<tr>
<td>Input_IN2</td>
<td>ARRAY [n..m] OF EBOOL, WORD, ARRAY [n..m] OF WORD, DWORD, ARRAY [n..m] OF DWORD, ARRAY [n..m] OF INT, Array [n..m] OF DINT</td>
<td>n and m are positive or negative integers or nil.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
</table>
| 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

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:

```
COPY_ARINT_AREBOOL

Array1   IN    OUT
Begin_Row P1
Element_number N
Destination_Row P2
```

Result_Array
Representation in LD

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

```
COPY_ARINT_AREBOOL
EN  ENO
Array1 IN  OUT
Begin_Row P1
Element_number N
Destination_Row p2
```

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);
```
The following table describes the input parameters:

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

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result_Array</td>
<td>ARRAY [n..m] OF EBOOL, ARRAY [n..m] OF INT, ARRAY [n..m] OF DINT</td>
<td>n and m are positive or negative integers or nil. Target table which contains the elements selected from Array1.</td>
</tr>
</tbody>
</table>

**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**

**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:

![FBD Diagram](image-url)
## Representation in LD

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

```
<table>
<thead>
<tr>
<th></th>
<th>DIV_INT_ARINT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
<td>ENO</td>
<td></td>
</tr>
<tr>
<td>IN1</td>
<td>OUT</td>
<td>Array1</td>
</tr>
<tr>
<td>IN2</td>
<td></td>
<td></td>
</tr>
</tbody>
</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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input_IN1</td>
<td>INT, DINT, ARRAY [n..m] OF INT, ARRAY [n..m] OF DINT</td>
<td>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.</td>
</tr>
<tr>
<td>Input_IN2</td>
<td>INT, DINT, ARRAY [n..m] OF INT, ARRAY [n..m] OF DINT</td>
<td>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.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
</table>
| 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**

### 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_ARWORD`
- `EQUAL_ARINT`
- `EQUAL_ARINT`
- `EQUAL_ARREAL`

### Representation in FBD

Representation applied to integer tables:

```
EQUAL_ARINT
IN1 Array1
IN2 Array2
P Position
OUT Equal1
```
Representation applied to integer tables:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1</td>
<td>ARRAY [n..m] OF WORD</td>
<td>n and m are positive or negative integers or nil.</td>
</tr>
<tr>
<td>Array2</td>
<td>ARRAY [n..m] OF DWORD</td>
<td>n and m are positive or negative integers or nil.</td>
</tr>
<tr>
<td>Position</td>
<td>INT</td>
<td>Rank of first element from which the search is launched.</td>
</tr>
</tbody>
</table>

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1</td>
<td>ARRAY [n..m] OF WORD</td>
<td>n and m are positive or negative integers or nil.</td>
</tr>
<tr>
<td>Array2</td>
<td>ARRAY [n..m] OF DWORD</td>
<td>n and m are positive or negative integers or nil.</td>
</tr>
<tr>
<td>Position</td>
<td>INT</td>
<td>Rank of first element from which the search is launched.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equal1</td>
<td>INT</td>
<td>Rank of first different elements. If the two tables are equivalent, Equal1 = -1.</td>
</tr>
</tbody>
</table>
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.
**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_ARREAL

### Representation in FBD

Representation applied to an integer table:

```
FIND_EQ_ARINT
 Array1 → IN1
 Value1 → IN2
 OUT → Row_Value1
```
Representation applied to an integer table:

- **FIND_EQ_ARINT**
  - **EN**
  - **ENO**
  - **IN1**
  - **OUT**
  - **Row_Value1**
  - **IN2**

Representation applied to an integer table:

- **LD Array1**
- **FIND_EQ_ARINT Value1**
- **ST Row_Value1**

Representation applied to an integer table:

- **Row_Value1**: FIND_EQ_ARINT(Array1, Value1);

Description of parameters

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1</td>
<td>ARRAY [n..m] OF INT, DINT, WORD, DWORD, REAL</td>
<td>n and m are positive or negative integers or nil.</td>
</tr>
<tr>
<td>Value1</td>
<td>INT, DINT, WORD, DWORD, REAL</td>
<td>Value whose rank is searched for in Array1. Of the same type as the elements of the table Array1.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row_Value1</td>
<td>INT</td>
<td>Rank of first element of Array1 equal to Value1. If none of the elements of the table is equal to Value1, Row_Value1 = -1</td>
</tr>
</tbody>
</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.
**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:

```
FIND_EQP_ARINT
<table>
<thead>
<tr>
<th>Array1</th>
<th>IN1</th>
<th>OUT</th>
<th>Row_Value1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value1</td>
<td>IN2</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>Begin1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Representation
in LD

Representation applied to an integer table:

```
FIND_EQP_ARINT

EN   ENO
Array 1  IN1  OUT  Row_Value1
Value1  IN2
Begin1  p
```

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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1</td>
<td>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</td>
<td>n and m are positive or negative integers or nil.</td>
</tr>
<tr>
<td>Value1</td>
<td>WORD, DWORD, INT, DINT, REAL.</td>
<td>Value whose rank is searched for in Array1. Of the same type as the elements of the table Array1.</td>
</tr>
<tr>
<td>Begin1</td>
<td>INT</td>
<td>Rank the search starts from</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row_Value1</td>
<td>INT</td>
<td>Rank of first element of Array1 equal to Value1. If none of the elements of the table is equal to Value1, Row_Value1 = -1</td>
</tr>
</tbody>
</table>

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

**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:

```
FIND_GT_ARINT
```

```
Array1 IN1 OUT Row_Value1
Value1 IN2
```

**Representation in LD**

Representation applied to an integer table:

```
FIND_GT_ARINT
```

```
EN ENO
Array1 IN1 OUT Row_Value1
Value1 IN2
```
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:

<table>
<thead>
<tr>
<th>Input</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1</td>
<td>ARRAY [n..m] OF WORD, DWORD, ARRAY [n..m] OF INT, ARRAY [n..m] OF DINT, ARRAY [n..m] OF REAL</td>
<td>n and m are positive or negative integers or nil.</td>
</tr>
<tr>
<td>Value1</td>
<td>WORD, DWORD, INT, DINT, REAL</td>
<td>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.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Output</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row_Value1</td>
<td>INT</td>
<td>Rank of the first element of Array1 &gt; than Value1. If none of the elements of the table is greater than Value1, Row_Value1 = -1</td>
</tr>
</tbody>
</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_LT_***: First element of a table less than a given value**

### 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:

```
<table>
<thead>
<tr>
<th>FIND_LT_ARINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1   IN1  OUT Row_Value1</td>
</tr>
<tr>
<td>Value1   IN2</td>
</tr>
</tbody>
</table>
```
Representation applied to an integer table:

**FIND_LT_ARINT**

<table>
<thead>
<tr>
<th>EN</th>
<th>ENO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1</td>
<td>IN1</td>
</tr>
<tr>
<td>Value1</td>
<td>IN2</td>
</tr>
</tbody>
</table>

Row_Value1

Representation applied to an integer table:

**LD** Array1

**FIND_LT_ARINT** Value1

**ST** Row_Value1

Representation applied to an integer table:

Row_Value1 := FIND_LT_ARINT(Array1, Value1);

Description of parameters

The following table describes the input parameters:

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

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row_Value1</td>
<td>INT</td>
<td>Rank of the first element of Array1 &lt; Value1. If none of the elements of the table is less than Value1, Row_Value1 = -1.</td>
</tr>
</tbody>
</table>
Runtime errors

When the table contains an invalid value or if \texttt{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**

### 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_ARBOOL`
- `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:

```
<table>
<thead>
<tr>
<th>Array1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>LENGTH_ARINT</td>
</tr>
<tr>
<td>OUT</td>
<td>Length_Array1</td>
</tr>
</tbody>
</table>
```
Representation applied to an integer table:

**Representation in LD**

```
   LENGTH_ARINT
   EN  ENO
   Array1  IN  OUT  Length_Array1
```

**Representation in IL**

```
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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1</td>
<td>ARRAY [n..m] OF EBOOL</td>
<td>n and m are positive or negative integers or nil.</td>
</tr>
<tr>
<td></td>
<td>ARRAY [n..m] OF WORD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ARRAY [n..m] OF DWORD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ARRAY [n..m] OF INT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ARRAY [n..m] OF DINT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ARRAY [n..m] OF REAL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ARRAY [n..m] OF BOOL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ARRAY [n..m] OF BYTE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ARRAY [n..m] OF DATE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ARRAY [n..m] OF DT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ARRAY [n..m] OF STRING</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ARRAY [n..m] OF TIME</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ARRAY [n..m] OF TOD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ARRAY [n..m] OF UINT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ARRAY [n..m] OF UDINT</td>
<td></td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length_Array1</td>
<td>INT</td>
<td>Table length (number of table elements).</td>
</tr>
</tbody>
</table>
MAX_***: Maximum value of table elements

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:

```
Array1 IN OUT Max1
```

Representation in LD

Representation applied to an integer table:

```
Array1 EN ENO
IN OUT Max1
```
Representation applied to an integer table:
LD Array1
MAX_ARINT
ST Max1

Representation applied to an integer table:
Max1 := MAX_ARINT(Array1);

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1</td>
<td>ARRAY [n..m] OF WORD, DWORD, INT, DINT, REAL</td>
<td>n and m are positive or negative integers or nil.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max1</td>
<td>WORD, DWORD, INT, DINT, REAL</td>
<td>Maximum value contained in the table. This result is of the same type as the table elements.</td>
</tr>
</tbody>
</table>

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

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:

```
  Array1  MIN_ARINT
    IN    OUT  Min1
```

Representation in LD
Representation applied to an integer table:

```
  Array1  MIN_ARINT
    EN    ENO
    IN    OUT  Min1
```
### 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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1</td>
<td>ARRAY [n..m] OF WORD, DWORD, DINT, INT, REAL</td>
<td><code>n</code> and <code>m</code> are positive or negative integers or nil.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min1</td>
<td>WORD, DWORD, DINT, INT, REAL</td>
<td>Minimum value contained in the table. This result is of the same type as the table elements.</td>
</tr>
</tbody>
</table>

### 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**

## 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:

```
MOD_INT_ARINT
IN1    OUT
IN2
Array 1
```
**Representation in LD**

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

```
MOD_INT_ARINT
  EN  ENO
Input_IN1  IN1  OUT  Array1
Input_IN2  IN2
```

**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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input_IN1</td>
<td>INT, DINT, ARRAY [n..m] OF INT, DINT</td>
<td>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.</td>
</tr>
<tr>
<td>Input_IN2</td>
<td>INT, DINT, ARRAY [n..m] OF INT, DINT</td>
<td>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.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
</table>
| Array1    | ARRAY [n..m] OF INT, 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

### 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_ARBOOL
- 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:

```
MOVE_INT_ARINT
IN       OUT
Val1     Array1
```

### Representation in LD

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

```
MOVE_INT_ARINT
EN       ENO
Val1     Array1
```
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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Val1</td>
<td>BOOL,</td>
<td>Val1 contains the value to be assigned to each element of the table Array1.</td>
</tr>
<tr>
<td></td>
<td>WORD,</td>
<td>Of the same type as the elements of the table Array1.</td>
</tr>
<tr>
<td></td>
<td>DWORD,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INT,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DINT,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>REAL</td>
<td></td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1</td>
<td>ARRAY [n..m] OF EBOOL, ARRAY [n..m] OF WORD, ARRAY [n..m] OF DWORD, Array1 is a table each element of which is of the value Val1.</td>
<td>n and m are positive or negative integers or nil. Array1 is a table each element of which is of the value Val1.</td>
</tr>
<tr>
<td></td>
<td>ARRAY [n..m] OF INT, ARRAY [n..m] OF DINT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ARRAY [n..m] OF REAL</td>
<td></td>
</tr>
</tbody>
</table>
MOVE_***_***: Table conversion

**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:

```
  MOVE_AREBOOL_INT
  IN1 OUT1
```

**Representation in LD**

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

```
  MOVE_AREBOOL_INT
   IN OUT
   EN ENO
  IN1 OUT
```
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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN1</td>
<td>INT, DINT, ARRAY [n..m] OF EBOOL</td>
<td>n and m are positive or negative integers or nil.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUT1</td>
<td>INT, DINT, ARRAY [n..m] OF EBOOL</td>
<td>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.</td>
</tr>
</tbody>
</table>
MUL_***_***: Multiplication of tables

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
</table>
| **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. |

<table>
<thead>
<tr>
<th>Available functions</th>
</tr>
</thead>
</table>
| **The available functions for the multiplication of the elements of a table by a number are as follows:**  
  ● MUL_ARINT_INT,  
  ● MUL_ARINT_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_ARINT. |

<table>
<thead>
<tr>
<th>Representation in FBD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Representation applied to the multiplication of the elements of an integer table by an integer:</td>
</tr>
</tbody>
</table>

```
     Input_IN1  Input_IN2  OUT
MUL_ARINT_INT
     IN1       IN2       Array1
```
**Representation in LD**

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

```
  MUL_ARINT_INT
  EN      ENO
  Input_IN1     IN1    OUT
  Input_IN2     IN2
```

**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);
```
The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input_IN1</td>
<td>INT, DINT, ARRAY [n..m] OF INT, DINT</td>
<td>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.</td>
</tr>
<tr>
<td>Input_IN2</td>
<td>INT, DINT, ARRAY [n..m] OF INT, DINT</td>
<td>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.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1</td>
<td>ARRAY [n..m] OF INT, DINT</td>
<td>According to the type of Input_IN1 and Input_IN2, each element of Array1 is the multiplication:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>of a single or double integer Input_IN1 by the corresponding element of the table Input_IN2 or else,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>of the elements of the table Input_IN1 by single or double integers Input_IN2 or else,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>of the elements of the table Input_IN1 by the respective elements of the table Input_IN2.</td>
</tr>
</tbody>
</table>

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**

**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_ARBOOL**
- **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:

```
NOT_ARWORD
Array1 IN
OUT Result_Array
```
### Representation applied to a 16-bit string:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1</td>
<td>ARRAY [n..m] OF EBOOL, WORD, DWORD</td>
<td>n and m are positive or negative integers or nil.</td>
</tr>
</tbody>
</table>

### Representation applied to a 16-bit string:

- LD `Array1`  
- ST `Result_Array`

### Representation applied to a 16-bit string:

- Result_Array := NOT_ARWORD(Array1);

### Description of parameters

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1</td>
<td>ARRAY [n..m] OF EBOOL, WORD, DWORD</td>
<td>n and m are positive or negative integers or nil.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result_Array</td>
<td>ARRAY [n..m] OF EBOOL, WORD, DWORD</td>
<td>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.</td>
</tr>
</tbody>
</table>
### OCCUR_***: Occurrence of a value in a table

#### 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_AWORD,
- OCCUR_ARDWORD,
- OCCUR_ARINT,
- OCCUR_ARDINT,
- OCCUR_ARREAL.

#### Representation in FBD

Representation applied to an integer table:

```plaintext
<table>
<thead>
<tr>
<th>OCCUR_ARINT</th>
<th>IN1</th>
<th>OUT</th>
<th>Occur_Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1</td>
<td>IN2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

#### Representation in LD

Representation applied to an integer table:

```plaintext
<table>
<thead>
<tr>
<th>OCCUR_ARINT</th>
<th>EN</th>
<th>ENO</th>
<th>Occur_Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1</td>
<td>IN1</td>
<td>OUT</td>
<td></td>
</tr>
<tr>
<td>Value1</td>
<td>IN2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Representation applied to an integer table:
LD Array1
OCCUR_ARINT Value1
ST Occur_Number

Representation applied to an integer table:
Occur_Number := OCCUR_ARINT(Array1,Value1);

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1</td>
<td>ARRAY [n..m] OF WORD, DWORD, INT, DINT, REAL</td>
<td>n and m are positive or negative integers or nil.</td>
</tr>
<tr>
<td>Value1</td>
<td>WORD, DWORD, INT, DINT, REAL</td>
<td>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 Array1.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Occur_Number</td>
<td>INT</td>
<td>Number of occurrences of Value1 in the table Array1.</td>
</tr>
</tbody>
</table>
### OR_***_***: Logical OR between tables and variables

<table>
<thead>
<tr>
<th>Description</th>
<th>The function carries out a logical OR (bit to bit) between:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• the elements of two tables,</td>
</tr>
<tr>
<td></td>
<td>• between a single type variable and the elements of a table,</td>
</tr>
<tr>
<td></td>
<td>• between the elements of a table and a single type variable.</td>
</tr>
</tbody>
</table>

**Note:** The result is always a table.

The additional parameters EN and ENO can be configured.

<table>
<thead>
<tr>
<th>Available functions</th>
<th>The functions available in the general library are the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• OR_AREBOOL (logical OR of two BOOL tables).</td>
</tr>
<tr>
<td></td>
<td>• OR_ARWORD (logical OR of two WORD tables).</td>
</tr>
<tr>
<td></td>
<td>• OR_ARWORD_WORD (logical OR of each element of a WORD table with a WORD).</td>
</tr>
<tr>
<td></td>
<td>• OR_ARDWORD_DWORD (logical OR of each element of a DWORD table with a DWORD).</td>
</tr>
<tr>
<td></td>
<td>• OR_ARDWORD (logical OR of two DWORD tables).</td>
</tr>
</tbody>
</table>

The functions available in the **Obsolete** library are the following:

<p>|                     | • OR_ARINT_INT (logical OR of each element of an INT table with an INT). |
|                     | • OR_ARINTDINT (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). |</p>
<table>
<thead>
<tr>
<th>Representation in FBD</th>
<th>Representation applied to a 16-bit string table and a 16-bit string:</th>
</tr>
</thead>
<tbody>
<tr>
<td>OR_ARWORD_WORD</td>
<td>Input_IN1</td>
</tr>
<tr>
<td></td>
<td>Input_IN2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Representation in LD</th>
<th>Representation applied to a 16-bit string table and a 16-bit string:</th>
</tr>
</thead>
<tbody>
<tr>
<td>OR_ARWORD_WORD</td>
<td>EN</td>
</tr>
<tr>
<td></td>
<td>Input_IN1</td>
</tr>
<tr>
<td></td>
<td>Input_IN2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Representation in IL</th>
<th>Representation applied to a 16-bit string table and a 16-bit string:</th>
</tr>
</thead>
<tbody>
<tr>
<td>LD_IN1</td>
<td>OR_ARWORD_WORD Input_IN2</td>
</tr>
<tr>
<td>ST_Array1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Representation in ST</th>
<th>Representation applied to a 16-bit string and a 16-bit string table:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1 := OR_ARWORD_WORD(Input_IN1, Input_IN2);</td>
<td></td>
</tr>
</tbody>
</table>
### Description of parameters

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input_IN1</td>
<td>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</td>
<td>( n ) and ( m ) are positive or negative integers or nil.</td>
</tr>
<tr>
<td>Input_IN2</td>
<td>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</td>
<td>( n ) and ( m ) are positive or negative integers or nil.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
</table>
| 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

#### 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,
- ROLARDINT,
- ROL_ARREAL.

#### Representation in FBD

Representation applied to an integer table:

```
<table>
<thead>
<tr>
<th>Positions</th>
<th>N</th>
<th>Array1</th>
</tr>
</thead>
<tbody>
<tr>
<td>INOUT</td>
<td>INOUT</td>
<td>Array1</td>
</tr>
</tbody>
</table>
```

ROL_ARINT
Representation applied to an integer table:

```
ROL_ARINT
EN  ENO
Positions  N
Array1  INOUT  INOUT  Array1
```

Representation applied to an integer table:

**LD** Positions
ROL_ARINT Array1

Representation applied to an integer table:

ROL_ARINT(Positions, Array1);

Description of parameters

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positions</td>
<td>INT</td>
<td>Shift value according to the ascending indices of the table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Example</strong>: Positions = 2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: if the value of Positions is negative or nil, no shift is carried out.</td>
</tr>
</tbody>
</table>

The following table describes the input/output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1</td>
<td>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</td>
<td>The elements of Array1 are moved a number of positions equal to Positions. The shift is carried out according to the ascending indices. <strong>Example</strong>: 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.</td>
</tr>
</tbody>
</table>
ROR_***: Rotate shift to right

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_ARMWORD,
- ROR_ARINT,
- ROR_ARDINT,
- ROR_ARREAL.

Representation in FBD
Representation applied to an integer table:

<table>
<thead>
<tr>
<th>Positions</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1</td>
<td>INOUT</td>
</tr>
<tr>
<td>INOUT</td>
<td>INOUT</td>
</tr>
</tbody>
</table>
**ROR_***

**Representation in LD**

Representation applied to an integer table:

```
ROR_ARINT

<table>
<thead>
<tr>
<th>EN</th>
<th>ENO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positions</td>
<td>N</td>
</tr>
<tr>
<td>Array1</td>
<td>INOUT</td>
</tr>
</tbody>
</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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positions</td>
<td>INT</td>
<td>Shift value according to the descending indices of the table.</td>
</tr>
</tbody>
</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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1</td>
<td>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</td>
<td>The elements of Array1 are moved a number of positions equal to Positions. The shift is carried out according to the descending indicators.</td>
</tr>
</tbody>
</table>

**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**

**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:

```
<table>
<thead>
<tr>
<th>Direction</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1</td>
<td>INOUT</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>SORT_ARINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1</td>
</tr>
</tbody>
</table>
```

**Representation in LD**

Representation applied to an integer table:

```
<table>
<thead>
<tr>
<th>Direction</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1</td>
<td>INOUT</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>SORT_ARINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
</tr>
<tr>
<td>ENO</td>
</tr>
<tr>
<td>Array1</td>
</tr>
</tbody>
</table>
```
Representation applied to an integer table:
LD Direction
SORT_ARINT Array1

Representation applied to an integer table:
SORT_ARINT(Direction, Array1);

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direction</td>
<td>INT</td>
<td>Direction of sort to be carried out:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Direction ≥ 0: ascending sort,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Direction &lt; 0: descending sort.</td>
</tr>
</tbody>
</table>

The following table describes the input/output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1</td>
<td>ARRAY [n..m] OF WORD, DWORD, INT, DINT, REAL</td>
<td>Table sorted in the direction specified in Direction, n and m are positive or negative integers, or nil.</td>
</tr>
</tbody>
</table>
**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:

```
SUB_INT_ARINT

IN1   OUT
Input_IN1
IN2
Input_IN2
Array1
```
Representation applied to the subtraction of the elements of a table of integers from an integer:

**LD**

```
<table>
<thead>
<tr>
<th>SUB_INT_ARINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
</tr>
<tr>
<td>ENO</td>
</tr>
<tr>
<td>Input_IN1</td>
</tr>
<tr>
<td>IN1</td>
</tr>
<tr>
<td>OUT Input_IN2</td>
</tr>
<tr>
<td>IN2</td>
</tr>
<tr>
<td>Array1</td>
</tr>
</tbody>
</table>
```

**IL**

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

**ST**

```
Array1 := SUB_INT_ARINT(Input_IN1, Input_IN2);
```
Description of parameters

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input_IN1</td>
<td>INT, DINT, ARRAY [n..m] OF INT, ARRAY [n..m] OF DINT</td>
<td>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.</td>
</tr>
<tr>
<td>Input_IN2</td>
<td>INT, DINT, ARRAY [n..m] OF INT, ARRAY [n..m] OF DINT</td>
<td>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.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
</table>
| 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

**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:

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

**Description:**

<table>
<thead>
<tr>
<th>Element</th>
<th>Signification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1</td>
<td>Table declared in the following way: ARRAY [n..m] OF ...</td>
</tr>
</tbody>
</table>

**Representation in FBD**

Representation applied to an integer table:

```
<table>
<thead>
<tr>
<th></th>
<th>SUM_ARINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1</td>
<td>IN</td>
</tr>
<tr>
<td></td>
<td>OUT</td>
</tr>
<tr>
<td></td>
<td>Sum1</td>
</tr>
</tbody>
</table>
```
Representation applied to an integer table:

```
SUM_ARINT
EN   ENO
Array1 IN   OUT
Sum1
```

Representation applied to an integer table:

```
LD Array1
SUM_ARINT
ST Sum1
```

Representation applied to an integer table:

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

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1</td>
<td>ARRAY [n..m] OF INT, DINT, REAL</td>
<td>Double or single integer tables or tables of reals, n and m are positive or negative integers, or nil.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum1</td>
<td>INT, DINT, REAL</td>
<td>Sum of table elements assigned to input. The sum is of the same type as the table elements.</td>
</tr>
</tbody>
</table>

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**

### 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:

```
Array1  INOUT      INOUT  Array1
       SWAP_ARINT
```

### Representation in LD

Representation applied to an integer table:

```
Array1  INOUT      INOUT  Array1
       SWAP_ARINT
EN      ENO
```

### Representation in IL

Representation applied to an integer table:

```
LD Array1
SWAP_ARINT
```
Representation applied to an integer table:

```
SWAP_ARINT(Array1);
```

The following table describes the input/output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array1</td>
<td>ARRAY [n..m] OF INT</td>
<td>n and m are positive or negative integers or nil. On output, the bytes of Array1 have been permuted.</td>
</tr>
<tr>
<td></td>
<td>ARRAY [n..m] OF WORD</td>
<td></td>
</tr>
</tbody>
</table>
XOR_***_***: Exclusive OR between tables

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_ARBOOL (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:

```
XOR_ARWORD_WORD
IN1  IN2  OUT  Array1
Input_IN1  Input_IN2
```

Representation in LD

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

```
XOR_ARWORD_WORD
EN   ENO
IN1  OUT  Array1
Input_IN1  Input_IN2
```

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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input_IN1</td>
<td>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</td>
<td>n and m are positive or negative integers or nil.</td>
</tr>
<tr>
<td>Input_IN2</td>
<td>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</td>
<td>n and m are positive or negative integers or nil.</td>
</tr>
</tbody>
</table>

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.
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:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Chapter Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>Introduction to integer regulation functions</td>
<td>119</td>
</tr>
<tr>
<td>30</td>
<td>PID_INT: PID controller</td>
<td>125</td>
</tr>
<tr>
<td>31</td>
<td>PWM_INT: Pulse width modulation of a numerical value</td>
<td>133</td>
</tr>
<tr>
<td>32</td>
<td>SERVO_INT: Servo drive function</td>
<td>137</td>
</tr>
</tbody>
</table>
Introduction to integer regulation functions

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:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Introduction</td>
<td>120</td>
</tr>
<tr>
<td>Principal of the regulation loop</td>
<td>121</td>
</tr>
<tr>
<td>Development methodology for a regulation application</td>
<td>122</td>
</tr>
<tr>
<td>Programming a regulation function</td>
<td>123</td>
</tr>
<tr>
<td>Behavior of functions in operating modes</td>
<td>124</td>
</tr>
</tbody>
</table>
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.
Introduction to regulation EFs

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).

```
Application -> Configuration
  Configuration of interfaces
  Discrete, Analog, Counts

Application -> Data
  Entry of constant data, mnemonics,
  numerical values

Programming: Ladder, List
  MAST, FAST, SR
  Regulation, Human Machine Interface

PLC -> Connect
  Transfer of the application
to the PLC

Animation tables
  Variable table

Debugging
  program and
  Debugging
  from the operator terminal

File -> Save
  Saving of the application

Use of the control loops

Exploitation of process via
the operator terminal

Documentation
  Application documentation
```
Introduction to regulation EFs

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

Description

Subject of this Chapter
This chapter describes the PID_INT function.

What's in this Chapter?
This chapter contains the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function description</td>
<td>126</td>
</tr>
<tr>
<td>Description of Derived Data</td>
<td>130</td>
</tr>
</tbody>
</table>
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.
The following illustration provides the operating synoptic for the PID function.

**The Setpoint branch**

- Internal Setpoint (Internal measurement used)
- Setpoint limit (10000, 0)
- Measurement limit (0)
- Measurement used

**The Measurement branch**

- Process value (PV)
- Measurement (Internal measurement used)

**The PID action**

- Deviation
- Derivative action on deviation
- Integrated action
- Corrector P.I.D
- Correlator

**The PID operating modes**

- Monitoring without step by step of the command on switching from Auto -> Manu
- OUT_MAX
- OUT_MIN
- OUT_MAN
- AUTO
- OUTP

**HUMAN MACHINE INTERFACE**

- PV_MMI
- PV_SUP
- SP_MMI
- PV_INF

Limiter

127
**Representation in FBD**

**Representation:**

```
PID_INT
```

```
Input_Tag   TAG   OUT   PID_Out
Input_Unit  UNIT
Input_PV    PV
In_Out_Auto AUTO   AUTO   In_Out_Auto
In_Out_Para PARA   PARA   In_Out_Para
```

**Representation in LD**

**Representation:**

```
PID_INT
```

```
EN   ENO
Input_Tag   TAG   OUT   PID_Out
Input_Unit  UNIT
Input_PV    PV
In_Out_Auto AUTO   AUTO   In_Out_Auto
In_Out_Para PARA   PARA   In_Out_Para
```

**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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input_Tag</td>
<td>STRING</td>
<td>Name of PID used by the terminal. String of 8 characters</td>
</tr>
<tr>
<td>Input_Unit</td>
<td>STRING</td>
<td>Unit of measurement used by the terminal. String of 6 characters</td>
</tr>
<tr>
<td>Input_PV</td>
<td>INT</td>
<td>Process value input Measurement format [0..10000].</td>
</tr>
</tbody>
</table>

The following table describes the input/output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>In_Out_Auto</td>
<td>EBOOL</td>
<td>Input/output bit which indicates and manages the operating modes of the PID and the terminal: 0 : manual, 1 : auto.</td>
</tr>
<tr>
<td>In_Out_Para</td>
<td>ARRAY [n..m] OF INT</td>
<td>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.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PID_Out</td>
<td>INT</td>
<td>Analog output of PID, if TI = 0, an offset of 5000 is added to the OUT output in auto mode. Output format [0;,+10000].</td>
</tr>
</tbody>
</table>
## Description of Derived Data

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Rank</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP</td>
<td>PARA[0]</td>
<td>Internal setpoint in 0 - 10000 format.</td>
</tr>
<tr>
<td>OUT_MAN</td>
<td>PARA[1]</td>
<td>Value of the manual output of the PID (between 1 and 1000).</td>
</tr>
<tr>
<td>KP</td>
<td>PARA[2]</td>
<td>Proportional gain of the PID (x100), signed without unit (-10000&lt;KP&lt;+10000). The Kp sign determines the direction of the PID’s action (negative: forward, positive: reverse).</td>
</tr>
<tr>
<td>TI</td>
<td>PARA[3]</td>
<td>The PID’s integral time (between 0 and 20000) is shown in 0.1 seconds.</td>
</tr>
<tr>
<td>TD</td>
<td>PARA[4]</td>
<td>The PID’s derivative time (between 0 and 10000) is shown in 0.1 seconds.</td>
</tr>
<tr>
<td>TS</td>
<td>PARA[5]</td>
<td>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.</td>
</tr>
<tr>
<td>OUT_MAX</td>
<td>PARA[6]</td>
<td>Upper limit of the PID’s output in automatic (between 0 and 10000).</td>
</tr>
<tr>
<td>OUT_MIN</td>
<td>PARA[7]</td>
<td>Lower limit of the PID’s output in automatic (between 0 and 10000).</td>
</tr>
<tr>
<td>PV_DEV</td>
<td>PARA[8].0</td>
<td>Derived action choice 0 = on process variable, 1 = on deviation.</td>
</tr>
<tr>
<td>NO_BUMP</td>
<td>PARA[8].4</td>
<td>Bumpless or non-bumpless mode. 0 = non-bumpless, 1 = bumpless.</td>
</tr>
<tr>
<td>DEVAL_MMI</td>
<td>PARA[8].8</td>
<td>= 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.</td>
</tr>
<tr>
<td>PV_SUP*</td>
<td>PARA[9]</td>
<td>Upper limit of the measurement scale’s range, in a physical unit (x100) (between –9 999 999 and +9 999 999).</td>
</tr>
<tr>
<td>PV_INF*</td>
<td>PARA[11]</td>
<td>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 -9 999 999 and +9 999 999).</td>
</tr>
</tbody>
</table>
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:

\[
\varepsilon_t = SP - PV \\
\text{OUT} = KP \left[ \varepsilon t + Dt \right] / 100 + 5000 \\
Dt = \text{derived action}
\]

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

\[
\varepsilon_t = SP - PV \\
\Delta \text{OUT} = KP \left[ \Delta t + (TS/10. TI) \varepsilon t + \Delta Dt \right] / 100 \\
\text{OUT} = \text{OUT} + \Delta \text{OUT} \\
Dt = \text{derived action}
\]

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.

---

**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).
PID_INT
**PWM_INT: Pulse width modulation of a numerical value**

### 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:
**PWM_INT**

**Representation in FBD**

Representation:

```
  PWM_INT
   Input_INP ——— INP ——— PW_O ——— PW_O_Out
   In_Out_Para ——— PARA ——— PARA ——— In_Out_Para
```

**Representation in LD**

Representation:

```
  PWM_INT
   EN ——— ENO
   Input_INP ——— INP ——— PW_O ——— PW_O_Out
   In_Out_Para ——— PARA ——— PARA ——— In_Out_Para
```

**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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input_INP</td>
<td>INT</td>
<td>Analog value to be modulated in pulse width (format [0 – 10000]).</td>
</tr>
</tbody>
</table>

The following table describes the input/output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>In_Out_Para</td>
<td>ARRAY [n..m] OF INT</td>
<td>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.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PW_O_Out</td>
<td>EBOOL</td>
<td>Analog output of PID, if TI = 0, an offset of 5000 is added to the OUT output in auto mode.</td>
</tr>
</tbody>
</table>
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) = $IPA \times T_{MOD} / 1000$

The following timing diagram illustrates this formula:

![Timing Diagram]

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)$ \times 10 \times T_{MOD}$.

The PID is in the MAST task, the MAST's period is $50 \times 10^{-3}$ s, $TS = 500 \times 10^{-2}$ 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 $500 \times 10 / 50 = 100 \times 10^{-3}$ s.

The PWM function can therefore be programmed in the MAST task.

The resolution will be $1/100$. 
SERVO_INT: Servo drive function

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:

```
SERVO_INT
Input_Inp INP UP Out_Up
Input_Pot POT DOWN Out_Down
In_Out_Pid PID PID In_Out_Pid
In_Out_Para PARA PARA In_Out_Para
```
SERVO_INT

Representation in LD

Representation:

```
SERVO_INT
  EN  ENO
  INP UP Out_Up
  POT DOWN Out_Down
  PID PID In_Out_Pid
  PARA PARA In_Out_Para
```

Representation in IL

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

Representation in ST

```
SERVO_INT(Input_Inp, Input_Pot, In_Out_Pid, In_Out_Para, Out_Up, Out_Down);
```
The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input_Inp</td>
<td>INT</td>
<td>Position setpoint ([0 -10000] format) that has to be connected to the PID output.</td>
</tr>
<tr>
<td>Input_Pot</td>
<td>INT</td>
<td>Position copy, ([0 -10000] format) 0 : closed valve; 10000: open valve.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the copy does not exist, Input_POT must be initialized at -10000. This particular value indicates &quot;no copy&quot;.</td>
</tr>
</tbody>
</table>

The following table describes the input/output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>In_Out_Pid</td>
<td>ARRAY [n..m] OF INT</td>
<td>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.</td>
</tr>
<tr>
<td>In_Out_Para</td>
<td>ARRAY [n..m] OF INT</td>
<td>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):</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- In_Out_PARA[0] also called T_MOTOR is the valve opening time expressed in 10^{-2} s.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- In_Out_PARA[1] also called T_MINI is the minimum pulse expressed in 10^{-2} s.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- In_Out_PARA[2] also called HYST is the hysteresis value in [0-10000] format.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> 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.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Out_Up</td>
<td>EBOOL</td>
<td>Output signal for the motor’s Out_Up operating direction.</td>
</tr>
<tr>
<td>Out_Down</td>
<td>EBOOL</td>
<td>Output signal for the motor’s Out_Down operating direction.</td>
</tr>
</tbody>
</table>
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.

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:

\[ T_{\text{IMP}} \text{ (expressed in } 10^{-3} \text{ s)} = \text{OUT} \times \frac{T_{\text{MOTOR}}}{1000} \]

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:

```plaintext
<table>
<thead>
<tr>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The PID output variation is +20% (( T_{\text{MOTOR}} ) pulse = 25 s for a 100% variation). In this case, the pulse affects the UP output for a period of 5 s.</td>
</tr>
<tr>
<td>2</td>
<td>The PID variation is +2%, which would correspond to a pulse of 0.5 s. This pulse is less than ( T_{\text{MINI}} ) (= 1 s.), and it does not affect the outputs.</td>
</tr>
<tr>
<td>3</td>
<td>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.</td>
</tr>
<tr>
<td>4</td>
<td>A variation of -24% appears and the activated pulse is therefore of 6 s on the DOWN output.</td>
</tr>
<tr>
<td>5</td>
<td>Before the following second has elapsed, another variation of +22% brings the system back to a global variation of 2% &lt; the variation of ( T_{\text{MINI}} ) (4%). The function finishes carrying out the minimal pulse of 1 s.</td>
</tr>
</tbody>
</table>
```

**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

## 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:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Chapter Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>EQ: Equal to</td>
<td>145</td>
</tr>
<tr>
<td>34</td>
<td>GE: Greater than or equal to</td>
<td>147</td>
</tr>
<tr>
<td>35</td>
<td>GT: Greater than</td>
<td>151</td>
</tr>
<tr>
<td>36</td>
<td>LE: Less than or equal to</td>
<td>155</td>
</tr>
<tr>
<td>37</td>
<td>LT: Less than</td>
<td>159</td>
</tr>
<tr>
<td>38</td>
<td>NE: Not equal to</td>
<td>163</td>
</tr>
</tbody>
</table>
EQ: Equal to

**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**

\[ \text{OUT} = 1, \text{if } (\text{IN}_1 = \text{IN}_2) \land (\text{IN}_2 = \text{IN}_3) \land \ldots \land (\text{IN}_{(n-1)} = \text{IN}_n) \]

**Representation in FBD**

<table>
<thead>
<tr>
<th>Value1</th>
<th>IN1</th>
<th>EQ</th>
<th>IN2</th>
<th>OUT</th>
<th>Result</th>
</tr>
</thead>
</table>

**Representation in LD**

<table>
<thead>
<tr>
<th>Value1</th>
<th>IN1</th>
<th>EQ</th>
<th>EN</th>
<th>ENO</th>
<th>IN2</th>
<th>OUT</th>
<th>Result</th>
</tr>
</thead>
</table>
**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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value1</td>
<td>BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD</td>
<td>1. Input</td>
</tr>
<tr>
<td>Value2</td>
<td>BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD</td>
<td>2. Input</td>
</tr>
<tr>
<td>Value(n)</td>
<td>BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD</td>
<td>n. input</td>
</tr>
</tbody>
</table>

n = max 31

Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result</td>
<td>BOOL</td>
<td>Output</td>
</tr>
</tbody>
</table>

**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

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

$$\text{OUT} = 1, \text{if } (\text{IN}_1 \geq \text{IN}_2) \& (\text{IN}_2 \geq \text{IN}_3) \& \ldots \& (\text{IN}_{(n-1)} \geq \text{IN}_n)$$

Representation in FBD

<table>
<thead>
<tr>
<th>Value1</th>
<th>IN1</th>
<th>OUT</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value2</td>
<td>IN2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Representation in LD

Representation:

```
EN                ENO
Value1           IN1       OUT
Value2           IN2
```

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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value1</td>
<td>BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD</td>
<td>1. Input</td>
</tr>
<tr>
<td>Value2</td>
<td>BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD</td>
<td>2. Input</td>
</tr>
<tr>
<td>Valuen</td>
<td>BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD</td>
<td>n. input (n = max 31)</td>
</tr>
</tbody>
</table>

Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result</td>
<td>BOOL</td>
<td>Output</td>
</tr>
</tbody>
</table>
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

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

\[ \text{OUT} = 1, \text{if} \ (\text{IN}_1 > \text{IN}_2) \land (\text{IN}_2 > \text{IN}_3) \land \ldots (\text{IN}_{(n-1)} > \text{IN}_n) \]

Representation in FBD

<table>
<thead>
<tr>
<th>Value1</th>
<th>IN1</th>
<th>GT</th>
<th>Value2</th>
<th>IN2</th>
<th>OUT</th>
<th>Result</th>
</tr>
</thead>
</table>

EN and ENO
### Representation in LD

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value1</td>
<td>BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD</td>
<td>1. Input</td>
</tr>
<tr>
<td>Value2</td>
<td>BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD</td>
<td>2. Input</td>
</tr>
<tr>
<td>Value(n)</td>
<td>BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD</td>
<td>n. input</td>
</tr>
</tbody>
</table>

### Representation in IL

LD Value1
GT Value2
ST Result

### Representation in ST

Result := GT (Value1, Value2) ;

### Parameter description

**Description of the input parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value1</td>
<td>BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD</td>
<td>1. Input</td>
</tr>
<tr>
<td>Value2</td>
<td>BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD</td>
<td>2. Input</td>
</tr>
<tr>
<td>Value(n)</td>
<td>BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD</td>
<td>n. input</td>
</tr>
</tbody>
</table>

**Description of the output parameter:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result</td>
<td>BOOL</td>
<td>Output</td>
</tr>
</tbody>
</table>
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

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

\[ \text{OUT} = 1, \text{if } (\text{IN}_1 \leq \text{IN}_2) \& (\text{IN}_2 \leq \text{IN}_3) \& \ldots \& (\text{IN}_{(n-1)} \leq \text{IN}_n) \]

Representation in FBD

```
<table>
<thead>
<tr>
<th>Value1</th>
<th>IN1</th>
<th>LE</th>
<th>Value2</th>
<th>IN2</th>
<th>OUT</th>
<th>Result</th>
</tr>
</thead>
</table>
```


### Representation in LD

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result</td>
<td>BOOL</td>
<td>Output</td>
</tr>
</tbody>
</table>

### Representation in IL

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

### Representation in ST

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

### Parameter Description

#### Description of the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value1</td>
<td>BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD</td>
<td>1. Input</td>
</tr>
<tr>
<td>Value2</td>
<td>BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD</td>
<td>2. Input</td>
</tr>
<tr>
<td>Valuen</td>
<td>BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD</td>
<td>n. input</td>
</tr>
</tbody>
</table>

n = max 31

#### Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result</td>
<td>BOOL</td>
<td>Output</td>
</tr>
</tbody>
</table>
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**

**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**

\[ \text{OUT} = 1, \text{if } (\text{IN}_1 < \text{IN}_2) \& (\text{IN}_2 < \text{IN}_3) \& .. \& (\text{IN}_{n-1} < \text{IN}_n) \]

**Representation in FBD**

Representation:

\[
\begin{array}{ccc}
\text{LT} & \text{IN}_1 & \text{OUT} \\
\text{Value 1} & \text{IN}_1 & \text{Result} \\
\text{Value 2} & \text{IN}_2 & \\
\end{array}
\]
### Representation in LD

### Representation in IL

### Representation in ST

### Parameter description

**Description of the input parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value1</td>
<td>BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD</td>
<td>1. Input value</td>
</tr>
<tr>
<td>Value2</td>
<td>BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD</td>
<td>2. Input value</td>
</tr>
<tr>
<td>Valuen</td>
<td>BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD</td>
<td>n. input value (n = max 31)</td>
</tr>
</tbody>
</table>

**Description of the output parameter:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result</td>
<td>BOOL</td>
<td>Output value</td>
</tr>
</tbody>
</table>
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

## 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

<table>
<thead>
<tr>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUT = 1, if IN1 &lt;&gt; IN2</td>
</tr>
</tbody>
</table>

## Representation in FBD

<table>
<thead>
<tr>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Representation:</strong></td>
</tr>
<tr>
<td>Value1</td>
</tr>
<tr>
<td>Value2</td>
</tr>
</tbody>
</table>

## Representation in LD

<table>
<thead>
<tr>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Representation:</strong></td>
</tr>
<tr>
<td>EN</td>
</tr>
<tr>
<td>Value1</td>
</tr>
<tr>
<td>Value2</td>
</tr>
</tbody>
</table>

## Representation in IL

<table>
<thead>
<tr>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Representation:</strong></td>
</tr>
<tr>
<td>LD Value1</td>
</tr>
<tr>
<td>NE Value2</td>
</tr>
<tr>
<td>ST Result</td>
</tr>
</tbody>
</table>
Representation in ST

Result := NE (Value1, Value2) ;

Parameter description

Description of the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value1</td>
<td>BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD</td>
<td>1. Input</td>
</tr>
<tr>
<td>Value2</td>
<td>BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT, TOD</td>
<td>2. Input</td>
</tr>
</tbody>
</table>

Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result</td>
<td>BOOL</td>
<td>Output</td>
</tr>
</tbody>
</table>

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

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:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Chapter Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>39</td>
<td>ADD_***_TIME: Addition of a duration to a date</td>
<td>167</td>
</tr>
<tr>
<td>40</td>
<td>DIVTIME: Division</td>
<td>169</td>
</tr>
<tr>
<td>41</td>
<td>MULTIME: Multiplication</td>
<td>171</td>
</tr>
<tr>
<td>42</td>
<td>SUB_<em><strong>_</strong></em>: Calculates the time difference between two dates or times</td>
<td>173</td>
</tr>
<tr>
<td>43</td>
<td>SUB_***_TIME: Subtraction of a duration from a date</td>
<td>175</td>
</tr>
</tbody>
</table>
ADD_***_TIME: Addition of a duration to a date

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:

```
<table>
<thead>
<tr>
<th>Source_Value</th>
<th>Time_to_Add</th>
<th>ADD_TOD_TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN1</td>
<td>IN2</td>
<td>OUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Result_Value</td>
</tr>
</tbody>
</table>
```

Representation in LD

Representation applied to a time of day:

```
<table>
<thead>
<tr>
<th>Source_Value</th>
<th>Time_to_Add</th>
<th>ADD_TOD_TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
<td>ENO</td>
<td>OUT</td>
</tr>
<tr>
<td>IN1</td>
<td>IN2</td>
<td>Result_Value</td>
</tr>
</tbody>
</table>
```

167
Representation applied to a time of day:

LD Source_Value
ADD_TOD_TIME Time_to_Add
ST Result_Value

Representation applied to a time of day:

Result_Value := ADD_TOD_TIME(Source_Value, Time_to_Add);

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source_Value</td>
<td>DT, TOD</td>
<td>Date or time.</td>
</tr>
<tr>
<td>Time_to_Add</td>
<td>TIME</td>
<td>Duration to be added to Source_Value</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> 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.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result_Value</td>
<td>DT, TOD</td>
<td>Result_Value is of the same type as Source_Value.</td>
</tr>
</tbody>
</table>

**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**

**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**

\[ \text{OUT} = \frac{(\text{IN1})}{(\text{IN2})} \]

**Representation in FBD**

[Diagram of FBD representation]

**Representation in LD**

[Diagram of LD representation]

**Representation in IL**

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

Representation in ST

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

Parameter description

Description of input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME_variable</td>
<td>TIME</td>
<td>Dividend</td>
</tr>
<tr>
<td>Divisor</td>
<td>INT, DINT, UINT, UDINT, REAL</td>
<td>Divisor</td>
</tr>
</tbody>
</table>

Description of output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quotient</td>
<td>TIME</td>
<td>Quotient</td>
</tr>
</tbody>
</table>

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

## 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

$$\text{OUT} = \text{IN1} \times \text{IN2}$$

## Representation in FBD

Representation:

```
TIME_variable   \hspace{1cm} \text{IN1} \hspace{1cm} \text{OUT} \hspace{1cm} \text{Product}
\text{Factor}   \hspace{1cm} \text{IN2}
```

## Representation in LD

Representation:

```
\text{MULTIME}
\text{EN} \hspace{1cm} \text{ENO}
\text{TIME_variable} \hspace{1cm} \text{IN1} \hspace{1cm} \text{OUT} \hspace{1cm} \text{Product}
\text{Factor} \hspace{1cm} \text{IN2}
```

## Representation in IL

Representation:

```
LD \text{TIME_variable}
\text{MULTIME} \text{Factor}
\text{ST Product}
```
**MULTIME**

**Representation in ST**

Representation:

Product := MULTIME (TIME_variable, Factor) ;

**Parameter description**

Description of input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME_variable</td>
<td>TIME</td>
<td>Multiplicand (factor)</td>
</tr>
<tr>
<td>Factor</td>
<td>INT, DINT, UINT, UDINT, REAL</td>
<td>Multiplier (factor)</td>
</tr>
</tbody>
</table>

Description of output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>TIME</td>
<td>Product</td>
</tr>
</tbody>
</table>

**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**

### 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:

```
    SUB_TOD_TOD
    IN1    OUT    Delay1
Input_IN1
Input_IN2
```

### Representation in LD

Representation applied to a time of day:

```
    SUB_TOD_TOD
    EN    ENO
    IN1    OUT    Delay1
Input_IN1
Input_IN2
```
Representation applied to a time of day:
LD Input_IN1
SUB_TOD_TOD Input_IN2
ST Delay1

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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input_IN1</td>
<td>DATE, DT, TOD</td>
<td>Date or time for which we wish to calculate the difference with Input_IN2.</td>
</tr>
<tr>
<td>Input_IN2</td>
<td>DATE, DT, TOD</td>
<td>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.</td>
</tr>
</tbody>
</table>

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

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay1</td>
<td>TIME</td>
<td>Delay1 contains the time expressed as an absolute value elapsed between the two entries Input_IN1 and Input_IN2.</td>
</tr>
</tbody>
</table>

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.
Subtraction of a duration from a date

**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:

```
<table>
<thead>
<tr>
<th>Source_Value</th>
<th>Time_to_Sub</th>
<th>IN1</th>
<th>OUT</th>
<th>Result_Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUB_TOD_TIME</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**Representation in LD**
Representation applied to a time of day:

```
<table>
<thead>
<tr>
<th>Source_Value</th>
<th>Time_to_Sub</th>
<th>IN1</th>
<th>OUT</th>
<th>Result_Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUB_TOD_TIME</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

EN | ENO
Representation applied to a time of day:
LD Source_Value
SUB_TOD_TIME Time_to_Sub
ST Result_Value

Representation applied to a time of day:
Result_Value := SUB_TOD_TIME(Source_Value, Time_to_Sub);

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source_Value</td>
<td>DT, TOD</td>
<td>Date or time.</td>
</tr>
<tr>
<td>Time_to_Sub</td>
<td>TIME</td>
<td>Duration to subtract from Source_Value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result_Value</td>
<td>DT, TOD</td>
<td>Result_Value is of the same type as Source_Value.</td>
</tr>
</tbody>
</table>

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 to 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 to 1 and Result_Value applies:
- 00:00:00 for the type TOD.
- 00001-01-01-00:00:00 for the type DT.
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:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Chapter Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>AND: AND function</td>
<td>179</td>
</tr>
<tr>
<td>45</td>
<td>F_TRIG: Falling edge detection</td>
<td>181</td>
</tr>
<tr>
<td>46</td>
<td>FE: Detection of Falling Edge</td>
<td>183</td>
</tr>
<tr>
<td>47</td>
<td>NOT: Negation</td>
<td>185</td>
</tr>
<tr>
<td>48</td>
<td>OR: OR function</td>
<td>187</td>
</tr>
<tr>
<td>49</td>
<td>R_TRIG: Rising edge detection</td>
<td>189</td>
</tr>
<tr>
<td>50</td>
<td>RE: Detection of Rising Edge</td>
<td>191</td>
</tr>
<tr>
<td>51</td>
<td>RESET: Setting of a bit to 0</td>
<td>193</td>
</tr>
<tr>
<td>52</td>
<td>ROL: Rotate left</td>
<td>195</td>
</tr>
<tr>
<td>53</td>
<td>ROR: Rotate right</td>
<td>197</td>
</tr>
<tr>
<td>54</td>
<td>RS: Bistable function block, reset dominant</td>
<td>199</td>
</tr>
<tr>
<td>55</td>
<td>SET: Setting of a bit to 1</td>
<td>201</td>
</tr>
<tr>
<td>56</td>
<td>SHL: Shift left</td>
<td>203</td>
</tr>
<tr>
<td>57</td>
<td>SHR: Shift right</td>
<td>205</td>
</tr>
<tr>
<td>58</td>
<td>SR: Bistable function block, set dominant</td>
<td>207</td>
</tr>
<tr>
<td>59</td>
<td>TRIGGER: Detection of all edges</td>
<td>209</td>
</tr>
<tr>
<td>60</td>
<td>XOR: Exclusive OR function</td>
<td>211</td>
</tr>
</tbody>
</table>
## AND: AND function

<table>
<thead>
<tr>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Function description</strong></td>
<td>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. <strong>EN</strong> and <strong>ENO</strong> can be configured as additional parameters.</td>
</tr>
</tbody>
</table>
| **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:  
```
<table>
<thead>
<tr>
<th>Value_1</th>
<th>IN1</th>
<th>OUT</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value_2</td>
<td>IN2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

AND

Representation in LD

Representation:

\[
\begin{array}{c|c|c|c|c}
& \text{EN} & \text{ENO} & \text{IN1} & \text{IN2} \\
\hline
\text{Value}_1 & & & \text{OUT} & \text{Result} \\
\text{Value}_2 & & & & \\
\end{array}
\]

Representation in IL

LD Value_1
AND Value_2
ST Result

Representation in ST

Result := AND (Value_1, Value_2) ;

Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value_1</td>
<td>BOOL, BYTE, WORD, DWORD</td>
<td>Input bit sequence</td>
</tr>
<tr>
<td>Value_2</td>
<td>BOOL, BYTE, WORD, DWORD</td>
<td>Input bit sequence</td>
</tr>
<tr>
<td>Value_n</td>
<td>BOOL, BYTE, WORD, DWORD</td>
<td>Input bit sequence (n = max. 32)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result</td>
<td>BOOL, BYTE, WORD, DWORD</td>
<td>Output bit sequence</td>
</tr>
</tbody>
</table>
F_TRIG: Falling edge detection

**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:

```
F_TRIG_Instance

ClockInput       CLK   Q    Output
```

**Representation in LD**

```
F_TRIG_Instance

EN  F_TRIG  ENO

ClockInput   CLK       Q    Output
```

**Representation in IL**

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

**Representation in ST**

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

Description of the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>BOOL</td>
<td>Clock input</td>
</tr>
</tbody>
</table>

Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>BOOL</td>
<td>Output</td>
</tr>
</tbody>
</table>
**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:

```
Start_Button IN FE OUT Start_Pulse
```

### Representation in LD

Representation:

```
Start_Button Start_Pulse
```

### 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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start_Button</td>
<td>EBOOL</td>
<td>Discrete input or output or internal bit whose Falling Edge we wish to detect.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start_Pulse</td>
<td>BOOL</td>
<td>Internal bit or output representing the Falling Edge.</td>
</tr>
</tbody>
</table>

Trend diagram

Timing diagram:

Start_Button

Start_Pulse

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

**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**

\[ \text{OUT} = \text{NOT} \ \text{IN} \]

**Representation in FBD**

```
        NOT
       IN   OUT
Value    NegValue
```

**Representation in LD**

```
        NOT
       EN   ENO
       IN   OUT
Value    NegValue
```
NOT

**Representation in IL**

Representation:
LD Value
NOT
ST NegValue

**Representation in ST**

Representation:
NegValue := NOT (Value) ;

**Parameter description**

Description of the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>BOOL, BYTE, WORD, DWORD</td>
<td>Input bit sequence</td>
</tr>
</tbody>
</table>

Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>NegValue</td>
<td>BOOL, BYTE, WORD, DWORD</td>
<td>Negated bit sequence</td>
</tr>
</tbody>
</table>
OR: OR function

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

\[ \text{OUT} = \text{IN}_1 \text{ OR } \text{IN}_2 \text{ OR } .. \text{ OR } \text{IN}_n \]

Representation in FBD

<table>
<thead>
<tr>
<th>Value_1</th>
<th>Value_2</th>
<th>IN1</th>
<th>IN2</th>
<th>OUT</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
OR

**Representation in LD**

Representation:

```
  OR
 EN  ENO
  Value_1 IN1  OUT  Result
  Value_2 IN2
```

**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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value_1</td>
<td>BOOL, BYTE, WORD, DWORD</td>
<td>Input bit sequence</td>
</tr>
<tr>
<td>Value_2</td>
<td>BOOL, BYTE, WORD, DWORD</td>
<td>Input bit sequence</td>
</tr>
<tr>
<td>Value_n</td>
<td>BOOL, BYTE, WORD, DWORD</td>
<td>Input bit sequence</td>
</tr>
<tr>
<td></td>
<td>n = max. 32</td>
<td></td>
</tr>
</tbody>
</table>

Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result</td>
<td>BOOL, BYTE, WORD, DWORD</td>
<td>Output bit sequence</td>
</tr>
</tbody>
</table>
R_TRIG: Rising edge detection

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

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

Representation in LD

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

Representation in IL

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

Representation in ST

```
R_TRIG_Instance (CLK:=ClockInput, Q=>Output)
```
### Parameter description

Description of the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>BOOL</td>
<td>Clock input</td>
</tr>
</tbody>
</table>

Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>BOOL</td>
<td>Output</td>
</tr>
</tbody>
</table>
**RE: Detection of Rising Edge**

**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:

```
  RE
IN  OUT
Start_Button  Start_Pulse
```

**Representation in LD**
Representation:
```
LD Start_Button
RE
ST Start_Pulse
```

**Representation in IL**
Representation:
```
Start_Pulse := RE (Start_Button);
```
Description of parameters

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start_Button</td>
<td>EBOOL</td>
<td>Discrete input or output, internal bit whose Rising Edge we wish to detect</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start_Pulse</td>
<td>BOOL</td>
<td>Internal bit or output representing the Rising Edge.</td>
</tr>
</tbody>
</table>

Trend diagram

Timing diagram:

Start_Button

Start_Pulse

T is the Start_Button update delay

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

<table>
<thead>
<tr>
<th>Description</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Function description</strong></td>
<td>The <strong>RESET</strong> function sets the bit associated with it to zero.</td>
<td></td>
</tr>
<tr>
<td><strong>Representation in FBD</strong></td>
<td>Representation:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="FBD Diagram" /></td>
<td></td>
</tr>
<tr>
<td><strong>Representation in LD</strong></td>
<td>Representation:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="LD Diagram" /></td>
<td></td>
</tr>
<tr>
<td><strong>Representation in IL</strong></td>
<td>Representation:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>CAL RESET(Bit_to_Reset)</code></td>
<td></td>
</tr>
<tr>
<td><strong>Representation in ST</strong></td>
<td>Representation:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>RESET (Bit_to_Reset);</code></td>
<td></td>
</tr>
</tbody>
</table>
The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit_to_Reset</td>
<td>BOOL</td>
<td>Discrete input or output or internal bit we wish to set to 0.</td>
</tr>
</tbody>
</table>
**ROL: Rotate left**

**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:

```
ROL
```

InputPattern

IN

OUT

Number

OutputPattern
ROL

**Representation in LD**

<table>
<thead>
<tr>
<th>ROL</th>
<th>EN</th>
<th>ENO</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>OUT</td>
<td>OutputPattern</td>
</tr>
<tr>
<td>N</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Representation in IL**

LD InputPattern
ROL Number
ST OutputPattern

**Representation in ST**

OutputPattern := ROL (InputPattern, Number) ;

**Parameter description**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| 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:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| OutputPattern | For ROL: BOOL, BYTE, WORD, DWORD  
For ROL_INT: INT  
For ROL_DINT: DINT | this is the bit pattern rotated |
ROR: Rotate right

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:

```
IN  OUT
Number
```

ROR
Representation in LD

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>InputPattern</td>
<td></td>
<td>this is the bit pattern to be rotated</td>
</tr>
<tr>
<td>Number</td>
<td></td>
<td>this is the number of spaces to be rotated</td>
</tr>
</tbody>
</table>

Representation in IL

LD InputPattern
ROR Number
ST OutputPattern

Representation in ST

OutputPattern := ROR (InputPattern, Number) ;

Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>InputPattern</td>
<td>BOOL, BYTE, WORD, DWORD</td>
<td>this is the bit pattern to be rotated</td>
</tr>
<tr>
<td>Number</td>
<td>UINT, INT, DINT</td>
<td>this is the number of spaces to be rotated</td>
</tr>
</tbody>
</table>

Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>OutputPattern</td>
<td>BOOL, BYTE, WORD, DWORD</td>
<td>this is the bit pattern rotated</td>
</tr>
</tbody>
</table>
RS: Bistable function block, reset dominant

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:

```
RS_Instance
Set S
Reset R1
RS
Q1
Output
```

Representation in LD

Representation:

```
RS_Instance

Set S
Reset R1
RS
EN
ENO

Q1
Output
```

Representation in IL

Representation:

CAL RS_Instance (S:=Set, R1:=Reset, Q1=>Output)
Representation in ST
RS_Instance (S:=Set, R1:=Reset, Q1=>Output) ;

Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>BOOL</td>
<td>Set</td>
</tr>
<tr>
<td>R1</td>
<td>BOOL</td>
<td>Reset (dominant)</td>
</tr>
</tbody>
</table>

Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>BOOL</td>
<td>Output</td>
</tr>
</tbody>
</table>
## SET: Setting of a bit to 1

<table>
<thead>
<tr>
<th>Description</th>
<th>Function description</th>
<th>Representation in FBD</th>
<th>Representation in LD</th>
<th>Representation in IL</th>
<th>Representation in ST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The <code>SET</code> function sets the bit associated with it to 1.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Representation in FBD

```
SET OUT Bit_to_Set
```

### Representation in LD

```
<table>
<thead>
<tr>
<th></th>
<th>Bit_to_Set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>S</code></td>
</tr>
</tbody>
</table>
```

### Representation in IL

```
CALL SET(Bit_to_Set)
```

### Representation in ST

```
SET (Bit_to_Set);
```
The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit_to_Set</td>
<td>BOOL</td>
<td>Discrete input or output or internal bit we wish to set to 1.</td>
</tr>
</tbody>
</table>
# SHL: Shift left

## Description

**Function description**

This function shifts the bit pattern at the \textit{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 \textit{IN} input and \textit{OUT} output must be identical.

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

\texttt{EN} and \texttt{ENO} can be configured as additional parameters.

## Further available functions

Additionally, the Obsolete library provides the following functions:

- \texttt{SHL\_DINT}
- \texttt{SHL\_INT}

The functionality of these functions is identical to the function \texttt{SHL}.

## Representation in FBD

**Representation:**

```
\begin{array}{|c|c|c|}
\hline
\text{InputPattern} & \text{IN} & \text{OUT} \\
\hline
\text{Number} & \text{N} & \text{ShiftedPattern} \\
\hline
\end{array}
```

---

---

---

---

---
SHL

Representation in LD

<table>
<thead>
<tr>
<th>Representation:</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHL</td>
</tr>
<tr>
<td>EN</td>
</tr>
<tr>
<td>ENO</td>
</tr>
<tr>
<td>InputPattern</td>
</tr>
<tr>
<td>IN</td>
</tr>
<tr>
<td>OUT</td>
</tr>
<tr>
<td>Number</td>
</tr>
<tr>
<td>N</td>
</tr>
</tbody>
</table>

Representation in IL

<table>
<thead>
<tr>
<th>Representation:</th>
</tr>
</thead>
<tbody>
<tr>
<td>LD InputPattern</td>
</tr>
<tr>
<td>SHL Number</td>
</tr>
<tr>
<td>ST ShiftedPattern</td>
</tr>
</tbody>
</table>

Representation in ST

<table>
<thead>
<tr>
<th>Representation:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShiftedPattern := SHL {InputPattern, Number} ;</td>
</tr>
</tbody>
</table>

Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>InputPattern</td>
<td>For SHL: BOOL, BYTE, WORD, DWORD</td>
<td>this is the bit pattern to be shifted</td>
</tr>
<tr>
<td></td>
<td>For SHL_INT: INT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For SHL_DINT: DINT</td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>For SHL: UINT</td>
<td>this is the number of spaces to be shifted</td>
</tr>
<tr>
<td></td>
<td>For SHL_INT, SHL_DINT: INT</td>
<td></td>
</tr>
</tbody>
</table>

Description of the input parameters:

For example: InputPattern = 2#0100000011110001.

Example: Number = 4.

Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShiftedPattern</td>
<td>For SHL: BOOL, BYTE, WORD, DWORD</td>
<td>this is the bit pattern shifted</td>
</tr>
<tr>
<td></td>
<td>For SHL_INT: INT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For SHL_DINT: DINT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For SHL: UINT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For SHL_INT, SHL_DINT: INT</td>
<td></td>
</tr>
</tbody>
</table>

For example: with the data from the previous table, the result is: ShiftedPattern = 2#0000111100010000.
**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:**

```
SHR
InputPattern ____ IN  OUT  ShiftedPattern
Number ___ N
```

Shr

Representation in LD

<table>
<thead>
<tr>
<th>Representation:</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHR</td>
</tr>
<tr>
<td>EN</td>
</tr>
<tr>
<td>ENO</td>
</tr>
<tr>
<td>InputPattern IN</td>
</tr>
<tr>
<td>OUT - ShiftedPattern N</td>
</tr>
</tbody>
</table>

Representation in IL

LD InputPattern
SHR Number
ST ShiftedPattern

Representation in ST

ShiftedPattern := SHR (InputPattern, Number) ;

Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>InputPattern</td>
<td>BOOL, BYTE, WORD, DWORD</td>
<td>this is the bit pattern to be shifted</td>
</tr>
<tr>
<td>Number</td>
<td>UINT</td>
<td>this is the number of spaces to be shifted</td>
</tr>
</tbody>
</table>

Example: InputPattern = 2#0100000011110001.

Example: Number = 4.

Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShiftedPattern</td>
<td>BOOL, BYTE, WORD, DWORD</td>
<td>this is the bit pattern shifted</td>
</tr>
</tbody>
</table>

Example: with the data from the previous table, the result is: ShiftedPattern = 2#0000010000001111.
SR: Bistable function block, set dominant

Description

Function description

The function block is used as SR memory with the property "Set dominant". Output $Q_1$ becomes "1" when the $S_1$ input becomes "1". This state remains even if input $S_1$ reverts back to "0". Output $Q_1$ changes back to "0" when input $R$ becomes "1". If the inputs $S_1$ and $R$ are both "1" simultaneously, the dominating input $S_1$ will set the output $Q_1$ to "1".

When the function block is called for the first time, the initial state of $Q_1$ is "0". $EN$ and $ENO$ can be configured as additional parameters.

Representation in FBD

```
SR_Instance

Set S1
Reset R

SR Q1 Output
```

Representation in LD

```
SR_Instance

EN
S1
R

Output
```

Representation in IL

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

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

Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>BOOL</td>
<td>Set (dominant)</td>
</tr>
<tr>
<td>R</td>
<td>BOOL</td>
<td>Reset</td>
</tr>
</tbody>
</table>

Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>BOOL</td>
<td>Output</td>
</tr>
</tbody>
</table>
TRIGGER: Detection of all edges

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:

```
TRIGGER_Instance

TRIGGER

ClockInput ——— CLK ——— RISE ——— RisingEdge

EDGE ——— AnyEdge

FALL ——— FallingEdge
```
### Representation in LD

**Representation:**

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

### 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:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>BOOL</td>
<td>Clock input</td>
</tr>
</tbody>
</table>

**Description of the output parameter:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RISE</td>
<td>BOOL</td>
<td>Indicator of a rising edge</td>
</tr>
<tr>
<td>EDGE</td>
<td>BOOL</td>
<td>Indicator of all types of edges</td>
</tr>
<tr>
<td>FALL</td>
<td>BOOL</td>
<td>Indicator of a falling edge</td>
</tr>
</tbody>
</table>
XOR: Exclusive OR function

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:

```
   XOR
   Value_1 IN1 OUT Result
   Value_2 IN2
```
XOR

Representation in LD

Representation:

![LD Diagram]

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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value_1</td>
<td>BOOL, BYTE, WORD, DWORD</td>
<td>Input bit sequence</td>
</tr>
<tr>
<td>Value_2</td>
<td>BOOL, BYTE, WORD, DWORD</td>
<td>Input bit sequence</td>
</tr>
<tr>
<td>Value_n</td>
<td>BOOL, BYTE, WORD, DWORD</td>
<td>Input bit sequence</td>
</tr>
<tr>
<td></td>
<td></td>
<td>n = max 32</td>
</tr>
</tbody>
</table>

Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result</td>
<td>BOOL, BYTE, WORD, DWORD</td>
<td>Output bit sequence</td>
</tr>
</tbody>
</table>
# Mathematics

## Introduction

## Overview

This section describes the elementary functions and elementary function blocks of the Mathematics family.
This part contains the following chapters:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Chapter Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>61</td>
<td>ABS: Absolute value computation</td>
<td>215</td>
</tr>
<tr>
<td>62</td>
<td>ACOS: Arc cosine</td>
<td>217</td>
</tr>
<tr>
<td>63</td>
<td>ADD: Addition</td>
<td>219</td>
</tr>
<tr>
<td>64</td>
<td>ADD_TIME: Addition</td>
<td>221</td>
</tr>
<tr>
<td>65</td>
<td>ASIN: Arc sine</td>
<td>223</td>
</tr>
<tr>
<td>66</td>
<td>ATAN: Arc tangent</td>
<td>225</td>
</tr>
<tr>
<td>67</td>
<td>COS: Cosine</td>
<td>227</td>
</tr>
<tr>
<td>68</td>
<td>DEC: Decrementation of a variable</td>
<td>229</td>
</tr>
<tr>
<td>69</td>
<td>DIV: Division</td>
<td>231</td>
</tr>
<tr>
<td>70</td>
<td>DIVMOD: Division and Modulo</td>
<td>233</td>
</tr>
<tr>
<td>71</td>
<td>EXP: Natural exponential</td>
<td>235</td>
</tr>
<tr>
<td>72</td>
<td>EXPT_REAL_*** : Exponentiation of one value by another value</td>
<td>237</td>
</tr>
<tr>
<td>73</td>
<td>INC: Incrementation of a variable</td>
<td>239</td>
</tr>
<tr>
<td>74</td>
<td>LN: Natural logarithm</td>
<td>241</td>
</tr>
<tr>
<td>75</td>
<td>LOG : Base 10 logarithm</td>
<td>243</td>
</tr>
<tr>
<td>76</td>
<td>MOD: Modulo</td>
<td>245</td>
</tr>
<tr>
<td>77</td>
<td>MOVE: Assignment</td>
<td>247</td>
</tr>
<tr>
<td>78</td>
<td>MUL: Multiplication</td>
<td>249</td>
</tr>
<tr>
<td>79</td>
<td>NEG: Negation</td>
<td>251</td>
</tr>
<tr>
<td>80</td>
<td>SIGN: Sign evaluation</td>
<td>253</td>
</tr>
<tr>
<td>81</td>
<td>SIN: Sine</td>
<td>255</td>
</tr>
<tr>
<td>82</td>
<td>SUB: Subtraction</td>
<td>257</td>
</tr>
<tr>
<td>83</td>
<td>SUB_TIME: Subtraction</td>
<td>259</td>
</tr>
<tr>
<td>84</td>
<td>SQRT_*** : Square root</td>
<td>261</td>
</tr>
<tr>
<td>85</td>
<td>TAN: Tangent</td>
<td>263</td>
</tr>
</tbody>
</table>
ABS: Absolute value computation

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

\[ \text{OUT} = |\text{IN}| \]

Representation in FBD

Representation:

```
     ABS
    ┌───────┐    ┌──┐    ┌──┐
    │      │    │ IN │    │ OUT │
    └──────┘    └──┘    └──┘
          Result
```

Representation in LD

Representation:

```
     ABS
    ┌──┐    ┌──┐    ┌──┐
    │ EN │    │ ENO │    ┌──┐
    └──┘    └──┘    └──┘
          Value
           ┌──┐    ┌──┐
           │ IN │    │ OUT │
           └──┘    └──┘
                 Result
```
ABS

**Representation in IL**

Representation:

LD Value
ABS
ST Result

**Representation in ST**

Representation:
Result := ABS (Value) ;

**Parameter description**

Description of the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>INT, DINT, UINT, UDINT, REAL</td>
<td>Input value</td>
</tr>
</tbody>
</table>

Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result</td>
<td>INT, DINT, UINT, UDINT, REAL</td>
<td>Output value</td>
</tr>
</tbody>
</table>

**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

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:

\[ \text{Angle} = \arccos(\text{Cos}_\text{Value}) \]

Representation in FBD

Representation:

```
<table>
<thead>
<tr>
<th>ACOS_REAL</th>
<th>IN</th>
<th>OUT</th>
<th>Angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cos_Value</td>
<td>IN</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
ACOS

### Representation in LD
Illustration:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cos_Value</td>
<td>REAL</td>
<td>Cosine of angle calculated at block output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1 ≤ Cos_Value ≤ 1</td>
</tr>
</tbody>
</table>

### 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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cos_Value</td>
<td>REAL</td>
<td>Cosine of angle calculated at block output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1 ≤ Cos_Value ≤ 1</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle</td>
<td>REAL</td>
<td>Angle expressed in radians, whose cosine has the value Cos_Value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 ≤ Angle ≤ π</td>
</tr>
</tbody>
</table>

### 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

**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**

\[ \text{OUT} = \text{IN}_1 + \text{IN}_2 + \ldots \text{IN}_n \]

**Representation in FBD**

<table>
<thead>
<tr>
<th>Value1</th>
<th>IN1</th>
<th>OUT</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value2</td>
<td>IN2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Representation in LD**

| EN    |  | ENO  |
|-------| |------|
| Value1 | IN1 | OUT  | Sum |
| Value2 | IN2  |      |     |
ADD

**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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value1</td>
<td>INT, DINT, UINT, UDINT, REAL</td>
<td>Summand</td>
</tr>
<tr>
<td>Value2</td>
<td>INT, DINT, UINT, UDINT, REAL</td>
<td>Summand</td>
</tr>
<tr>
<td>Valuen</td>
<td>INT, DINT, UINT, UDINT, REAL</td>
<td>Summand n = max 32</td>
</tr>
</tbody>
</table>

Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum</td>
<td>INT, DINT, UINT, UDINT, REAL</td>
<td>Sum</td>
</tr>
</tbody>
</table>

**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

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:

```
ADD_TIME
TimeValue1 IN1 OUT Sum
TimeValue2 IN2
```

Representation in LD

Representation:

```
ADD_TIME
EN ENO
TimeValue1 IN1 OUT Sum
TimeValue2 IN2
```

Representation in IL

Representation:

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

**Representation in ST**

Representation:

\[ \text{Sum} := \text{ADD\_TIME} (\text{TimeValue1, TimeValue2}) ; \]

**Parameter description**

Description of the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>TimeValue1</td>
<td>TIME</td>
<td>Summand</td>
</tr>
<tr>
<td>TimeValue2</td>
<td>TIME</td>
<td>Summand</td>
</tr>
</tbody>
</table>

Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum</td>
<td>TIME</td>
<td>Sum</td>
</tr>
</tbody>
</table>

**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

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:

\[ \text{Angle} = \arcsin(\text{Sin}_\text{Value}) \]

Representation in FBD

```
\begin{array}{|c|c|}
\hline
\text{Sin}_\text{Value} & \text{ASIN}_\text{REAL} \\
\hline
\text{IN} & \text{OUT} \\
\hline
\text{Angle} \\
\hline
\end{array}
```

Representation in LD

```
\begin{array}{|c|c|c|c|}
\hline
\text{Sin}_\text{Value} & \text{ASIN}_\text{REAL} \\
\hline
\text{EN} & \text{ENO} \\
\hline
\text{IN} & \text{OUT} \\
\hline
\text{Angle} \\
\hline
\end{array}
```
**ASIN**

**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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sin_Value</td>
<td>REAL</td>
<td>Sine of angle calculated at block output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1 ≤ Sin_Value ≤ 1</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle</td>
<td>REAL</td>
<td>Angle expressed in radians, whose sine has the value Sin_Value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-π/2 ≤ Angle ≤ +π/2</td>
</tr>
</tbody>
</table>

**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

**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:

\[ \text{Angle} = \arctan(\text{Tan\_Value}) \]

**Representation in FBD**

Representation:

```
ATAN_REAL
IN       OUT
Tan_Value Angle
```

**Representation in LD**

Representation:

```
ATAN_REAL
EN       ENO
IN       OUT
Tan_Value Angle
```
ATAN

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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tan_Value</td>
<td>REAL</td>
<td>Tangent of angle calculated at block output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1.#INF &lt; Tan_Value &lt; +1.#INF</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle</td>
<td>REAL</td>
<td>Angle expressed in radians, whose tangent has the value Tan_Value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- π/2 &lt; Angle &lt; +π/2</td>
</tr>
</tbody>
</table>

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**

**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:

\[ \text{Cos}_\text{Value} = \cos(\text{Angle}) \]

**Representation in FBD**

Representation:

```
COS_REAL

IN  OUT
Angle  Cos_Value
```

**Representation in LD**

Representation:

```
COS_REAL

EN  ENO
IN  OUT
Angle  Cos_Value
```
COS

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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle</td>
<td>REAL</td>
<td>Angle expressed in radians.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$-2^{63} \leq \text{Angle} \leq 2^{63}$</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cos_Value</td>
<td>REAL</td>
<td>Cosine of Angle expressed in radians.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$-1 \leq \text{Cos_Value} \leq 1$</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Function description</strong></td>
</tr>
<tr>
<td>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.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Representation in FBD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Representation applied to an integer:</td>
</tr>
<tr>
<td><img src="image1" alt="FBD Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Representation in LD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Representation applied to an integer:</td>
</tr>
<tr>
<td><img src="image2" alt="LD Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Representation in IL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Representation applied to an integer:</td>
</tr>
<tr>
<td>CAL DEC(Value1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Representation in ST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Representation applied to an integer:</td>
</tr>
<tr>
<td>DEC(Value1);</td>
</tr>
</tbody>
</table>
The following table describes the input/output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value1</td>
<td>INT, DINT, UINT, UDINT</td>
<td>Each time the program uses this EF the variable Value1 is decremented by one unit.</td>
</tr>
</tbody>
</table>

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

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

\[ \text{OUT} = \left( \text{IN1} \div \text{IN2} \right) \]

Representation in FBD

<table>
<thead>
<tr>
<th>Dividend</th>
<th>Divisor</th>
<th>OUT</th>
<th>Quotient</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN1</td>
<td>IN2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
DIV

**Representation in LD**

```
<table>
<thead>
<tr>
<th>EN</th>
<th>ENO</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN1</td>
<td>OUT</td>
</tr>
<tr>
<td>IN2</td>
<td></td>
</tr>
</tbody>
</table>
```

**Representation in IL**

- LD Dividend
- DIV Divisor
- ST Quotient

**Representation in ST**

Quotient := DIV (Dividend, Divisor);

**Parameter description**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dividend</td>
<td>INT, DINT, UINT, UDINT, REAL</td>
<td>Dividend</td>
</tr>
<tr>
<td>Divisor</td>
<td>INT, DINT, UINT, UDINT, REAL</td>
<td>Divisor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quotient</td>
<td>INT, DINT, UINT, UDINT, REAL</td>
<td>Quotient</td>
</tr>
</tbody>
</table>

**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

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:

\[ DV = \frac{IN1}{IN2} \]
\[ MD = IN1 \mod IN2 \]

Representation in FBD

Representation:

```
DIVMOD
Dividend IN1 DV Quotient
Divisor IN2 MD Modulo
```  

Representation in LD

Representation:

```
DIVMOD
EN ENO
Dividend IN1 DV Quotient
Divisor IN2 MD Modulo
```
DIVMOD

**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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dividend</td>
<td>INT, DINT, UINT, UDINT</td>
<td>Dividend</td>
</tr>
<tr>
<td>Divisor</td>
<td>INT, DINT, UINT, UDINT</td>
<td>Divisor</td>
</tr>
</tbody>
</table>

Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quotient</td>
<td>INT, DINT, UINT, UDINT</td>
<td>Quotient</td>
</tr>
<tr>
<td>Modulo</td>
<td>INT, DINT, UINT, UDINT</td>
<td>Modulo</td>
</tr>
</tbody>
</table>

**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.
**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:

\[ \text{Exp}_\text{Real}_\text{Value} = \exp(\text{Real}_\text{Value}) \]

**Representation in FBD**

Representation:

![FBD Diagram]

**Representation in LD**

Representation:

![LD Diagram]
**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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real_Value</td>
<td>REAL</td>
<td>Real value of which we wish to obtain the Natural exponential</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-87.33654 &lt; Real_Value &lt; 88.72283</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exp_Real_Value</td>
<td>REAL</td>
<td>Natural exponential of Real_Value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 &lt; Exp_Real_Value &lt; 1.#INF</td>
</tr>
</tbody>
</table>

**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

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:

\[ \text{Expt}_\text{Real}_\text{Value} = \text{Value}_1^{\text{Exponent}} \]

Representation in FBD
Representation applied to a real number:

```
    | EXPT_REAL_REAL |
----|----------------|
IN1 | Value1         |
IN2 | Exponent      |
OUT | Expt_Real_Value |
```
Representation in LD

Representation applied to a real number:

```
EXPT_REAL_REAL
EN       ENO
IN1      OUT
Value1   Expt_Real_Value
Exponent IN2
```

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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value1</td>
<td>REAL</td>
<td>Value for which you want to find the exponential by Exponent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$0 \leq \text{Value1} &lt; \infty$</td>
</tr>
<tr>
<td>Exponent</td>
<td>INT, UINT,</td>
<td>Exponent of the exponential</td>
</tr>
<tr>
<td></td>
<td>DINT, UDINT,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$-\infty &lt; \text{Exponent} &lt; +\infty$</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expt_Real_Value</td>
<td>REAL</td>
<td>Natural exponential of Value1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$-1 &lt; \text{Expt_Real_Value} &lt; +\infty$</td>
</tr>
</tbody>
</table>

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

### 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:

```
Value1   INC
        INOUT INOUT
```

### Representation in LD

Representation applied to an integer:

```
Value1   INC
        EN ENO
        INOUT INOUT
```

### Representation in IL

Representation applied to an integer:

```
CAL INC(Value1)
```

### Representation in ST

Representation applied to an integer:

```
INC(Value1);
```
The following table describes the input/output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value1</td>
<td>INT, DINT, UINT, UDINT</td>
<td>Each time the program uses this EF, the variable Value1 is incremented by one unit.</td>
</tr>
</tbody>
</table>

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**

**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:

\[ \text{Ln}_\text{Real}_\text{Value} = \ln(\text{Real}_\text{Value}) \]

**Representation in FBD**

Representation:

```
  | Real_Value  | LN_REAL        |
  | IN          | OUT            |
  | Ln_Real_Value|
```

**Representation in LD**

Representation:

```
  | Real_Value  | LN_REAL        |
  | IN          | EN             |
  |             | ENO            |
  |             | OUT            |
  | Lan_Real_Value|
```
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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real_Value</td>
<td>REAL</td>
<td>Real value of which we wish to obtain the natural logarithm.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 &lt; Real_Value &lt; 1.#INF</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ln_Real_Value</td>
<td>REAL</td>
<td>Natural logarithm of Real_Value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1.#INF &lt; Ln_Real_Value &lt; +1.#INF</td>
</tr>
</tbody>
</table>

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**

**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:

\[ \text{Log}_x \text{Real}_x \text{Value} = \text{Log}(\text{Real}_x \text{Value}) \]

**Representation in FBD**

Representation:

```
LOG_REAL
IN OUT Log_Real_Value
Real_Value
```

**Representation in LD**

Representation:

```
LOG_REAL
EN ENO Log_Real_Value
Real_Value
```

243
LOG

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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real_Value</td>
<td>REAL</td>
<td>Real value of which we wish to obtain the natural logarithm. 0 &lt; Real_Value &lt; 1.#INF</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log_Real_Value</td>
<td>REAL</td>
<td>Natural logarithm of Real_Value -1.#INF &lt; Log_Real_Value &lt; +1.#INF</td>
</tr>
</tbody>
</table>

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

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

\[ \text{OUT} = \text{IN}_1 \mod \text{IN}_2 \]

Representation in FBD

Representation:

```
MOD
Dividend \rightarrow IN1 \rightarrow OUT \rightarrow Rest
Divisor \rightarrow IN2
```

Representation in LD

Representation:

```
MOD
EN \rightarrow ENO
Dividend \rightarrow IN1 \rightarrow OUT \rightarrow Rest
Divisor \rightarrow IN2
```

Representation in IL

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

Representation in ST

Representation:

\[
\text{Rest} := \text{MOD} (\text{Dividend}, \text{Divisor}) ;
\]

Parameter description

Description of the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dividend</td>
<td>INT, DINT, UINT, UDINT</td>
<td>Dividend</td>
</tr>
<tr>
<td>Divisor</td>
<td>INT, DINT, UINT, UDINT</td>
<td>Divisor</td>
</tr>
</tbody>
</table>

Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remainder</td>
<td>INT, DINT, UINT, UDINT</td>
<td>Modulo</td>
</tr>
</tbody>
</table>

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

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

\[ \text{OUT} = \text{IN} \]

Representation in FBD

Representation:

```
  MOVE
  IN     OUT
```

Representation in LD

This function cannot 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:

```
  MOVE
  EN    ENO
  IN    OUT
```


MOVE

**Representation in IL**

Representation:
LD Input
MOVE
ST Output

**Representation in ST**

Representation:
Output := MOVE (Input) ;

**Parameter description**

**Description of the input parameter:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>ANY</td>
<td>Input value</td>
</tr>
</tbody>
</table>

**Description of the output parameter:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>ANY</td>
<td>Output value</td>
</tr>
</tbody>
</table>
MUL: Multiplication

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 * IN2 * ... * IN_n

Representation in FBD
Representation:

```
MUL
Factor1 IN1 OUT Product
Factor2 IN2
```

Representation in LD
Representation:

```
MUL
EN ENO
Factor1 IN1 OUT Product
Factor2 IN2
```
### 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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factor1</td>
<td>INT, DINT, UINT, UDINT, REAL</td>
<td>Multiplicand (factor)</td>
</tr>
<tr>
<td>Factor2</td>
<td>INT, DINT, UINT, UDINT, REAL</td>
<td>Multiplier (factor)</td>
</tr>
<tr>
<td>Factorn</td>
<td>INT, DINT, UINT, UDINT, REAL</td>
<td>Multiplier (factor)</td>
</tr>
</tbody>
</table>

#### Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>INT, DINT, UINT, UDINT, REAL</td>
<td>Product</td>
</tr>
</tbody>
</table>

### 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

Description

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

- $6 \rightarrow -6$
- $-4 \rightarrow 4$

**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

<table>
<thead>
<tr>
<th>Input</th>
<th>IN</th>
<th>OUT</th>
<th>NegatedOutput</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEG</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Representation in LD

Representation:

```
  NEG
  EN  ENO
  IN  OUT  NegatedOutput
```

Representation in IL

Representation:

```
LD Input
NEG
ST NegatedOutput
```

Representation in ST

Representation:

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

Parameter description

Description of the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>INT, DINT, UINT, UDINT, REAL</td>
<td>Input</td>
</tr>
</tbody>
</table>

Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>NegatedOut</td>
<td>INT, DINT, UINT, UDINT, REAL</td>
<td>Negated output</td>
</tr>
</tbody>
</table>

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)
- an input value of the data type UDINT or UINT is to be converted.
## SIGN: Sign evaluation

### Description

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

### Formula

Block formula:

- $\text{OUT} = 1$, if $\text{IN} < 0$
- $\text{OUT} = 0$, if $\text{IN} \geq 0$

**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.

### Representation in FBD

<table>
<thead>
<tr>
<th>Value</th>
<th>SIGN</th>
<th>OUT</th>
<th>Negativ</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SIGN:

Representation in LD

Representation:

```
  SIGN
  EN
  Value
  IN
  OUT
  ENO
```

Representation in IL

Representation:

- LD Value
- SIGN
- ST Negativ

Representation in ST

Representation:

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

Parameter description

Description of input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>INT, DINT, REAL</td>
<td>Signed input</td>
</tr>
</tbody>
</table>

Description of output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negative</td>
<td>BOOL</td>
<td>Sign evaluation</td>
</tr>
</tbody>
</table>

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

**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:

\[
\text{Sin\_Value} = \sin(\text{Angle})
\]

**Representation in FBD**

Representation:

```
SIN_REAL

Angle IN  OUT  Sin_Value
```

**Representation in LD**

Representation:

```
SIN_REAL

EN  ENO

Angle IN  OUT  Sin_Value
```
SIN

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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle</td>
<td>REAL</td>
<td>Angle expressed in radians.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-2^{63} \leq \text{Angle} \leq +2^{63}</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sin_Value</td>
<td>REAL</td>
<td>Sine of Angle.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1 \leq \text{Sin_Value} \leq 1</td>
</tr>
</tbody>
</table>

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

Description

Function description
The function subtracts the value at the \texttt{Value2} input from the value at the \texttt{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 \texttt{TIME} data type, you can use the block \texttt{SUB\_TIME} (See \texttt{SUB\_TIME: Subtraction, p. 259}). \texttt{EN} and \texttt{ENO} can be configured as additional parameters.

Formula

\[ \text{Difference} = \text{Value1} - \text{Value2} \]

Representation in FBD

\[
\begin{array}{c}
\text{SUB} \\
\text{IN1} \quad \text{OUT} \\
\text{IN2} \\
\end{array}
\]

\[\text{Value1} \quad \text{Value2} \quad \text{Difference} \]

Representation in LD

\[
\begin{array}{c}
\text{SUB} \\
\text{EN} \quad \text{ENO} \\
\text{IN1} \quad \text{OUT} \\
\text{IN2} \\
\end{array}
\]

\[\text{Value1} \quad \text{Value2} \quad \text{Difference} \]
SUB

**Representation in IL**

 Representation:
LD Value1
SUB Value2
ST Difference

**Representation in ST**

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

**Parameter description**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value1</td>
<td>INT, DINT, UINT, UDINT, REAL</td>
<td>Minuend</td>
</tr>
<tr>
<td>Value2</td>
<td>INT, DINT, UINT, UDINT, REAL</td>
<td>Subtrahend</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difference</td>
<td>INT, DINT, UINT, UDINT, REAL</td>
<td>Difference</td>
</tr>
</tbody>
</table>

**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

### 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

<table>
<thead>
<tr>
<th>SUB_TIME</th>
<th>IN1</th>
<th>OUT</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TimeValue1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TimeValue2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Representation in LD

<table>
<thead>
<tr>
<th>SUB_TIME</th>
<th>EN</th>
<th>ENO</th>
</tr>
</thead>
<tbody>
<tr>
<td>TimeValue1</td>
<td>IN1</td>
<td>OUT</td>
</tr>
<tr>
<td>TimeValue2</td>
<td>IN2</td>
<td></td>
</tr>
</tbody>
</table>

### Representation in IL

```LD
LD TimeValue1
SUB TimeValue2
ST Difference
```
Representation in ST

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

Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>TimeValue1</td>
<td>TIME</td>
<td>Minuend</td>
</tr>
<tr>
<td>TimeValue2</td>
<td>TIME</td>
<td>Subtrahend</td>
</tr>
</tbody>
</table>

Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difference</td>
<td>TIME</td>
<td>Difference</td>
</tr>
</tbody>
</table>

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
### 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:

```
Value1  SQRT_REAL  IN    OUT  Sqrt_Value1
```

### Representation in LD

Representation applied to an integer:

```
Value1  EN    ENO

SQRT_REAL  IN    OUT  Sqrt_Value1
```
**SQRT_***

**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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value1</td>
<td>INT, DINT, REAL</td>
<td>Variable whose square root you want to extract. 0 ≤ Value1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sqrt_Value1</td>
<td>INT, DINT, REAL</td>
<td>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.</td>
</tr>
</tbody>
</table>

**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*).

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*).
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:

\[ \text{Tan}_\text{Value} = \tan(\text{Angle}) \]

Representation in FBD

Representation:

```
IN  OUT
Angle  TAN_REAL  Tan_Value
```

Representation in LD

Representation:

```
EN  ENO
IN  OUT
Angle  TAN_REAL  EN  ENO  Tan_Value
```

263
### Representation in IL

**Representation:**

- `LD Angle`
- `TAN_REAL`
- `ST Tan_Value`

### Representation in ST

**Representation:**

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

### Description of parameters

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle</td>
<td>REAL</td>
<td>Angle expressed in radians.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-2^63 ≤ Angle ≤ +2^63</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tan_Value</td>
<td>REAL</td>
<td>Tangent of Angle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1.#INF &lt; Tan_Value &lt; +1.#INF</td>
</tr>
</tbody>
</table>

### 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.
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:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Chapter Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>86</td>
<td>AVE: Averaging</td>
<td>267</td>
</tr>
<tr>
<td>87</td>
<td>LIMIT: Limit</td>
<td>271</td>
</tr>
<tr>
<td>88</td>
<td>LIMIT_IND: Limit with indicator</td>
<td>275</td>
</tr>
<tr>
<td>89</td>
<td>MAX: Maximum value function</td>
<td>279</td>
</tr>
<tr>
<td>90</td>
<td>MIN: Minimum value function</td>
<td>281</td>
</tr>
<tr>
<td>91</td>
<td>MUX: Multiplexer</td>
<td>283</td>
</tr>
<tr>
<td>92</td>
<td>SEL: Binary selection</td>
<td>287</td>
</tr>
</tbody>
</table>
**AVE: Averaging**

**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:

\[
Y = \frac{\sum (K_i \times X_i)}{\sum (K_i)}
\]

**Representation in FBD**

Representation:

```
    AVE
   /     |
  /      |
 FactorFirstValue -> K_X1    Y  Result
    |
 FirstValue   -> K_X2
    |
 FactorSecondValue -> K_X3
    |
 SecondValue   -> K_X4
```

Representation in LD

Representation:

\[
\begin{array}{c}
\text{EN} \quad \text{ENO} \\
\text{FactorFirstValue} \quad K_X1 \\
\text{FirstValue} \quad K_X2 \\
\text{FactorSecondValue} \quad K_X3 \\
\text{SecondValue} \quad K_X4 \\
\end{array}
\]

Representation in IL

LD FactorFirstValue
AVE FirstValue, FactorSecondValue, SecondValue
ST Result

Representation in ST

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

Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>FactorFirstValue</td>
<td>INT, DINT, UINT, UDINT, REAL</td>
<td>Factor (K1) for first value</td>
</tr>
<tr>
<td>FirstValue</td>
<td>INT, DINT, UINT, UDINT, REAL</td>
<td>First value (X1)</td>
</tr>
<tr>
<td>FactorSecondValue</td>
<td>INT, DINT, UINT, UDINT, REAL</td>
<td>Factor (K2) for second value</td>
</tr>
<tr>
<td>SecondValue</td>
<td>INT, DINT, UINT, UDINT, REAL</td>
<td>Second value (X2)</td>
</tr>
<tr>
<td>( n )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Factor HalfnValue</td>
<td>INT, DINT, UINT, UDINT, REAL</td>
<td>Factor for value ( 2^n ) of (K_X(n-1))</td>
</tr>
<tr>
<td>HalfnValue</td>
<td>INT, DINT, UINT, UDINT, REAL</td>
<td>Value ( 2^n ) of (K_X(n)) ( n = \text{max} 32 )</td>
</tr>
</tbody>
</table>
Description of output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result</td>
<td>INT, DINT, UINT, UDINT, REAL</td>
<td>Average value</td>
</tr>
</tbody>
</table>

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

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**

\[
\text{OUT} = \text{IN}, \text{if } (\text{IN} \geq \text{MN}) \land (\text{IN} \leq \text{MX})
\]

\[
\text{OUT} = \text{MN}, \text{if } (\text{IN} < \text{MN})
\]

\[
\text{OUT} = \text{MX}, \text{if } (\text{IN} > \text{MX})
\]

**Representation in FBD**

```
Limit
  +------------------+
  | LowerLimit       |
  | MN               |
  | Input            |
  | IN               |
  | UpperLimit       |
  | MX               |
  +------------------+
```

OUT = Output
**LIMIT**

<table>
<thead>
<tr>
<th>Representation in LD</th>
<th>Representation:</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Diagram" /></td>
<td>Output := LIMIT (UpperLimit, Input, LowerLimit) ;</td>
</tr>
</tbody>
</table>

**Representation in IL**

LD UpperLimit
LIMIT Input, LowerLimit
ST Output

**Representation in ST**

Output := LIMIT (UpperLimit, Input, LowerLimit) ;
## Parameter description

### Description of the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>LowerLimit</td>
<td>BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME</td>
<td>lower limit</td>
</tr>
<tr>
<td>Input</td>
<td>BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME</td>
<td>Input</td>
</tr>
<tr>
<td>UpperLimit</td>
<td>BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME</td>
<td>upper limit</td>
</tr>
</tbody>
</table>

### Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME</td>
<td>Output</td>
</tr>
</tbody>
</table>

### Runtime error

If there is an unauthorized floating point number at the input, an error message is returned.
**LIMIT_IND: Limit with indicator**

### 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:**

\[
\text{OUT} = \begin{cases} 
\text{IN}, & \text{if } (\text{IN} \leq \text{MX}) \& \text{IN} \geq \text{MN} \\
\text{MN}, & \text{if } (\text{IN} < \text{MN}) \\
\text{MX}, & \text{if } (\text{IN} > \text{MX}) 
\end{cases}
\]

\[
\text{MN\_IND} = \begin{cases} 
0, & \text{if } \text{IN} \geq \text{MN} \\
1, & \text{if } \text{IN} < \text{MN}
\end{cases}
\]

\[
\text{MX\_IND} = \begin{cases} 
0, & \text{if } \text{IN} \leq \text{MX} \\
1, & \text{if } \text{IN} > \text{MX}
\end{cases}
\]
LIMIT_IND

**Representation in FBD**

<table>
<thead>
<tr>
<th>LIMIT_IND</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LimitMinimum</td>
<td>MN</td>
</tr>
<tr>
<td>Input</td>
<td>IN</td>
</tr>
<tr>
<td>LimitMaximum</td>
<td>MX</td>
</tr>
</tbody>
</table>

**Representation in LD**

<table>
<thead>
<tr>
<th>LIMIT_IND</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LimitMinimum</td>
<td>MN</td>
</tr>
<tr>
<td>Input</td>
<td>IN</td>
</tr>
<tr>
<td>LimitMaximum</td>
<td>MX</td>
</tr>
</tbody>
</table>

**Representation in IL**

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

**Representation in ST**

```
LIMIT_IND (LimitMinimum, Input, LimitMaximum,
      MinimumViolation, Output, MaximumViolation);
```
### Description of the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limit Minimum</td>
<td>BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME</td>
<td>Limit of minimum value</td>
</tr>
<tr>
<td>Input</td>
<td>BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME</td>
<td>Input</td>
</tr>
<tr>
<td>Limit Maximum</td>
<td>BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME</td>
<td>Limit of maximum value</td>
</tr>
</tbody>
</table>

### Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Violation</td>
<td>BOOL</td>
<td>Display of minimum value violation</td>
</tr>
<tr>
<td>Output</td>
<td>BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME</td>
<td>Output</td>
</tr>
<tr>
<td>Maximum Violation</td>
<td>BOOL</td>
<td>Display of maximum value violation</td>
</tr>
</tbody>
</table>
**MAX: Maximum value function**

**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**

\[ \text{OUT} = \text{MAX} \{ \text{IN1}, \text{IN2}, \ldots, \text{INn} \} \]

**Representation in FBD**

- MAX
- Value1 \(\rightarrow\) IN1
- Value2 \(\rightarrow\) IN2
- OUT
- Maximum

**Representation in LD**

- MAX
- EN \(\rightarrow\) IO
- Value1 \(\rightarrow\) IN1
- Value2 \(\rightarrow\) IN2
- OUT
- Maximum

**Representation in IL**

- LD Value1
- MAX Value2
- ST Maximum
MAX

Representation in ST

Maximum := MAX (Value1, Value2) ;

Parameter description

Description of input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value1</td>
<td>BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME</td>
<td>1. Input value</td>
</tr>
<tr>
<td>Value2</td>
<td>BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME</td>
<td>2. Input value</td>
</tr>
<tr>
<td>Valuen</td>
<td>BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME</td>
<td>n. Input value (n = \text{max} 32)</td>
</tr>
</tbody>
</table>

Description of output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum</td>
<td>BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME</td>
<td>Maximum value</td>
</tr>
</tbody>
</table>

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

**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**

\[ \text{OUT} = \text{MIN}\{\text{IN1, IN2, \ldots, INn}\} \]

**Representation in FBD**

[Diagram of FBD representation]

**Representation in LD**

[Diagram of LD representation]

**Representation in IL**

LD Value1
MIN Value2
ST Minimum
MIN

Representation in ST

Minimum := MIN (Value1, Value2) ;

Parameter description

Description of input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value1</td>
<td>BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME</td>
<td>1. Input value</td>
</tr>
<tr>
<td>Value2</td>
<td>BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME</td>
<td>2. Input value</td>
</tr>
</tbody>
</table>
| Valuen    | BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME | n. Input value  
  n = max 32 |

Description of output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME</td>
<td>Minimum value</td>
</tr>
</tbody>
</table>

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

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**

```
<table>
<thead>
<tr>
<th>Input0</th>
<th>IN0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input1</td>
<td>IN1</td>
</tr>
<tr>
<td>Selection</td>
<td>K</td>
</tr>
<tr>
<td>MUX</td>
<td>OUT</td>
</tr>
</tbody>
</table>
```

Output
### MUX

#### Representation in LD

**Representation:**

![MUX Diagram](image)

#### Representation in IL

Representation:

LD Selection
MUX Input0, Input1
ST Output

#### Representation in ST

Representation:

Output := MUX (Selection, Input0, Input1) ;
Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>INT, DINT, UINT, UDINT</td>
<td>Selection input K = 0...30</td>
</tr>
<tr>
<td>IN0</td>
<td>ANY</td>
<td>1. Input</td>
</tr>
<tr>
<td>IN1</td>
<td>ANY</td>
<td>2. Input</td>
</tr>
<tr>
<td>IN2</td>
<td>ANY</td>
<td>3. Input</td>
</tr>
<tr>
<td>INn</td>
<td>ANY</td>
<td>n+1. input, n = max. 30</td>
</tr>
</tbody>
</table>

Description of output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUT</td>
<td>ANY</td>
<td>Output</td>
</tr>
</tbody>
</table>

Runtime error

An error message is returned if the value range of the \(K\) input (selector) is exceeded.

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

**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**

```
SEL
Selection G OUT Output
Input0 IN0
Input1 IN1
```

**Representation in LD**

```
SEL
EN ENO
Selection G OUT Output
Input0 IN0
Input1 IN1
```
### 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:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection</td>
<td>BOOL</td>
<td>Selection input</td>
</tr>
<tr>
<td>Input0</td>
<td>ANY</td>
<td>Input 0</td>
</tr>
<tr>
<td>Input1</td>
<td>ANY</td>
<td>Input 1</td>
</tr>
</tbody>
</table>

**Description of the output parameter:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>ANY</td>
<td>Output</td>
</tr>
</tbody>
</table>
Strings

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:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Chapter Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>93</td>
<td>CONCAT_STR: Concatenation of two character strings</td>
<td>291</td>
</tr>
<tr>
<td>94</td>
<td>DELETE_INT: Deletion of a sub-string of characters</td>
<td>293</td>
</tr>
<tr>
<td>95</td>
<td>EQUAL_STR: Comparison of two character strings</td>
<td>295</td>
</tr>
<tr>
<td>96</td>
<td>FIND_INT: Finding a sub-string of characters</td>
<td>297</td>
</tr>
<tr>
<td>97</td>
<td>INSERT_INT: Insertion of a sub-string of characters</td>
<td>299</td>
</tr>
<tr>
<td>98</td>
<td>LEFT_INT: Extraction of characters to the left</td>
<td>303</td>
</tr>
<tr>
<td>99</td>
<td>LEN_INT: Length of character string</td>
<td>305</td>
</tr>
<tr>
<td>100</td>
<td>MID_INT: Extraction of a sub-string of characters</td>
<td>307</td>
</tr>
<tr>
<td>101</td>
<td>REPLACE_INT: Replacement of a sub-string of characters</td>
<td>309</td>
</tr>
<tr>
<td>102</td>
<td>RIGHT_INT: Extraction of a character string to the right</td>
<td>313</td>
</tr>
</tbody>
</table>
# CONCAT_STR: Concatenation of two character strings

## Description

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

## Representation in FBD

```
CONCAT_STR String1 String2
               IN1   IN2   OUT
               Result_String
```

## Representation in LD

```
CONCAT_STR
               EN   ENO

String1 String2
               IN1   IN2   OUT
               Result_String
```

## Representation in IL

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

Representation in ST

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

Description of parameters

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>String1</td>
<td>STRING</td>
<td>First character string to concatenate.</td>
</tr>
<tr>
<td>String2</td>
<td>STRING</td>
<td>Second character string to concatenate.</td>
</tr>
</tbody>
</table>

Example: String1 contains "SWITCH TO"
Example: String2 contains "RUN"

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result_String</td>
<td>STRING</td>
<td>Resulting string is equal to the content of the two strings String1 and String2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: for the values in the example provided in the previous table, Result_String contains 'SWITCH TO RUN'</td>
</tr>
</tbody>
</table>

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

### 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:

```
+---+---+---+---+
| IN | OUT |
+---+---+---+---+
| P  |  N  |
+---+---+---+---+
```

### Representation in LD

Representation:

```
+---+---+---+---+
| EN | ENO|
+---+---+---+---+
| IN | OUT |
+---+---+---+---+
| P  |  N  |
+---+---+---+---+
```
DELETE_INT

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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>String1</td>
<td>STRING</td>
<td>Original character string from which we wish to delete certain elements.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: String1 contains &quot;SWITCH TO STOP&quot;</td>
</tr>
<tr>
<td>Length_Str</td>
<td>INT</td>
<td>Length of string to be deleted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: Length_Str = 10</td>
</tr>
<tr>
<td>Position</td>
<td>INT</td>
<td>Rank of first character of the string to be deleted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: Position = 1</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result_String</td>
<td>STRING</td>
<td>Resulting string equal to content of String1 from which have been removed Length_Str characters starting from the rank Position.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: for the values in the example provided in the previous table, Result_String contains ‘STOP’ (10 characters are deleted starting from position 1).</td>
</tr>
</tbody>
</table>
EQUAL_STR: Comparison of two character strings

**Description**

**Function description**

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

**Representation in FBD**

Representation:

```
EQUAL_STR
IN1 OUT
String1 String2
IN2 Position
```

**Representation in LD**

Representation:

```
EQUAL_STR
EN ENO
IN1 OUT
String1 String2
IN2 Position
```

**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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>String1</td>
<td>STRING</td>
<td>First character string to compare.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Example:</strong> String1 contains &quot;SWITCH TO STOP&quot;</td>
</tr>
<tr>
<td>String2</td>
<td>STRING</td>
<td>Second character string to compare.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Example:</strong> String2 contains &quot;SWITCH TO RUN&quot;</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
<td>INT</td>
<td>Position of first character that differs between the two strings String1 and String2. When the two strings are identical, Position = -1. <strong>Example:</strong> with the values indicated in the example in the previous table, Position = 11</td>
</tr>
</tbody>
</table>

**Note:** upper case characters are treated as different to lower case characters.
FIND_INT: Finding a sub-string of characters

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 in LD

Representation in IL

LD String1
FIND_INT String2
ST Position
**Representation in ST**

Position := FIND_INT(String1, String2);

**Description of parameters**

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>String1</td>
<td>STRING</td>
<td>Character string in which the search is carried out.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: String1 contains &quot;SWITCH TO STOP&quot;</td>
</tr>
<tr>
<td>String2</td>
<td>STRING</td>
<td>Character string containing the text to find</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: String2 contains &quot;STOP&quot;</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
<td>INT</td>
<td>If String2 is contained in String1, Position contains the rank of the first character of String2 found in String1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When String2 is not contained in String1, Position = -1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: with the values indicated in the example in the previous table, Position = 11</td>
</tr>
</tbody>
</table>
INSERT_INT: Insertion of a substring of characters

**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:

```
  String1  IN1  OUT  Result_String
  String2  IN2  
P     Position
```

**Representation in LD**

Representation:

```
  EN  ENO
  String1  IN1  OUT  Result_String
  String2  IN2  
P     Position
```
**INSERT_INT**

**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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
</table>
| 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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
</table>
| 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

**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:

```
<table>
<thead>
<tr>
<th>IN</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>String1</td>
<td></td>
</tr>
<tr>
<td>Length_Str</td>
<td></td>
</tr>
</tbody>
</table>
```

**Representation in LD**

Representation:

```
<table>
<thead>
<tr>
<th>EN</th>
<th>ENO</th>
</tr>
</thead>
<tbody>
<tr>
<td>String1</td>
<td></td>
</tr>
<tr>
<td>Length_Str</td>
<td></td>
</tr>
</tbody>
</table>
```

**Representation in IL**

Representation:

```
LD String1
LEFT_INT Length_Str
ST Result_String
```
Representation in ST

Result_String := LEFT_INT(String1, Length_Str);

Description of parameters

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>String1</td>
<td>STRING</td>
<td>String of characters from which we wish to extract the Length_Str leftmost characters. Example: String1 contains &quot;SWITCH TO STOP&quot;</td>
</tr>
<tr>
<td>Length_Str</td>
<td>INT</td>
<td>Number of characters to be extracted. Example: Length_Str = 10.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result_String</td>
<td>STRING</td>
<td>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).</td>
</tr>
</tbody>
</table>

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

#### Description

The LEN_INT function calculates the number of characters of a character string. The additional parameters \( En \) and \( EnO \) can be configured.

#### Function description

<table>
<thead>
<tr>
<th>Representation in FBD</th>
<th>Representation:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="image" alt="FBD Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Representation in LD</th>
<th>Representation:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="image" alt="LD Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Representation in IL</th>
<th>Representation:</th>
</tr>
</thead>
<tbody>
<tr>
<td>LD String1 LEN_INT ST Length_Str</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Representation in ST</th>
<th>Representation:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length_Str:= LEN_INT(String1);</td>
<td></td>
</tr>
</tbody>
</table>
**Description of parameters**

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>String1</td>
<td>STRING</td>
<td>String of characters for which the length is to be determined.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Example</strong>: String1 contains &quot;SWITCH TO STOP&quot;</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length_Str</td>
<td>INT</td>
<td>Length_Str contains the length of the character string String1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Example</strong>: with the values indicated in the example in the previous table, Length_Str = 14</td>
</tr>
</tbody>
</table>
**MID_INT: Extraction of a sub-string of characters**

### 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

```
MID_INT
  String1 IN
  Length_Str N
  Position P

Result_String OUT
```

### Representation in LD

```
EN
MID_INT
ENO

String1 IN
Length_Str N
Position P

Result_String OUT
```
**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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>String1</td>
<td>STRING</td>
<td>Original string containing the sub-string to be extracted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Example</strong>: String1 contains &quot;SWITCH TO STOP&quot;</td>
</tr>
<tr>
<td>Length_Str</td>
<td>INT</td>
<td>Length of the sub-string to be extracted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Example</strong>: Length_Str = 4</td>
</tr>
<tr>
<td>Position</td>
<td>INT</td>
<td>Rank of first character of the sub-string to be extracted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Example</strong>: Position = 11</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result_String</td>
<td>STRING</td>
<td>Sub-string of String1 starting from rank Position over a length of Length_Str.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Example</strong>: for the values in the example provided in the previous table, Result_String contains ‘STOP’.</td>
</tr>
</tbody>
</table>
REPLACE_INT: Replacement of a sub-string of characters

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:

```
<table>
<thead>
<tr>
<th>REPLACE_INT</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN1</td>
</tr>
<tr>
<td>IN2</td>
</tr>
<tr>
<td>N</td>
</tr>
<tr>
<td>P</td>
</tr>
<tr>
<td>OUT</td>
</tr>
</tbody>
</table>
```

Representation in LD

Representation:

```
<table>
<thead>
<tr>
<th>REPLACE_INT</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
</tr>
<tr>
<td>ENO</td>
</tr>
<tr>
<td>IN1</td>
</tr>
<tr>
<td>IN2</td>
</tr>
<tr>
<td>N</td>
</tr>
<tr>
<td>P</td>
</tr>
<tr>
<td>OUT</td>
</tr>
<tr>
<td>Result_String</td>
</tr>
</tbody>
</table>
```
REPLACE_INT

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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>String1</td>
<td>STRING</td>
<td>Original string of characters into which is inserted a sub-string of characters starting from Position over a length of Length_Str.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: String1 contains &quot;SWITCH TO RUN&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>String2</td>
<td>STRING</td>
<td>Character string to be inserted in String1 to replace the existing characters.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: String2 contains 'STOP'</td>
</tr>
<tr>
<td>Length_Str</td>
<td>INT</td>
<td>Number of characters to be replaced in String1 by String2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: Length_Str = 3</td>
</tr>
<tr>
<td>Position</td>
<td>INT</td>
<td>Rank of first character of the sub-string to be replaced</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: Position = 11</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result_String</td>
<td>STRING</td>
<td>The string string2 has replaced the Length_Str characters starting from the rank Position in the string String1 to form Result_String.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: for the values in the example provided in the previous table, Result_String contains 'SWITCH TO STOP'.</td>
</tr>
</tbody>
</table>
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

**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:

```
RIGHT_INT
String1 IN OUT Result_String
Length_Str N
```

**Representation in LD**

Representation:

```
RIGHT_INT
EN ENO
String1 IN OUT Result_String
Length_Str N
```

**Representation in IL**

Representation:

```
LD String1
RIGHT_INT Length_Str
ST Result_String
```
**RIGHT_INT**

**Representation in ST**

Representation:

Result_String:= RIGHT_INT(String1, Length_Str);

**Description of parameters**

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>String1</td>
<td>STRING</td>
<td>String of characters from which we wish to extract the Length_Str rightmost characters.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: String1 contains &quot;SWITCH TO STOP&quot;</td>
</tr>
<tr>
<td>Length_Str</td>
<td>INT</td>
<td>Number of characters to be extracted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: Length_Str = 4</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result_String</td>
<td>STRING</td>
<td>String containing the Length_Str rightmost characters of String1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: for the values in the example provided in the previous table, Result_String contains 'STOP' (4 rightmost characters of String1).</td>
</tr>
</tbody>
</table>

**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.
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:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Chapter Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>103</td>
<td>CTD, CTD_***: Down counter</td>
<td>317</td>
</tr>
<tr>
<td>104</td>
<td>CTU, CTU_***: Up counter</td>
<td>321</td>
</tr>
<tr>
<td>105</td>
<td>CTUD, CTUD_***: Up/Down counter</td>
<td>325</td>
</tr>
<tr>
<td>106</td>
<td>TOF: Off delay</td>
<td>329</td>
</tr>
<tr>
<td>107</td>
<td>TON: On delay</td>
<td>331</td>
</tr>
<tr>
<td>108</td>
<td>TP: Pulse</td>
<td>333</td>
</tr>
</tbody>
</table>
CTD, CTD_***: Down counter

### 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 ≤ 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**

CTD_Instance

- **CTD**
- **Trigger**
- **CD**
- **Q**
- **Output**
- **Load**
- **LD**
- **PresetValue**
- **PV**
- **CV**
- **CountValue**

---

**Representation in LD**

CTD_Instance

- **CTD**
- **EN**
- **ENO**
- **Output**
- **Trigger**
- **CD**
- **Q**
- **Load**
- **LD**
- **PresetValue**
- **PV**
- **CV**
- **CountValue**

---

**Representation in IL**

```
CAL CTD_Instance (CD:=Trigger, LD:=Load, PV:=PresetValue, Q=>Output, CV=>CountValue)
```

---

**Representation in ST**

```
CTD_Instance (CD:=Trigger, LD:=Load, PV:=PresetValue, Q=>Output, CV=>CountValue) ;
```
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD</td>
<td>BOOL</td>
<td>Trigger input</td>
</tr>
<tr>
<td>LD</td>
<td>BOOL</td>
<td>Load data</td>
</tr>
<tr>
<td>PV</td>
<td>When CTD: INT, DINT, UINT, UDINT</td>
<td>Preset value</td>
</tr>
</tbody>
</table>

Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>BOOL</td>
<td>Output</td>
</tr>
<tr>
<td>CV</td>
<td>When CTD: INT, DINT, UINT, UDINT</td>
<td>Count value (actual value)</td>
</tr>
</tbody>
</table>
**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 \( CV \geq PV \), 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.

**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**
CTU, CTU_

### Representation in FBD

CTU_Instance (CU:=Trigger, R:=Reset, PV:=PresetValue, Q=>Output, CV=>CountValue)

### Representation in LD

CTU_Instance (CU:=Trigger, R:=Reset, PV:=PresetValue, Q=>Output, CV=>CountValue)

### Representation in IL

CAL CTU_Instance (CU:=Trigger, R:=Reset, PV:=PresetValue, Q=>Output, CV=>CountValue)

### Representation in ST

CTU_Instance (CU:=Trigger, R:=Reset, PV:=PresetValue, Q=>Output, CV=>CountValue) ;
### Parameter description

#### Description of the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>CU</td>
<td>BOOL</td>
<td>Trigger input</td>
</tr>
<tr>
<td>R</td>
<td>BOOL</td>
<td>Reset</td>
</tr>
</tbody>
</table>
| PV        | When CTU: INT  
When CTU_***: INT, DINT, UINT, UDINT | Preset value |

#### Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>BOOL</td>
<td>Output</td>
</tr>
</tbody>
</table>
| CV        | When CTU: INT  
When CTU_***: INT, DINT, UINT, UDINT | Count value (actual value) |
CTU, CTU***
CTUD, CTUD_***: Up/Down counter

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".

When CV ≤ 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

CTUD_Instance

CTUD

UpTrigger CU QU UpDisplay
DownTrigger CD QD DownDisplay
Reset R
Load LD
PresetValue PV CV CountValue

Representation in LD

CTUD_Instance

CTUD

EN ENO

UpDisplay

DownDisplay

UpTrigger CU QU
DownTrigger CD QD
Reset R
Load LD
PresetValue PV CV CountValue

Representation in IL

CAL CTUD_Instance (CU:=UpTrigger, CD:=DownTrigger, 
R:=Reset, LD:=Load, PV:=PresetValue, QU=>UpDisplay, 
QD=>DownDisplay, CV=>CountValue)

Representation in ST

CTUD_Instance (CU:=UpTrigger, CD:=DownTrigger, 
R:=Reset, LD:=Load, PV:=PresetValue, QU=>UpDisplay, 
QD=>DownDisplay, CV=>CountValue) ;
### Description of the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>CU</td>
<td>BOOL</td>
<td>Up counter trigger input</td>
</tr>
<tr>
<td>CD</td>
<td>BOOL</td>
<td>Down counter trigger input</td>
</tr>
<tr>
<td>R</td>
<td>BOOL</td>
<td>Reset</td>
</tr>
<tr>
<td>LD</td>
<td>BOOL</td>
<td>Load data</td>
</tr>
<tr>
<td>PV</td>
<td>When CTUD: INT,</td>
<td>Preset value</td>
</tr>
<tr>
<td></td>
<td>When CTUD_***: INT,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DINT, UINT, UDINT</td>
<td></td>
</tr>
</tbody>
</table>

### Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>QU</td>
<td>BOOL</td>
<td>Up display</td>
</tr>
<tr>
<td>QD</td>
<td>BOOL</td>
<td>Down display</td>
</tr>
<tr>
<td>CV</td>
<td>When CTUD: INT</td>
<td>Count value (actual value)</td>
</tr>
<tr>
<td></td>
<td>When CTUD_***: INT,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DINT, UINT, UDINT</td>
<td></td>
</tr>
</tbody>
</table>
TOF: Off delay

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:

```
CAL TOF_Instance (IN:=StartDelay, PT:=PresetDelayTime, Q=>Output, ET=>InternalTime)
```

Representation in LD

Representation:

```
CAL TOF_Instance (IN:=StartDelay, PT:=PresetDelayTime, Q=>Output, ET=>InternalTime)
```

Representation in IL

Representation:

```
CAL TOF_Instance (IN:=StartDelay, PT:=PresetDelayTime, Q=>Output, ET=>InternalTime)
```
Representation in ST

```plaintext
TOF_Instance (IN:=StartDelay, PT:=PresetDelayTime, Q=>Output, ET=>InternalTime) ;
```

Parameter description

**Description of the input parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>BOOL</td>
<td>Start delay</td>
</tr>
<tr>
<td>PT</td>
<td>TIME</td>
<td>Preset delay time</td>
</tr>
</tbody>
</table>

**Description of the output parameter:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>BOOL</td>
<td>Output</td>
</tr>
<tr>
<td>ET</td>
<td>TIME</td>
<td>Internal time</td>
</tr>
</tbody>
</table>

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".

(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**

**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:  
TON_Instance
    TON
    IN Q Output
    PT ET InternalTime
StartDelay
PresetDelayTime
```

**Representation in LD**

```
Representation:  
TON_Instance
    TON
    EN ENO Output
    IN Q
    PT ET InternalTime
StartDelay
PresetDelayTime
```

**Representation in IL**

```
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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>BOOL</td>
<td>Start delay</td>
</tr>
<tr>
<td>PT</td>
<td>TIME</td>
<td>Preset delay time</td>
</tr>
</tbody>
</table>

Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>BOOL</td>
<td>Output</td>
</tr>
<tr>
<td>ET</td>
<td>TIME</td>
<td>Internal time</td>
</tr>
</tbody>
</table>

**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".
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:

```
TP_Instance
TP
IN TriggerPulse
PT PulseDuration
Q Output
ET InternalTime
```

**Representation in LD**

Representation:

```
TP_Instance
TP
EN TriggerPulse
IN
ENO PulseDuration
PT
Q Output
ET InternalTime
```

**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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>BOOL</td>
<td>Trigger pulse</td>
</tr>
<tr>
<td>PT</td>
<td>TIME</td>
<td>Preset pulse duration</td>
</tr>
</tbody>
</table>

Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>BOOL</td>
<td>Output</td>
</tr>
<tr>
<td>ET</td>
<td>TIME</td>
<td>Internal time</td>
</tr>
</tbody>
</table>

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

#### 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:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Chapter Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>109</td>
<td>BCD_TO_INT: Conversion of a BCD integer into pure binary</td>
<td>339</td>
</tr>
<tr>
<td>110</td>
<td>BIT_TO_BYTE: Type conversion</td>
<td>341</td>
</tr>
<tr>
<td>111</td>
<td>BIT_TO_WORD: Type conversion</td>
<td>345</td>
</tr>
<tr>
<td>112</td>
<td>BOOL_TO_***: Type conversion</td>
<td>347</td>
</tr>
<tr>
<td>113</td>
<td>BYTE_AS_WORD: Type conversion</td>
<td>349</td>
</tr>
<tr>
<td>114</td>
<td>BYTE_TO_BIT: Type conversion</td>
<td>351</td>
</tr>
<tr>
<td>115</td>
<td>BYTE_TO_***: Type conversion</td>
<td>355</td>
</tr>
<tr>
<td>116</td>
<td>DATE_TO_STRING: Conversion of a variable in DATE format into a character string</td>
<td>359</td>
</tr>
<tr>
<td>117</td>
<td>DBCD_TO_***: Conversion of a double BCD integer into binary</td>
<td>361</td>
</tr>
<tr>
<td>118</td>
<td>DEG_TO_RAD: Conversion of degrees to radians</td>
<td>363</td>
</tr>
<tr>
<td>119</td>
<td>DINT_AS_WORD: Type conversion</td>
<td>365</td>
</tr>
<tr>
<td>120</td>
<td>DINT_TO_***: Type conversion</td>
<td>367</td>
</tr>
<tr>
<td>121</td>
<td>DINT_TO_DBCD: Conversion of a double binary coded integer into a double Binary Coded Decimal integer</td>
<td>371</td>
</tr>
<tr>
<td>122</td>
<td>DT_TO_STRING: Conversion of a variable in DT format into a character string</td>
<td>373</td>
</tr>
<tr>
<td>123</td>
<td>DWORD_TO_***: Type conversion</td>
<td>375</td>
</tr>
<tr>
<td>124</td>
<td>GRAY_TO_INT: Conversion of an integer in Gray code into a binary coded integer</td>
<td>377</td>
</tr>
<tr>
<td>125</td>
<td>INT_AS_DINT: Concatenation of two integers to form a double integer</td>
<td>379</td>
</tr>
<tr>
<td>126</td>
<td>INT_TO_***: Type conversion</td>
<td>381</td>
</tr>
<tr>
<td>127</td>
<td>INT_TO_BCD: Conversion of a binary coded integer into a Binary Coded Decimal integer</td>
<td>385</td>
</tr>
<tr>
<td>128</td>
<td>INT_TO_DBCD: Conversion of a binary coded integer into a double Binary Coded Decimal integer</td>
<td>387</td>
</tr>
<tr>
<td>129</td>
<td>RAD_TO_DEG: Conversion of radians to degrees</td>
<td>389</td>
</tr>
<tr>
<td>130</td>
<td>REAL_AS_WORD: Type conversion</td>
<td>391</td>
</tr>
<tr>
<td>131</td>
<td>REAL_TO_***: Type conversion</td>
<td>393</td>
</tr>
<tr>
<td>132</td>
<td>REAL_TRUNC_***: Type conversion</td>
<td>397</td>
</tr>
<tr>
<td>133</td>
<td>STRING_TO_***: Conversion of a character string to a number of the INT, DINT or REAL type</td>
<td>399</td>
</tr>
<tr>
<td>134</td>
<td>TYPE_AS_WORD: Type conversion</td>
<td>401</td>
</tr>
<tr>
<td>135</td>
<td>TIME_TO_***: Type conversion</td>
<td>403</td>
</tr>
<tr>
<td>Chapter</td>
<td>Chapter Name</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>136</td>
<td>TIME_TO_STRING: Conversion of a variable in TIME format into a character string</td>
<td>405</td>
</tr>
<tr>
<td>137</td>
<td>TOD_TO_STRING: Conversion of a variable in TOD format into a character string</td>
<td>407</td>
</tr>
<tr>
<td>138</td>
<td>UDINT_AS_WORD: Type conversion</td>
<td>409</td>
</tr>
<tr>
<td>139</td>
<td>UDINT_TO_***: Type conversion</td>
<td>411</td>
</tr>
<tr>
<td>140</td>
<td>UINT_TO_***: Type conversion</td>
<td>415</td>
</tr>
<tr>
<td>141</td>
<td>WORD_AS_BYTE: Type conversion</td>
<td>419</td>
</tr>
<tr>
<td>142</td>
<td>WORD_AS_DINT: Type conversion</td>
<td>421</td>
</tr>
<tr>
<td>143</td>
<td>WORD_AS_REAL: Type conversion</td>
<td>423</td>
</tr>
<tr>
<td>144</td>
<td>WORD_AS_TIME: Type conversion</td>
<td>425</td>
</tr>
<tr>
<td>145</td>
<td>WORD_AS_UDINT: Type conversion</td>
<td>427</td>
</tr>
<tr>
<td>146</td>
<td>WORD_TO_BIT: Type conversion</td>
<td>429</td>
</tr>
<tr>
<td>147</td>
<td>WORD_TO_***: Type conversion</td>
<td>433</td>
</tr>
<tr>
<td>148</td>
<td>***_TO_STRING: Conversion of a variable into a character string</td>
<td>437</td>
</tr>
</tbody>
</table>
Type to type
# BCD_TO_INT: Conversion of a BCD integer into pure binary

## 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.

## 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:**

```
<table>
<thead>
<tr>
<th>BCD_Int</th>
<th>IN</th>
<th>OUT</th>
<th>Result_Int</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCD_TO_INT</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

## Representation in LD

**Representation:**

```
LD BCD_Int
BCD_TO_INT IN EN OUT Result_Int
ENO
```

## Representation in IL

**Representation:**

```
LD BCD_Int
BCD_TO_INT
ST Result_Int
ENO
```
BCD_TO_INT

**Representation in ST**

Representation:

```plaintext
Result_Int := BCD_TO_INT(BCD_Int);
```

**Description of parameters**

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCD_Int</td>
<td>INT</td>
<td>Integer in BCD format.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Example:</strong> BCD_Int = 16#99</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result_Int</td>
<td>INT</td>
<td>Result_Int is a binary coded integer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Example:</strong> with the value provided in the example in the previous table, Result_Int = 99</td>
</tr>
</tbody>
</table>

**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**

**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:

\[ \text{OUT} = \{ \text{BIT7, BIT6, ..., BIT0} \} \]
BIT_TO_BYTE:

Representation in FBD

Representation:

BIT_TO_BYTE

InputBit0 — BIT0
InputBit1 — BIT1
InputBit2 — BIT2
InputBit3 — BIT3
InputBit4 — BIT4
InputBit5 — BIT5
InputBit6 — BIT6
InputBit7 — BIT7

OutputByte

Representation in LD

Representation:

BIT_TO_BYTE

EN — ENO

InputBit0 — BIT0
InputBit1 — BIT1
InputBit2 — BIT2
InputBit3 — BIT3
InputBit4 — BIT4
InputBit5 — BIT5
InputBit6 — BIT6
InputBit7 — BIT7

OutputByte

Representation in IL

Representation:

LD InputBit0

BIT_TO_BYTE InputBit1, InputBit2, InputBit3, InputBit4, InputBit5, InputBit6, InputBit7

ST OutputByte


**BIT_TO_BYTE:**

**Representation in ST**

OutputByte := BIT_TO_BYTE (InputBit0, InputBit1, InputBit2, InputBit3, InputBit4, InputBit5, InputBit6, InputBit7);

**Parameter description**

Description of the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>InputBit0</td>
<td>BOOL</td>
<td>Input bit 0</td>
</tr>
<tr>
<td>InputBit1</td>
<td>BOOL</td>
<td>Input bit 1</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>InputBit7</td>
<td>BOOL</td>
<td>Input bit 7</td>
</tr>
</tbody>
</table>

Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>OutputByte</td>
<td>BYTE</td>
<td>Output value</td>
</tr>
</tbody>
</table>
**BIT_TO_WORD: Type conversion**

**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.

\[
\begin{align*}
\text{BIT0 (2^0)} & \quad \text{BIT1 (2^1)} \\
\vdots & \\
\text{BIT14 (2^{14})} & \quad \text{BIT15 (2^{15})}
\end{align*}
\]

\[
\text{EN and ENO can be configured as additional parameters.}
\]

**Formula**

Block formula:

\[
\text{OUT} = \{\text{BIT15,BIT14,\ldots,BIT0}\}
\]

**Representation in FBD**

Representation:

```
BIT0 \quad \text{BIT15} \quad \text{OUT} \quad \text{WORD_Output}
```

345
**BIT_TO_WORD**

**Representation in LD**

```
Representation:

<table>
<thead>
<tr>
<th>BIT_TO_WORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
</tr>
<tr>
<td>ENO</td>
</tr>
<tr>
<td>BIT0</td>
</tr>
<tr>
<td>:</td>
</tr>
<tr>
<td>:</td>
</tr>
<tr>
<td>BIT15</td>
</tr>
<tr>
<td>InputBit0</td>
</tr>
<tr>
<td>:</td>
</tr>
<tr>
<td>:</td>
</tr>
<tr>
<td>InputBit15</td>
</tr>
</tbody>
</table>
```

**Representation in IL**

```
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**

```
WORD_Output := BIT_TO_WORD (InputBit0, InputBit1, InputBit2, InputBit3, InputBit4, InputBit5, InputBit6, InputBit7, InputBit8, InputBit9, InputBit10, InputBit11, InputBit12, InputBit13, InputBit14, InputBit15);
```

**Parameter description**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>InputBit0</td>
<td>BOOL</td>
<td>Input bit 0</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>InputBit15</td>
<td>BOOL</td>
<td>Input bit 15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD_Output</td>
<td>WORD</td>
<td>Output value</td>
</tr>
</tbody>
</table>
## BOOL_TO_***: Type conversion

### 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:

```
<table>
<thead>
<tr>
<th></th>
<th>BOOLEAN</th>
<th>BOOLEAN_TO_INT</th>
<th>OUT</th>
<th>ConvertedVariable</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>BOOLEAN</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

---

347
BOOL_TO***

Representation in LD

Representation of an Integer application:

BOOL_TO_INT

IN

EN

ENO

OUT

ConvertedVariable

BOOL_variable

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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL_variable</td>
<td>BOOL</td>
<td>Input value</td>
</tr>
</tbody>
</table>

Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvertedVariable</td>
<td>BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME</td>
<td>Output value</td>
</tr>
</tbody>
</table>
**BYTE_AS_WORD: Type conversion**

**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:

\[ \text{OUT} = \{\text{HIGH, LOW}\} \]

**Representation in FBD**

```
  BYTE_AS_WORD
  BYTE_variable1  LOW  OUT
  BYTE_variable2  HIGH

  WORD_Output
```

**Representation in LD**

```
  BYTE_AS_WORD
  EN  ENO
  BYTE_variable1  LOW  OUT
  BYTE_variable2  HIGH

  WORD_Output
```
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>BYTE_variable1</td>
<td>BYTE</td>
<td>least significant byte</td>
</tr>
<tr>
<td>BYTE_variable2</td>
<td>BYTE</td>
<td>most significant byte</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD_Output</td>
<td>WORD</td>
<td>Output value</td>
</tr>
</tbody>
</table>
BYTE_TO_BIT: Type conversion

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:

```
BYTE_TO_BIT

BYTE_variable IN

BIT0 BOOL_variable1
BIT1 BOOL_variable2
BIT2 BOOL_variable3
BIT3 BOOL_variable4
BIT4 BOOL_variable5
BIT5 BOOL_variable6
BIT6 BOOL_variable7
BIT7 BOOL_variable8
```
**Representation in LD**

**Representation:**

```
BYTE_TO_BIT
  EN
  IN
  BYTE_variable
```

**Representation in IL**

```
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**

```
BYTE_TO_BIT (BYTE_variable, BOOL_variable1, BOOL_variable2,
  BOOL_variable3, BOOL_variable4, BOOL_variable5,
  BOOL_variable6, BOOL_variable7, BOOL_variable8);
```
## Description of the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>BYTE_variable</td>
<td>BYTE</td>
<td>Input</td>
</tr>
</tbody>
</table>

## Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL_variable0</td>
<td>BOOL</td>
<td>Output bit 0</td>
</tr>
<tr>
<td>BOOL_variable1</td>
<td>BOOL</td>
<td>Output bit 1</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>BOOL_variable7</td>
<td>BOOL</td>
<td>Output bit 7</td>
</tr>
</tbody>
</table>
BYTE_TO_***: Type conversion

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
**BYTE_TO_INT**

**Representation in FBD**

Representation of an Integer application:

```
  BYTE_variable IN OUT ConvertedVariable
              BYTE_TO_INT
```

**Representation in LD**

Representation of an Integer application:

```
  BYTE_TO_INT
  EN         ENO
  BYTE_variable IN OUT ConvertedVariable
```

**Representation in IL**

Representation of an Integer application:

```
  LD BYTE_variable
  BYTE_TO_INT
  ST ConvertedVariable
```

**Parameter description**

Description of input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>BYTE_variable</td>
<td>BYTE</td>
<td>Input value</td>
</tr>
</tbody>
</table>

Description of output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvertedVariable</td>
<td>BOOL, WORD, DWORD, INT, DINT, UINT, UDINT, REAL, TIME</td>
<td>Output value</td>
</tr>
</tbody>
</table>
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

**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

Representations:

- **Date1** → **DATE_TO_STRING** → **Result_Str**

### Representation in LD

Representations:

- **LD**
  - Date1
  - **DATE_TO_STRING**
  - **Result_Str**

### Representation in IL

Representations:

- **LD**
  - Date1
  - **DATE_TO_STRING**
  - **ST**
  - **Result_Str**

### Representation in ST

Representations:

- **Result_Str := DATE_TO_STRING(Date1);**
DATE_TO_STRING

**Description of parameters**

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date1</td>
<td>DATE</td>
<td>Date to be converted into character string format.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
</table>
| 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 is 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

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:

```
DBCD_VALUE
```

```
DBCD_TO_INT
IN  OUT  Result_Value
```

Representation in LD

```
DBCD_VALUE
```

```
DBCD_TO_INT
EN  ENO
```

```
IN  OUT  Result_Value
```
**DBCD_TO_***

**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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBCD_Value</td>
<td>DINT</td>
<td>Double integer in BCD format.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Example:</strong> DBCD_Value = 16#32767</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result_Value</td>
<td>INT,DINT</td>
<td>Result_Value is an integer or double integer in binary code.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Example:</strong> with the value provided in the example in the previous table, Result_Value = 32767</td>
</tr>
</tbody>
</table>

**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

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:

\[ \text{Angle in Radian} = \text{DEG_TO_RAD (Angle in Degree)} \]

Representation in FBD

```
DEG_TO_RAD

IN    OUT
```

Representation in LD

```
DEG_TO_RAD

EN    ENO

IN    OUT
```

Angle_in_Degree

Angle_in_Radian

ENO

IN

OUT

Angle_in_Radian

EN

OUT
**DEG_TO_RAD**

**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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle_in_Degree</td>
<td>REAL</td>
<td>Angle expressed in degrees.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-737280.0 &lt; Angle_in_Degree &lt; +737280.0.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle_in_Radian</td>
<td>REAL</td>
<td>Value of Angle expressed in radians.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-π ≤ Angle_in_Radian ≤ +π.</td>
</tr>
</tbody>
</table>

**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

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:

```
DINT_variable     IN  LOW  LowWord
                 HIGH HighWord
```

Representation in LD

Representation:

```
DINT_variable     IN  LOW  LowWord
                 HIGH HighWord
```

Representation in IL

Representation:

```
LD DINT_variable
DINT_AS_WORD LowWord, HighWord
```
**DINT_AS_WORD**

**Representation in ST**

Representation:

DINT_AS_WORD (DINT_variable, LowWord, HighWord);

**Parameter description**

Description of the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DINT_variable</td>
<td>DINT</td>
<td>Input</td>
</tr>
</tbody>
</table>

Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>LowWord</td>
<td>WORD</td>
<td>least significant word</td>
</tr>
<tr>
<td>HighWord</td>
<td>WORD</td>
<td>most significant word</td>
</tr>
</tbody>
</table>
### DINT_TO_***: Type conversion

#### 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:

<table>
<thead>
<tr>
<th>DINT_variable</th>
<th>IN</th>
<th>DINT_TO_INT</th>
<th>OUT</th>
<th>ConvertedVariable</th>
</tr>
</thead>
</table>

367
**Representation in LD**

Representation of an Integer application:

```
<table>
<thead>
<tr>
<th>DINT_TO_INT</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
</tr>
<tr>
<td>ENO</td>
</tr>
<tr>
<td>DINT_variable</td>
</tr>
<tr>
<td>IN</td>
</tr>
<tr>
<td>OUT</td>
</tr>
<tr>
<td>ConvertedVariable</td>
</tr>
</tbody>
</table>
```

**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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DINT_variable</td>
<td>DINT, BOOL, BYTE, WORD, DWORD, INT, UINT, UDINT, REAL, TIME</td>
<td>Input value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvertedVariable</td>
<td></td>
<td>Output value</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function description</td>
</tr>
<tr>
<td>The <strong>DINT_TO_DBCD</strong> function carries out the conversion of a double binary coded integer into an integer in Double Binary Coded Decimal (DBCD) format. The additional parameters <strong>EN</strong> and <strong>ENO</strong> can be configured.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Representation in FBD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Representation:</strong></td>
</tr>
<tr>
<td><img src="image" alt="FBD Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Representation in LD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Representation:</strong></td>
</tr>
<tr>
<td><img src="image" alt="LD Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Representation in IL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Representation:</strong></td>
</tr>
</tbody>
</table>
| LD D_Integer_1  
DINT_TO_BCD  
ST DBCD_Result |
**DINT_TO_DBCD**

**Representation in ST**

```
DBCD_Result := DINT_TO_BCD(D_Integer_1);
```

**Description of parameters**

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>D_Integer_1</td>
<td>DINT</td>
<td>Double binary coded integer between 0 and 99999999.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td>D_Integer_1 = 888888</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBCD_Result</td>
<td>DINT</td>
<td>DBCD_Result is a double integer in BCD format.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td>with the value provided in the example in the previous table, DBCD_Result = 16#00888888</td>
</tr>
</tbody>
</table>

**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

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:

```
DT_TO_STRING
```

IN

OUT

Date1

Result_Str

Representation in LD

Representation:

```
DT_TO_STRING
```

IN

OUT

EN

ENO

Date1

Result_Str

Representation in IL

Representation:

```
LD Date1
DT_TO_STRING
ST Result_Str
```
**Representation in ST**

Result_Str:= DT_TO_STRING(Date1);

**Description of parameters**

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date1</td>
<td>DT</td>
<td>Date to be converted into character string format.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
</table>
| 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**

### 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:

```
DWORD_variable IN DWORD_TO_INT OUT ConvertedVariable
```
DWORD_TO_***

Representation in LD

Representation of an Integer application:

```
LD DWORD_variable
DWORD_TO_INT
ST ConvertedVariable
```

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DWORD_variable</td>
<td>DWORD</td>
<td>Input value</td>
</tr>
</tbody>
</table>

Description of output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvertedVariable</td>
<td>BOOL, BYTE, WORD, INT, DINT, UINT, UDINT, REAL, TIME</td>
<td>Output value</td>
</tr>
</tbody>
</table>

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

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:

```
GRAY_TO_INT
GRAY_Int IN OUT Result_Int
```

Representation in LD

```
GRAY_TO_INT
GRAY_Int EN ENO
GRAY_Int IN OUT Result_Int
```

Representation in IL

```
LD GRAY_Int
GRAY_TO_INT
ST Result_Int
```
GRAY_TO_INT

Representation in ST

Representation:
Result_Int := GRAY_TO_INT(GRAY_Int);

Description of parameters

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRAY_Int</td>
<td>INT</td>
<td>Integer expressed in GRAY code.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result_Int</td>
<td>INT</td>
<td>Result_Int is a binary coded integer.</td>
</tr>
</tbody>
</table>
**INT_AS_DINT**: Concatenation of two integers to form a double integer

### 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:**

```
INT_AS_DINT
Low_Word LOW OUT Double_Word
High_Word HIGH
```

### Representation in LD

**Representation:**

```
INT_AS_DINT
EN ENO
Low_Word LOW OUT Double_Word
High_Word HIGH
```

### Representation in IL

**Representation:**

```
LD Low_Word
INT_AS_DINT High_Word
ST Double_Word
```
INT_AS_DINT

Representation in ST

Representation:

\[
\text{Double Word} := \text{INT AS DINT}(\text{Low Word}, \text{High Word});
\]

Description of parameters

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
</table>
| 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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
</table>
| 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**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The function converts an input value of the <strong>INT</strong> data type to a <strong>BOOL</strong>, <strong>BYTE</strong>, <strong>WORD</strong>, <strong>DWORD</strong>, <strong>DINT</strong>, <strong>UINT</strong>, <strong>UDINT</strong>, <strong>REAL</strong> or <strong>TIME</strong> output value.</td>
</tr>
</tbody>
</table>

| 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**. |

<table>
<thead>
<tr>
<th>Available functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of available functions:</td>
</tr>
<tr>
<td>• <strong>INT_TO_BOOL</strong></td>
</tr>
<tr>
<td>• <strong>INT_TO_BYTE</strong></td>
</tr>
<tr>
<td>• <strong>INT_TO_WORD</strong></td>
</tr>
<tr>
<td>• <strong>INT_TO_DWORD</strong></td>
</tr>
<tr>
<td>• <strong>INT_TO_DINT</strong></td>
</tr>
<tr>
<td>• <strong>INT_TO_UINT</strong></td>
</tr>
<tr>
<td>• <strong>INT_TO_UDINT</strong></td>
</tr>
<tr>
<td>• <strong>INT_TO_REAL</strong></td>
</tr>
<tr>
<td>• <strong>INT_TO_TIME</strong></td>
</tr>
</tbody>
</table>

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.
INT_TO_***

**Representation in FBD**

Representation of a double integer application:

```
  INT_TO_DINT

  └───┐
     └───┐
      └───┘

  INT_variable ──── IN
             OUT ──── ConvertedVariable
```

**Representation in LD**

Representation of a double integer application:

```
  INT_TO_DINT

  ┌───┐
  │ EN │
  │ ENO│
  └───┘

  INT_variable ──── IN
             OUT ──── ConvertedVariable
```

**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:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT_variable</td>
<td>INT</td>
<td>Input value</td>
</tr>
</tbody>
</table>

**Description of output parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvertedVariable</td>
<td>BOOL, BYTE, DWORD, WORD, DINT, UINT, UDINT, REAL, TIME</td>
<td>Output value</td>
</tr>
</tbody>
</table>
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 a 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**

**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.

**Function description**

- **Representation in FBD**

  Representation:

  ![INT_TO_BCD FBD representation](image)

  - `Integer_1` IN
  - `INT_TO_BCD`
  - `BCD_Result` OUT

- **Representation in LD**

  Representation:

  ![INT_TO_BCD LD representation](image)

  - `Integer_1` IN
  - `INT_TO_BCD` EN
  - `BCD_Result` OUT

- **Representation in IL**

  Representation:

  ```
  LD Integer_1
  INT_TO_BCD
  ST BCD_Result
  ```
INT_TO_BCD

Representation in ST

Representation:

\[ \text{BCD\_Result} := \text{INT\_TO\_BCD}(\text{Integer\_1}); \]

Description of parameters

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer_1</td>
<td>INT</td>
<td>Binary coded integer between 0 and 9999.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: Integer_1 = 99</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCD_Result</td>
<td>INT</td>
<td>BCD_Result is a BCD integer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: with the value provided in the example in the previous table, BCD_Result = 16#99</td>
</tr>
</tbody>
</table>

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

**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:

```
Integer_1 INT_TO_DBCD OUT DBCD_Result
```

**Representation in LD**

Representation:

```
INT_TO_DBCD
EN ENO
IN OUT DBCD_Result
```

**Representation in IL**

Representation:

```
LD Integer_1
INT_TO_BCD
ST DBCD_Result
```
INT_TO_DBCD

Representation in ST

Representation:
DBCD_Result := INT_TO_BCD(Integer_1);

Description of parameters

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer_1</td>
<td>INT</td>
<td>Binary coded integer between 0 and 32768. Example: Integer_1 = 30000</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBCD_Result</td>
<td>DINT</td>
<td>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</td>
</tr>
</tbody>
</table>

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

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:

\[
\text{RAD_TO_DEG} \; (\text{Angle\_in\_Radian}) = \text{Angle\_in\_Degree}
\]

Representation in FBD

Representation:

![FBD Diagram]

Representation in LD

Representation:

![LD Diagram]

Representation in IL

Representation:

LD Angle\_in\_Radian
RAD_TO_DEG
ST Angle\_in\_Degree
**RAD_TO_DEG**

**Representation in ST**

Representation:

\[ \text{Angle\_in\_Degree} := \text{RAD\_TO\_DEG}(\text{Angle\_in\_Radian}); \]

**Description of parameters**

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle_in_Radian</td>
<td>REAL</td>
<td>Value of Angle expressed in radians. [-4096\pi \leq \text{Angle_in_Radian} \leq +4096\pi]</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle_in_Degree</td>
<td>REAL</td>
<td>Angle expressed in degrees. [-360 &lt; \text{Angle_in_Degree} &lt; +360.]</td>
</tr>
</tbody>
</table>

**Runtime errors**

When \( \text{Angle\_in\_Degree} \) 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

**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.

**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:

```
REAL_variable IN  LOW  LowWord
REAL_variable IN  HIGH HighWord
```

**Representation in LD**

Representation:

```
REAL_variable IN  ENO
REAL_variable IN  LOW  LowWord
REAL_variable IN  HIGH HighWord
```

**Representation in IL**

Representation:

```
LD REAL_variable
REAL_AS_WORD LowWord, HighWord
```
**REAL_AS_WORD**

**Representation in ST**

Representation:
REAL_AS_WORD (REAL_variable, LowWord, HighWord);

**Parameter description**

Description of input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>REAL_variable</td>
<td>REAL</td>
<td>Input</td>
</tr>
</tbody>
</table>

Description of output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>LowWord</td>
<td>WORD</td>
<td>least significant word</td>
</tr>
<tr>
<td>HighWord</td>
<td>WORD</td>
<td>most significant word</td>
</tr>
</tbody>
</table>

**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

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.

ENO 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

- 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:

```
REAL_variable | REAL_TO_INT | ConvertedVariable
  IN          |            | OUT
```

Representation in LD

Representation of an Integer application:

```
REAL_variable | REAL_TO_INT | EN
  IN          |            | ENO

REAL_variable | IN
              | OUT
```

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);
```
### Description of input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>REAL_variable</td>
<td>REAL</td>
<td>Input value</td>
</tr>
</tbody>
</table>

### Description of output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvertedVariable</td>
<td>BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, TIME</td>
<td>Output value</td>
</tr>
</tbody>
</table>

### 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

Description

Function description
The function converts (by truncating towards zero) a REAL data type input value to an 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:

```
REAL_variable IN       REAL_TRUNC_INT
                   OUT    ConvertedVariable
```
REAL_TRUNC_INT

Representation of an Integer application:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>REAL_variable</td>
<td>REAL</td>
<td>Input value</td>
</tr>
</tbody>
</table>

Parameter description

Description of the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvertedVariable</td>
<td>INT, DINT, UINT, UDINT</td>
<td>Output value</td>
</tr>
</tbody>
</table>

Description of the output parameter:

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

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 in FBD

Representation applied to an integer:

```
STRING_TO_INT
IN  OUT
```

Representation in LD

Representation applied to an integer:

```
STRING_TO_INT
EN  ENO
IN  OUT
```
Representation applied to an integer:
LD String_1
STRING_TO_INT
ST Result_Value

Representation applied to an integer:
Result_Value := STRING_TO_INT(String_1);

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>String_1</td>
<td>STRING</td>
<td>Character string</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: String_1 = ‘-32500’</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result_Value</td>
<td>INT, DINT, REAL</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: with the value of the example in the above table, Result_Value = -32500.</td>
</tr>
</tbody>
</table>

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

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

```
TIME_variable  TIME_AS_WORD
   IN        LOW    LowWord
            HIGH   HighWord
```

Representation in LD

```
TIME_variable  TIME_AS_WORD
   IN        EN     ENO
            LOW    LowWord
                HIGH   HighWord
```

Representation in IL

```
LD TIME_variable
TIME_AS_WORD LowWord, HighWord
```
TIME_AS_WORD

Representation in ST

Representation:
TIME_AS_WORD (REAL_variable, LowWord, HighWord);

Parameter description

Description of input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME_variable</td>
<td>TIME</td>
<td>Input</td>
</tr>
</tbody>
</table>

Description of output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>LowWord</td>
<td>WORD</td>
<td>least significant word</td>
</tr>
<tr>
<td>HighWord</td>
<td>WORD</td>
<td>most significant word</td>
</tr>
</tbody>
</table>
TIME_TO_***: Type conversion

**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:

```
TIME_variable    TIME_TO_INT
                   IN    OUT
                 ConvertedVariable
```
Representation of an Integer application:

**Representation in LD**

```
TIME_TO_INT

EN  ENO
TIME_variable  IN  OUT  ConvertedVariable
```

**Representation in IL**

```
LD TIME_variable
TIME_TO_INT
ST ConvertedVariable
```

**Representation in ST**

```
ConvertedVariable := TIME_TO_INT (TIME_variable) ;
```

**Parameter description**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME_variable</td>
<td>TIME</td>
<td>Input value</td>
</tr>
</tbody>
</table>

**Description of the output parameter:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvertedVariable</td>
<td>BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL</td>
<td>Output value</td>
</tr>
</tbody>
</table>

**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

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:  

```
TIME_TO_STRING
```

```
IN     OUT
Time1   Result_Str
```

 Representation

in LD

Representation:  

```
TIME_TO_STRING
```

```
IN      OUT
Time1   ENO
EN      ENO
```

 Representation

in IL

Representation:  

```
LD Time1
TIME_TO_STRING
ST Result_Str
```

 Representation

in ST

Representation:  

```
Result_Str:= TIME_TO_STRING(Time1);
```

136
The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time1</td>
<td>TIME</td>
<td>Duration to be converted into character string format.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result_Str</td>
<td>STRING</td>
<td>Result_Str is a string of 15 characters which contains a duration in the following format: HHHHHH:MM:SS.D.</td>
</tr>
</tbody>
</table>

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**

**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:

```
TOD_TO_STRING
Time1 IN OUT Result_Str
```

**Representation in LD**

Representation:

```
TOD_TO_STRING
EN ENO
Time1 IN OUT Result_Str
```

**Representation in IL**

```
LD Timel
TOD_TO_STRING
ST Result_Str
```

**Representation in ST**

```
Result_Str:= =TOD_TO_STRING(Timel);
```
The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time1</td>
<td>TOD</td>
<td>Time of day to be converted into character string format.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result_Str</td>
<td>STRING</td>
<td>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).</td>
</tr>
</tbody>
</table>

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

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:

```
UDINT_variable  IN  LOW  LowWord
               HIGH  HighWord
```

Representation in LD

Representation:

```
UDINT_AS_WORD

EN  ENO

UDINT_variable  IN  LOW  LowWord
               HIGH  HighWord
```

Representation in IL

```
LD UDINT_variable
UDINT_AS_WORD LowWord, HighWord
```
UDINT_AS_WORD

**Representation in ST**

Representation:

\[ \text{UDINT\_AS\_WORD (UDINT\_variable, LowWord, HighWord);} \]

**Parameter description**

Description of input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDINT_variable</td>
<td>UDINT</td>
<td>Input</td>
</tr>
</tbody>
</table>

Description of output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>LowWord</td>
<td>WORD</td>
<td>least significant word</td>
</tr>
<tr>
<td>HighWord</td>
<td>WORD</td>
<td>most significant word</td>
</tr>
</tbody>
</table>
**UDINT_TO_***: Type conversion**

**Description**

The function converts an input value of the UDINT data type to an output value of the BOOLEAN, 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 BOOLEAN, 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:

```
UDINT_variable IN OUT ConvertedVariable
```

---

139

411
Representation of an Integer application:

**LD**

```
UDINT_TO_INT
```

**IN**

```
UDINT_variable
```

**OUT**

```
ConvertedVariable
```

**IL**

```
LD UDINT_variable
UDINT_TO_INT
ST ConvertedVariable
```

**ST**

```
ConvertedVariable := UDINT_TO_INT (UDINT_variable) ;
```

Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDINT_variable</td>
<td>UDINT</td>
<td>Input value</td>
</tr>
</tbody>
</table>

Description of output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvertedVariable</td>
<td>BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, REAL, TIME</td>
<td>Output value</td>
</tr>
</tbody>
</table>
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

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.

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

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.
Representation in FBD

Representation of an Integer application:

```
  UINT_TO_INT
       IN          OUT
      UINT_variable     ConvertedVariable
```

Representation in LD

Representation of an Integer application:

```
  UINT_TO_INT
       IN          OUT
      EN          ENO
      UINT_variable     OUT
```

Representation in IL

LD UINT_variable
UINT_TO_INT
ST ConvertedVariable

Parameter description

Description of input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>UINT_variable</td>
<td>UINT</td>
<td>Input value</td>
</tr>
</tbody>
</table>

Description of output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvertedVariable</td>
<td>BOOL, BYTE, WORD, DWORD, INT, DINT, UDINT, REAL, TIME</td>
<td>Output value</td>
</tr>
</tbody>
</table>
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_***
WORD_AS_BYTE: Type conversion

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:

```
WORD_variable     WORD_AS_BYTE
        IN    LOW    LowByte
        HIGH   HighByte
```

Representation in LD

```
EN    ENO

WORD_variable     LD WORD_variable
        IN    LOW    LowByte
        HIGH   HighByte
```

Representation in IL

```
LD WORD_variable
WORD_AS_BYTE LowByte, HighByte
```
### Representation in ST

Representation:

```plaintext
WORD_AS_BYTE (WORD_variable, LowByte, HighByte);
```

### Parameter description

**Description of the input parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD_variable</td>
<td>WORD</td>
<td>Input</td>
</tr>
</tbody>
</table>

**Description of the output parameter:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>LowByte</td>
<td>BYTE</td>
<td>least significant byte</td>
</tr>
<tr>
<td>HighByte</td>
<td>BYTE</td>
<td>most significant byte</td>
</tr>
</tbody>
</table>
WORD_AS_DINT: Type conversion

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:

```
| WORD_variable1 | LOW | OUT    |
| WORD_variable2 | HIGH|        |
```

Representation in LD

```
| WORD_AS_DINT | DINT_variable |
| EN            | ENO           |
| WORD_variable1 | LOW | OUT    |
| WORD_variable2 | HIGH |
```
### Representation in IL

Representation:
```plaintext
LD WORD_variable1
WORD_AS_DINT WORD_variable2
ST DINT_variable
```

### Representation in ST

Representation:
```plaintext
DINT_variable := WORD_AS_DINT (WORD_variable1,
                               WORD_variable2) ;
```

### Parameter description

#### Description of the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD_variable1</td>
<td>WORD</td>
<td>least significant word</td>
</tr>
<tr>
<td>WORD_variable2</td>
<td>WORD</td>
<td>most significant word</td>
</tr>
</tbody>
</table>

#### Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DINT_variable</td>
<td>DINT</td>
<td>Output value</td>
</tr>
</tbody>
</table>
WORD_AS_REAL: Type conversion

Description

**Function description**
The procedure converts an input value of the \texttt{WORD} data type to output values of the \texttt{REAL} data type. The input values are assigned to the word at the output according to the input names. \texttt{EN} and \texttt{ENO} can be configured as additional parameters.

**Formula**
Block formula:
\[
\text{OUT} = \{\text{HIGH}, \text{LOW}\}
\]

**Representation in FBD**

```plaintext
<table>
<thead>
<tr>
<th>WORD_variable1</th>
<th>LOW</th>
<th>OUT</th>
<th>REAL_Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD_variable2</td>
<td>HIGH</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**Representation in LD**

```plaintext
<table>
<thead>
<tr>
<th>WORD_variable1</th>
<th>LOW</th>
<th>OUT</th>
<th>REAL_Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD_variable2</td>
<td>HIGH</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD_variable1</td>
<td>WORD</td>
<td>least significant byte</td>
</tr>
<tr>
<td>WORD_variable2</td>
<td>WORD</td>
<td>most significant byte</td>
</tr>
</tbody>
</table>

Description of output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>REAL_Output</td>
<td>REAL</td>
<td>Output value</td>
</tr>
</tbody>
</table>

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

## 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:

\[
\text{OUT} = \{\text{HIGH}, \text{LOW}\}
\]

## Representation in FBD

<table>
<thead>
<tr>
<th>WORD_variable1</th>
<th>LOW</th>
<th>OUT</th>
<th>TIME_Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD_variable2</td>
<td>HIGH</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Representation in LD

<table>
<thead>
<tr>
<th>WORD_variable1</th>
<th>LOW</th>
<th>OUT</th>
<th>TIME_Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD_variable2</td>
<td>HIGH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EN</td>
<td></td>
<td>ENO</td>
<td></td>
</tr>
</tbody>
</table>

425
### 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:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD_variable1</td>
<td>WORD</td>
<td>least significant byte</td>
</tr>
<tr>
<td>WORD_variable2</td>
<td>WORD</td>
<td>most significant byte</td>
</tr>
</tbody>
</table>

**Description of output parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME_Output</td>
<td>TIME</td>
<td>Output value</td>
</tr>
</tbody>
</table>
WORD_AS_UDINT: Type conversion

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:
\[ \text{OUT} = \{\text{HIGH, LOW}\} \]

Representation in FBD

Representation:

\[
\begin{array}{c}
\text{WORD AS UDINT} \\
\text{LOW} & \text{OUT} & \text{UDINT Output} \\
\text{WORD_variable1} & \text{HIGH} & \\
\end{array}
\]

Representation in LD

Representation:

\[
\begin{array}{c}
\text{WORD AS UDINT} \\
\text{EN} & \text{ENO} \\
\text{LOW} & \text{OUT} & \text{UDINT Output} \\
\text{WORD_variable1} & \text{HIGH} & \\
\end{array}
\]
WORD_AS_UDINT

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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD_variable1</td>
<td>WORD</td>
<td>least significant byte</td>
</tr>
<tr>
<td>WORD_variable2</td>
<td>WORD</td>
<td>most significant byte</td>
</tr>
</tbody>
</table>

Description of output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDINT_Output</td>
<td>UDINT</td>
<td>Output value</td>
</tr>
</tbody>
</table>
**WORD_TO_BIT: Type conversion**

### 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:

```
WORD_variable   IN
    WORD_TO_BIT
       BIT0 (2^0)
       BIT1 (2^1)
       ...     ...
       BIT14 (2^14)
       BIT15 (2^15)

BOOL_variable1  
:               
BOOL_variable16
```

---

429
WORD_TO_BIT

**Representation in LD**

Representation:

```plaintext
WORD_TO_BIT

EN  Eno
    IN BIT0
        : BIT15
    0  BOOL_variable1
    1  BOOL_variable16

WORD_variable IN
```

**Representation in IL**

Representation:

```plaintext
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:

```plaintext
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);
```
### Description of the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD_variable</td>
<td>WORD</td>
<td>Input</td>
</tr>
</tbody>
</table>

### Description of the output parameter:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL_variable1</td>
<td>BOOL</td>
<td>Output BIT0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BOOL_variable16</td>
<td>BOOL</td>
<td>Output BIT15</td>
</tr>
</tbody>
</table>
WORD_TO_***: Type conversion

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:

```
  WORD_variable  IN  OUT  ConvertedVariable
            WORD_TO_INT
```

**Representation in LD**

Representation of an Integer application:

```
  WORD_TO_INT  EN  ENO
  WORD_variable  IN  OUT  ConvertedVariable
```

**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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD_variable</td>
<td>WORD</td>
<td>Input value</td>
</tr>
</tbody>
</table>

Description of output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Converted Variable</td>
<td>BOOL, BYTE, DWORD, INT, DINT, UINT, UDINT, REAL, TIME</td>
<td>Output value</td>
</tr>
</tbody>
</table>
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**

**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:

```
Value1 ---- REAL_TO_STRING ----> OUT Result_Str
```

**Representation in LD**
Representation applied to a real:

```
Value1 ---- REAL_TO_STRING ----> EN ENO ----> OUT Result_Str
```

**Representation in IL**
Representation applied to a real:

```
LD Value1
REAL_TO_STRING
ST Result_Str
```
Representation applied to a real:
Result_Str:= REAL_TO_STRING(Value1);

The following table describes the input parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value1</td>
<td>INT, DINT, REAL</td>
<td>Variable to be converted into character string format.</td>
</tr>
</tbody>
</table>

The following table describes the output parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result_Str</td>
<td>STRING</td>
<td>Result_Str is a character string whose length depends on the type of Value1:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 5 figures plus the sign making 6 characters for one INT (example: '-00045'),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 10 figures plus the sign, making 11 characters for one DINT (example: '-0000678911'),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 14 characters for a REAL (example: '-3.1234560e+25').</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Chapter Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>EFB Error Codes and Values</td>
<td>441</td>
</tr>
<tr>
<td>B</td>
<td>System objects</td>
<td>445</td>
</tr>
</tbody>
</table>
EFB Error Codes and Values

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:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tables of Error Codes for the Base Library</td>
<td>442</td>
</tr>
<tr>
<td>Common Floating Point Errors</td>
<td>444</td>
</tr>
</tbody>
</table>
### 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 error values created for EFBs of the Date & Time family.

<table>
<thead>
<tr>
<th>EFB name</th>
<th>Error code</th>
<th>ENO state in case of error</th>
<th>Error value in Dec</th>
<th>Error value in Hex</th>
<th>Error description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIVTIME</td>
<td>E_DIVIDE_BY_ZERO</td>
<td>F</td>
<td>-30176</td>
<td>0x8A20</td>
<td>Divide by zero</td>
</tr>
<tr>
<td>DIVTIME</td>
<td>E_NEGATIVE_INPUT_FOR_TIME_OPERATION</td>
<td>F</td>
<td>-30177</td>
<td>0x8A1F</td>
<td>A negative value cannot be converted to data type TIME</td>
</tr>
<tr>
<td>DIVTIME</td>
<td>E_ARITHMETIC_ERROR</td>
<td>F</td>
<td>-30170</td>
<td>0x8A26</td>
<td>Arithmetic error</td>
</tr>
<tr>
<td>DIVTIME</td>
<td>E_ERR_ARITHMETIC</td>
<td>F</td>
<td>-30003</td>
<td>0x8ACD</td>
<td>Arithmetic overflow (%S18 set)</td>
</tr>
<tr>
<td>DIVTIME</td>
<td>FP_ERROR</td>
<td>F</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MULTIME</td>
<td>E_ERR_ARITHMETIC</td>
<td>F</td>
<td>-30003</td>
<td>0x8ACD</td>
<td>Arithmetic overflow (%S18 set)</td>
</tr>
<tr>
<td>MULTIME</td>
<td>E_ARITHMETIC_ERROR_MUL_OV</td>
<td>F</td>
<td>-30172</td>
<td>0x8A24</td>
<td>Arithmetic error / Multiplication overflow</td>
</tr>
<tr>
<td>MULTIME</td>
<td>E_ARITHMETIC_ERROR_ADD_OV</td>
<td>F</td>
<td>-30173</td>
<td>0x8A23</td>
<td>Arithmetic error / Addition overflow</td>
</tr>
<tr>
<td>MULTIME</td>
<td>E_ARITHMETIC_ERROR_BIG_PAR</td>
<td>F</td>
<td>-30171</td>
<td>0x8A25</td>
<td>Arithmetic error / Parameter exceeds range</td>
</tr>
<tr>
<td>MULTIME</td>
<td>E_NEGATIVE_INPUT_FOR_TIME_OPERATION</td>
<td>F</td>
<td>-30177</td>
<td>0x8A1F</td>
<td>A negative value cannot be converted to data type TIME</td>
</tr>
<tr>
<td>MULTIME</td>
<td>FP_ERROR</td>
<td>F</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
## Table of error codes and errors values created for EFBs of the **Statistical** family.

<table>
<thead>
<tr>
<th>EFB name</th>
<th>Error code</th>
<th>ENO state in case of error</th>
<th>Error value in Dec</th>
<th>Error value in Hex</th>
<th>Error description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVE</td>
<td>E_INPUT_VALUE_OUT_OF_RANGE</td>
<td>F</td>
<td>-30183</td>
<td>0x8A19</td>
<td>Input value is out of range</td>
</tr>
<tr>
<td>AVE</td>
<td>E_DIVIDE_BY_ZERO</td>
<td>F</td>
<td>-30176</td>
<td>0x8A20</td>
<td>Divide by zero</td>
</tr>
<tr>
<td>AVE</td>
<td>FP_ERROR</td>
<td>F</td>
<td>-</td>
<td>-</td>
<td>See table Common Floating Point Errors, p. 444</td>
</tr>
<tr>
<td>AVE</td>
<td>E_ARITHMETIC_ERROR</td>
<td>F</td>
<td>-30170</td>
<td>0x8A26</td>
<td>Arithmetic error</td>
</tr>
<tr>
<td>AVE</td>
<td>E_FP_STATUS_FAILED</td>
<td>F</td>
<td>-30150</td>
<td>0x8A3A</td>
<td>Illegal floating point operation</td>
</tr>
<tr>
<td>AVE</td>
<td>E_ARITHMETIC_ERROR_MUL_OV</td>
<td>F</td>
<td>-30172</td>
<td>0x8A24</td>
<td>Arithmetic error / Multiplication overflow</td>
</tr>
<tr>
<td>AVE</td>
<td>E_ARITHMETIC_ERROR_ADD_OV</td>
<td>F</td>
<td>-30173</td>
<td>0x8A23</td>
<td>Arithmetic error / Addition overflow</td>
</tr>
<tr>
<td>AVE</td>
<td>E_ARITHMETIC_ERROR_BIG_PAR</td>
<td>F</td>
<td>-30171</td>
<td>0x8A25</td>
<td>Arithmetic error / Parameter exceeds range</td>
</tr>
<tr>
<td>AVE</td>
<td>E_ARITHMETIC_ERROR_UNSIGN_OV</td>
<td>F</td>
<td>-30174</td>
<td>0x8A22</td>
<td>Arithmetic error / Unsigned overflow</td>
</tr>
<tr>
<td>MAX</td>
<td>FP_ERROR</td>
<td>F</td>
<td>-</td>
<td>-</td>
<td>See table Common Floating Point Errors, p. 444</td>
</tr>
<tr>
<td>MIN</td>
<td>FP_ERROR</td>
<td>F</td>
<td>-</td>
<td>-</td>
<td>See table Common Floating Point Errors, p. 444</td>
</tr>
<tr>
<td>MUX</td>
<td>E_SELECTOR_OUT_OF_RANGE</td>
<td>F</td>
<td>-30175</td>
<td>0x8A21</td>
<td>Selector is out of range</td>
</tr>
</tbody>
</table>
Common Floating Point Errors

Introduction

The following table shows the common error codes and error values created for floating point errors.

Table of common floating point errors

<table>
<thead>
<tr>
<th>Error codes</th>
<th>Error value in Dec</th>
<th>Error value in Hex</th>
<th>Error description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP_ERROR</td>
<td>-30150</td>
<td>0x8A3A</td>
<td>Base value (not appearing as an error value)</td>
</tr>
<tr>
<td>E_FP_STATUS_FAILED_IE</td>
<td>-30151</td>
<td>0x8A39</td>
<td>Illegal floating point operation</td>
</tr>
<tr>
<td>E_FP_STATUS_FAILED_DE</td>
<td>-30152</td>
<td>0x8A38</td>
<td>Operand is denormalized - not a valid REAL number</td>
</tr>
<tr>
<td>E_FP_STATUS_FAILED_ZE</td>
<td>-30154</td>
<td>0x8A36</td>
<td>Illegal divide by zero</td>
</tr>
<tr>
<td>E_FP_STATUS_FAILED_ZE_IE</td>
<td>-30155</td>
<td>0x8A35</td>
<td>Illegal floating point operation / Divide by zero</td>
</tr>
<tr>
<td>E_FP_STATUS_FAILED_OE</td>
<td>-30158</td>
<td>0x8A32</td>
<td>Floating point overflow</td>
</tr>
<tr>
<td>E_FP_STATUS_FAILED_OE_IE</td>
<td>-30159</td>
<td>0x8A31</td>
<td>Illegal floating point operation / Overflow</td>
</tr>
<tr>
<td>E_FP_STATUS_FAILED_OE_ZE</td>
<td>-30162</td>
<td>0x8A2E</td>
<td>Floating point overflow / Divide by zero</td>
</tr>
<tr>
<td>E_FP_STATUS_FAILED_OE_ZE_IE</td>
<td>-30163</td>
<td>0x8A2D</td>
<td>Illegal floating point operation / Overflow / Divide by zero</td>
</tr>
<tr>
<td>E_FP_NOT_COMPARABLE</td>
<td>-30166</td>
<td>0x8A2A</td>
<td>Internal error</td>
</tr>
</tbody>
</table>
System objects

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:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>System bit introduction</td>
<td>446</td>
</tr>
<tr>
<td>Description of system bits %S9 to %S13</td>
<td>447</td>
</tr>
<tr>
<td>Description of system bits %S15 to %S21</td>
<td>448</td>
</tr>
<tr>
<td>Description of system words %SW12 to %SW18</td>
<td>451</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Bit Symbol</th>
<th>Function</th>
<th>Description</th>
<th>Initial state</th>
<th>Quantum</th>
<th>Premium Atrium</th>
</tr>
</thead>
</table>
| %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:

<table>
<thead>
<tr>
<th>Bit Symbol</th>
<th>Function</th>
<th>Description</th>
<th>Initial state</th>
<th>Quantum</th>
<th>Premium</th>
<th>Atrium</th>
</tr>
</thead>
<tbody>
<tr>
<td>%S15</td>
<td>STRINGERROR</td>
<td>Character string fault</td>
<td>0</td>
<td>YES</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>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 set to 1. This bit must be reset to 0 by the application.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>%S16</td>
<td>IOERRTSK</td>
<td>Task input/output fault</td>
<td>1</td>
<td>YES</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>%S17</td>
<td>CARRY</td>
<td>Rotate shift output</td>
<td>0</td>
<td>YES</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Normally at 0. During a rotate shift operation, this takes the state of the outgoing bit.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
System objects

<table>
<thead>
<tr>
<th>Bit Symbol</th>
<th>Function</th>
<th>Description</th>
<th>Initial state</th>
<th>Quantum</th>
<th>Premium Atrium</th>
</tr>
</thead>
<tbody>
<tr>
<td>%S18</td>
<td>OVERFLOW</td>
<td>Overflow or arithmetic error</td>
<td>0</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Normally set to 0, this is set to 1 in the event of a capacity overflow if there is:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• a result greater than + 32 767 or less than - 32 768, in single length,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• result greater than + 65 535, in unsigned integer,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• a result greater than + 2 147 483 647 or less than - 2 147 483 648, in double length,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• result greater than +4 294 967 296, in double length or unsigned integer,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• real values outside limits,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• division by 0,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• the root of a negative number,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• forcing to a non-existent step on a drum.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• stacking up of an already full register, emptying of an already empty register.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>When the %S18 bit switches to 1, the application stops in error state if the %S78 bit has been to set to 1.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>%S19</td>
<td>OVERRUN</td>
<td>Task period overrun (periodical scanning)</td>
<td>0</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## System objects

### %S20 INDEXOVF

<table>
<thead>
<tr>
<th>Bit symbol</th>
<th>Function</th>
<th>Description</th>
<th>Initial state</th>
<th>Quantum</th>
<th>Premium Atrium</th>
</tr>
</thead>
<tbody>
<tr>
<td>%S20 INDEXOVF</td>
<td>Index overflow</td>
<td>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 set to 1.</td>
<td>0</td>
<td>YES</td>
<td>YES</td>
</tr>
</tbody>
</table>

### %S21 1RSTTASKRUN

<table>
<thead>
<tr>
<th>Bit symbol</th>
<th>Function</th>
<th>Description</th>
<th>Initial state</th>
<th>Quantum</th>
<th>Premium Atrium</th>
</tr>
</thead>
<tbody>
<tr>
<td>%S21 1RSTTASKRUN</td>
<td>First task cycle</td>
<td>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. <strong>Notes:</strong> the bit %S21 does not have the same meaning in PL7 as in Unity Pro.</td>
<td>0</td>
<td>YES</td>
<td>YES</td>
</tr>
</tbody>
</table>

---

**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

<table>
<thead>
<tr>
<th>Word Symbol</th>
<th>Function</th>
<th>Description</th>
<th>Initial state</th>
<th>Quantum</th>
<th>Premium Atrium</th>
</tr>
</thead>
<tbody>
<tr>
<td>%SW12</td>
<td>UTWPORTADDR</td>
<td>Uni-Telway terminal port address</td>
<td>-</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Uni_Telway address of terminal port (in slave mode) as defined in the configuration and loaded into this word on cold start. <strong>Note:</strong> The modification of the value of this word is not taken into account by the system.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>%SW13</td>
<td>XWAYNETWADDR</td>
<td>Main address of the station</td>
<td>254 (16#00FE)</td>
<td>NO</td>
<td>YES</td>
</tr>
</tbody>
</table>
|             |           | 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). | | | |
| %SW14       | OSCOMMVERS | Commercial version of PLC processor | - | YES | YES |
|             |           | This word contains the commercial version of the PLC processor. **Example:** 16#0135  
|             |           | version: 01  
|             |           | issue number: 35 | | | |
| %SW15       | OSCOMMPATCH | PLC processor patch version | - | YES | YES |
|             |           | 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. | | | |
| %SW16       | OSINTVERS | Firmware version of PLC processor | - | YES | YES |
|             |           | This word contains the Firmware version of the PLC processor.  
|             |           | **Example:** 16#0143  
|             |           | version: 01  
|             |           | issue number: 43 | | | |
### System objects

<table>
<thead>
<tr>
<th>Word symbol</th>
<th>Function</th>
<th>Description</th>
<th>Initial state</th>
<th>Quantum</th>
<th>Premium Atrium</th>
</tr>
</thead>
</table>
| %SW17       | 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       | 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.

**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 addresses 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 is the abbreviation of Analog Output data type and is used when processing analog values. The %MW addresses 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, 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
      ANY_ARRAY_UINT
      ANY_ARRAY_TIME
      ANY_ARRAY_ANY_BIT
      ANY_ARRAY_DWORD
      ANY_ARRAY_WORD
      ANY_ARRAY_BYTE
      ANY_ARRAY_BOOL
      ANY_ARRAY_STRING
      ANY_ARRAY_STRING
      ANY_ARRAY_ANY_DATE
      ANY_ARRAY_DATE
      ANY_ARRAY_DATE_AND_TIME
      ANY_ARRAY_ANY_TIME
      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.

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).

The BOOL type word extract bit, for example: %MW10.4.

BYTE

When 8 bits are put together, this is called 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

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:

<table>
<thead>
<tr>
<th>Field</th>
<th>Limits</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year</td>
<td>[1990,2099]</td>
<td>Year</td>
</tr>
<tr>
<td>Month</td>
<td>[01,12]</td>
<td>The left 0 is always displayed, but can be omitted at the time of entry</td>
</tr>
<tr>
<td>Day</td>
<td>[01,31]</td>
<td>For the months 01/03/05/07/08/10/12</td>
</tr>
<tr>
<td></td>
<td>[01,30]</td>
<td>For the months 04/06/09/11</td>
</tr>
<tr>
<td></td>
<td>[01,29]</td>
<td>For the month 02 (leap years)</td>
</tr>
<tr>
<td></td>
<td>[01,28]</td>
<td>For the month 02 (non leap years)</td>
</tr>
</tbody>
</table>

DATE_AND_TIME

see DT
**Glossary**

**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 abbreviation 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 abbreviation 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:

<table>
<thead>
<tr>
<th>Field</th>
<th>Limits</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year</td>
<td>[1990,2099]</td>
<td>Year</td>
</tr>
<tr>
<td>Month</td>
<td>[01,12]</td>
<td>The left 0 is always displayed, but can be omitted at the time of entry</td>
</tr>
<tr>
<td>Day</td>
<td>[01,31]</td>
<td>For the months 01\03\05\07\08\10\12</td>
</tr>
<tr>
<td></td>
<td>[01,30]</td>
<td>For the months 04\06\09\11</td>
</tr>
<tr>
<td></td>
<td>[01,29]</td>
<td>For the month 02 (leap years)</td>
</tr>
<tr>
<td></td>
<td>[01,28]</td>
<td>For the month 02 (non leap years)</td>
</tr>
<tr>
<td>Hour</td>
<td>[00,23]</td>
<td>The left 0 is always displayed, but can be omitted at the time of entry</td>
</tr>
<tr>
<td>Minute</td>
<td>[00,59]</td>
<td>The left 0 is always displayed, but can be omitted at the time of entry</td>
</tr>
<tr>
<td>Second</td>
<td>[00,59]</td>
<td>The left 0 is always displayed, but can be omitted at the time of entry</td>
</tr>
</tbody>
</table>
**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:

<table>
<thead>
<tr>
<th>Base</th>
<th>Lower limit</th>
<th>Upper limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hexadecimal</td>
<td>16#0</td>
<td>16#FFFFFFFF</td>
</tr>
<tr>
<td>Octal</td>
<td>8#0</td>
<td>8#37777777777</td>
</tr>
<tr>
<td>Binary</td>
<td>2#0</td>
<td>2#111111111111111111111111</td>
</tr>
</tbody>
</table>

Representation examples:

<table>
<thead>
<tr>
<th>Data content</th>
<th>Representation in one of the bases</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000000000001010110110011011110</td>
<td>16#ADCDE</td>
</tr>
<tr>
<td>00000000000000010000000000000000</td>
<td>8#200000</td>
</tr>
<tr>
<td>0000000000001010111110011011110</td>
<td>2#10101011110011011110</td>
</tr>
</tbody>
</table>

**EBOOL**

EBOOL is the abbreviation 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 Enable, 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 is 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 Error Notification, 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”.

**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
**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:

<table>
<thead>
<tr>
<th>Decimal</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCD</td>
<td>0000 0001 0010 0011 0100 0101 0110 0111 1000 1001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gray</td>
<td>0000 0001 0011 0010 0110 0111 0101 0100 1100 1101</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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 \(-3.402824\times10^{38}\), the symbol \(-INF\) (for -infinite) is displayed,
- greater than \(+3.402824\times10^{38}\), 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: \(-2^{15}\) to \(2^{15} - 1\).
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.

### 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.).)

### 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.
**Multiple Token**

Operating mode of an SFC. In multitoken mode, the SFC may possess several active steps at the same time.

**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). **QNaNs** are allowed to propagate through most arithmetic operations without signaling an exception. **SNaNs** generally signal an invalid-operation exception whenever they appear as operands in arithmetic operations (See %SW17 and %S18).
Glossary

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.

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.
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 considered 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). QNaNs are allowed to propagate through most arithmetic operations without signaling an exception. SNaNs 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
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.

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:

<table>
<thead>
<tr>
<th>Field</th>
<th>Limits</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hour</td>
<td>[00,23]</td>
<td>The left 0 is always displayed, but can be omitted at the time of entry</td>
</tr>
<tr>
<td>Minute</td>
<td>[00,59]</td>
<td>The left 0 is always displayed, but can be omitted at the time of entry</td>
</tr>
<tr>
<td>Second</td>
<td>[00,59]</td>
<td>The left 0 is always displayed, but can be omitted at the time of entry</td>
</tr>
</tbody>
</table>

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.

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.
Glossary

**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.

**Variable**
Memory entity of the type BOOL, WORD, DWORD, etc., whose contents can be modified by the program during execution.

**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:

<table>
<thead>
<tr>
<th>Base</th>
<th>Lower limit</th>
<th>Upper limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hexadecimal</td>
<td>16#0</td>
<td>16#FFFF</td>
</tr>
<tr>
<td>Octal</td>
<td>8#0</td>
<td>8#177777</td>
</tr>
<tr>
<td>Binary</td>
<td>2#0</td>
<td>2#1111111111111111</td>
</tr>
</tbody>
</table>

Representation examples

<table>
<thead>
<tr>
<th>Data content</th>
<th>Representation in one of the bases</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000000011010011</td>
<td>16#D3</td>
</tr>
<tr>
<td>1010101010101010</td>
<td>8#125252</td>
</tr>
<tr>
<td>00000000011010011</td>
<td>2#11010011</td>
</tr>
</tbody>
</table>
Symbols
%S10, 447
%S11, 447
%S12, 447
%S13, 447
%S15, 448
%S16, 448
%S17, 448
%S18, 449
%S19, 449
%S20, 450
%S21, 450
%S9, 447
%SD18, 452
%SW12, 451
%SW13, 451
%SW14, 451
%SW15, 451
%SW16, 451
%SW17, 452
***_TO_STRING, 437

Numerics
100MSCOUNTER, 452
1RSTSCANRUN, 447
1RSTTASKRUN, 450

A
ABS, 215
Absolute value computation
   ABS, 215
ACOS, 217
ACOS_REAL, 217
ADD, 219
ADD_***_***, 41
ADD_***_TIME, 167
ADD_TIME, 221
Addition
   ADD, 219
   ADD_TIME, 221
Addition of a duration to a date
   ADD_***_TIME, 167
Addition of a number to elements of a table
   or addition of two tables
   ADD_***_***, 41
AND, 179
AND function
   AND, 179
AND_***_***, 45
Arc Cosine
   ACOS, 217
   ACOS_REAL, 217
Arc Sine
   ASIN, 223
   ASIN_REAL, 223
Arc tangent
   ATAN, 225
   ATAN_REAL, 225
## Index

**Ascending or descending sort**
- `SORT_***`, 103
- `ASIN`, 223
- `ASIN_REAL`, 223

**Assignment**
- `MOVE`, 247

**Assignment to tables**
- `MOVE_***_***`, 83
- `ATAN`, 225
- `ATAN_REAL`, 225
- `AVE`, 267

**Averaging**
- `AVE`, 267

**B**
- base 10 logarithm
  - `LOG`, 243
  - `LOG_REAL`, 243
- BCD to INT
  - `BCD_TO_INT`, 339

**Binary selection**
- `SEL`, 287

**Bistable function block, reset dominant**
- `RS`, 199

**Bistable function block, set dominant**
- `SR`, 207

**Bit to byte**
- `BIT_TO_BYTE`, 341

**Bit to word**
- `BIT_TO_WORD`, 345

**Block types**
- 20

**Boolean to***
- `BOOL_TO_***`, 347

**Byte as word**
- `BYTE_AS_WORD`, 349

**Byte to***
- `BYTE_TO_***`, 355

**Byte to bit**
- `BYTE_TO_BIT`, 351

**C**
- Calculates the time difference between two dates or times
  - `SUB_***_***`, 173
  - `CARRY`, 448

**Character string**
- `INSERT_INT`, 299
- `LEN_INT`, 305

**Character strings**
- `CONCAT_STR`, 291
- `DELETE_INT`, 293
- `EQUAL_STR`, 295
- `FIND_INT`, 297
- `LEFT_INT`, 303
- `MID_INT`, 307
- `REPLACE_INT`, 309
- `RIGHT_INT`, 313

**Comparison**
- `EQ`, 145
- `GE`, 147
- `GT`, 151
- `LE`, 155
- `LT`, 159
- `NE`, 163

**Comparison of two character strings**
- `EQUAL_STR`, 295

**Comparison of two tables**
- `EQUAL_***`, 57

**Concatenation of two character strings**
- `CONCAT_STR`, 291

**Concatenation of two integers**
- `INT_AS_DINT`, 379

**Conditional FFB Call**
- 25
Conversion
  ***_TO_STRING, 437
  BCD_TO_INT, 339
  DATE_TO_STRING, 359
  DBCD_TO_***, 361
  DEG_TO_RAD, 363
  DINT_TO_DBCD, 371
  DT_TO_STRING, 373
  GRAY_TO_INT, 377
  INT_AS_DINT, 379
  INT_TO_BCD, 385
  INT_TO_DBCD, 387
  RAD_TO_DEG, 389
  STRING_TO_DINT, 399
  STRING_TO_INT, 399
  STRING_TO_REAL, 399
  TIME_TO_STRING, 405
  TOD_TO_STRING, 407
Conversion of a character string to a number
  STRING_TO_***, 399
Conversion of a double integer into DBCD
  DINT_TO_DBCD, 371
Conversion of a variable in DATE format into a character string
  DATE_TO_STRING, 359
Conversion of a variable in DT format into a character string
  DT_TO_STRING, 373
Conversion of a variable in TIME format into a character string
  TIME_TO_STRING, 405
Conversion of a variable in TOD format into a character string
  TOD_TO_STRING, 407
Conversion of a variable into a character string
  ***_TO_STRING, 437
Conversion of an integer in Gray code into a binary coded integer
  GRAY_TO_INT, 377
Conversion of an integer into a double BCD integer
  INT_TO_DBCD, 387
Conversion of an integer into BCD
  INT_TO_BCD, 385

Conversion of BCD into binary
  BCD_TO_INT, 339
Conversion of DBCD into binary
  DBCD_TO_***, 361
Conversion of degrees to radians
  DEG_TO_RAD, 363
Conversion of radians to degrees
  RAD_TO_DEG, 389
Copy on tables
  COPY_***_***, 49
COS, 227
COS_REAL, 227
Cosine
  COS, 227
  COS_REAL, 227
CTD, 317
CTD_***, 317
CTU, 321
CTU_***, 321
CTUD, 325
CTUD_***, 325

D
Date & Time
  DIVTIME, 169
  MULTIME, 171
Date and time management
  ADD_***_TIME, 167
  SUB_***_***, 173
  SUB_***_TIME, 175
  DATE_TO_STRING, 359
  DBCD_TO_***, 361
  DEC, 229
Decrementation of a variable
  DEC, 229
  DEG_TO_RAD, 363
  DELETE_INT, 293
Deletion of a sub-string of characters
  DELETE_INT, 293
Derived function block, 20
Detection of all edges
  TRIGGER, 209
Detection of Falling Edge
  FE, 183
Index

Detection of falling edges
  F_TRIG, 181
Detection of Rising Edge
  RE, 191
Detection of rising edges
  R_TRIG, 189
DINT_AS_WORD, 365
DINT_TO_***, 367
DINT_TO_DBCD, 371
DINT_TO_STRING, 437
DIV, 231
DIV_***_***, 53
Division
  DIV, 231
  DIVTIME, 169
Division and Modulo
  DIVMOD, 233
Division of tables
  DIV_***_***, 53
DIVMOD, 233
DIVTIME, 169
Down counter
  CTD, 317
  CTD_***, 317
DT_TO_STRING, 373
DWORD_TO_***, 375

E
Elementary Function, 20
Elementary function block, 20
EN, 24
ENO, 24
EQ, 145
Equal to
  EQ, 145
  EQUAL_***, 57
  EQUAL_STR, 295
Error Codes, 441
Error Values, 441
Exclusive OR between tables
  XOR_***_***, 113
Exclusive OR function
  XOR, 211
EXP, 235
EXP_REAL, 235

Exponentiation of one value by another value
  EXPT_REAL_***_***, 237
  EXPT_REAL_***, 237
  EXPT_REAL_DINT, 237
  EXPT_REAL_INT, 237
  EXPT_REAL_REAL, 237
  EXPT_REAL_UDINT, 237
  EXPT_REAL_UINT, 237
Extraction of a character string to the right
  RIGHT_INT, 313
Extraction of a sub-string of characters
  MID_INT, 307
Extraction of characters to the left
  LEFT_INT, 303

F
F_TRIG, 181
FE, 183
FIND_EQ_***, 61
FIND_EQP_***, 63
FIND_GT_***, 67
FIND_INT, 297
FIND_LT_***, 69
Finding a sub-string of characters
  FIND_INT, 297
First element of a table equal to a given value
  FIND_EQ_***, 61
First element of a table equal to a value starting from a given rank
  FIND_EQP_***, 63
First element of a table greater than a given value
  FIND_GT_***, 67
First element of a table less than a given value
  FIND_LT_***, 69
FLOATSTAT, 452
Index

G
GE, 147
GRAY_TO_INT, 377
Greater than
GT, 151
Greater than or equal to
GE, 147
GT, 151

I
INC, 239
Incrementation of a variable
INC, 239
INDEXOVF, 450
INSERT_INT, 299
Insertion of a sub-string of characters
INSERT_INT, 299
INT_AS_DINT, 379
INT_TO_***, 381
INT_TO_BCD, 385
INT_TO_DBCD, 387
INT_TO_STRING, 437
Integer regulation
PID_INT, 125
PWM_INT, 133
IOERR, 447
IOERRRTSK, 448

L
LE, 155
LEFT_INT, 303
LEN_INT, 305
Length of a table
LENGTH_***, 73
Length of character string
LEN_INT, 305
LENGTH_***, 73
Less than
LT, 159
Less than or equal to
LE, 155
LIMIT, 271

Limit
LIMIT, 271
Limit with Indicator
LIMIT_IND, 275
LIMIT_IND, 275
LN, 241
LN_REAL, 241
LOG, 243
LOG_REAL, 243
Logic
AND, 179
F_TRIG, 181
FE, 183
NOT, 185
OR, 187
R_TRIG, 189
RE, 191
RESET, 193
ROL, 195
ROR, 197
RS, 199
SET, 201
SHL, 203
SHR, 205
SR, 207
TRIGGER, 209
XOR, 211
Logical AND between tables and variables
AND_***_***, 45
Logical negation of tables
NOT_***, 91
Logical OR between tables and variables
OR_***_***, 95
LT, 159
Index

M

Math
ACOS, 217
ACOS_REAL, 217
ASIN, 223
ASIN_REAL, 223
ATAN, 225
ATAN_REAL, 225
COS, 227
COS_REAL, 227
EXP, 235
EXP_REAL, 235
INC, 229, 239
LN, 241
LN_REAL, 241
LOG, 243
LOG_REAL, 243
SIN, 255
SIN_REAL, 255
TAN, 263
TAN_REAL, 263

Mathematic
ADD_TIME, 221
SUB_TIME, 259

Mathematics
ABS, 215
ADD, 219
DIV, 231
DIVMOD, 233
EXPT_REAL_***, 237
MOD, 245
MOVE, 247
MUL, 249
NEG, 251
SIGN, 253
SQRT_***, 261
SUB, 257
MAX, 279
MAX_***, 75
Minimum value function
MIN, 281
Minimum value of table elements
MIN_***, 77
MOD, 245
MOD_***_***, 79
Modulo
MOD, 245
MOVE, 247
MOVE_***_***, 83, 85
Assignment to tables, 83
Table conversion, 85
MUL, 249
MUL_***_***, 87
MULTIME, 171
Multiplexer
MUX, 283
Multiplication
MUL, 249
MULTIME, 171
Multiplication of tables
MUL_***_***, 87
MUX, 283

N

Natural exponential
EXP, 235
EXP_REAL, 235
Natural logarithm
LN, 241
LN_REAL, 241
NE, 163
NEG, 251
Negation
NEG, 251
NOT, 185
NOT, 185
Not equal to
NE, 163
NOT_***, 91
O
OCCUR_***, 93
Occurrence of a value in a table
   OCCUR_***, 93
Off delay
   TOF, 329
On delay
   TON, 331
OR, 187
OR function
   OR, 187
OR_***_***, 95
OSCOMMPATCH, 451
OSCOMMVERS, 451
OSINTVERS, 451
OUTDIS, 447
OVERFLOW, 449
OVERRUN, 449

P
Permutation of the bytes of a table
   SWAP_***, 111
PID controller
   PID_INT, 125
PCLRUNNING, 447
Procedure, 20
Pulse
   TP, 333
Pulse width modulation
   PWM_INT, 133
PWM_INT, 133

R
R_TRIG, 189
RAD_TO_DEG, 389
RE, 191
REAL_AS_WORD, 391
REAL_TO_***, 393
REAL_TO_STRING, 437
REAL_TRUNC_***, 397
Regulation
   SERVO_INT, 137

S
SERVO_INT, 137
SERVO_INT, 137
SET, 201
Setting of a bit to 0
   RESET, 193
Setting of a bit to 1
   SET, 201
Shift left
   SHL, 203
Shift right
   SHR, 205
SIGN, 253
Sign evaluation
   SIGN, 253
SIN, 255
SIN_REAL, 255
Sine
   SIN, 255
   SIN_REAL, 255
Index

SORT_***, 103
SORT_***, 261
Square root
  SORT_***, 261
SR, 207
Statistics
  AVE, 267
  LIMIT, 271
  LIMIT_IND, 275
  MAX, 279
  MIN, 281
  MUX, 283
  SEL, 287
STRING_TO_***, 399
STRING_TO_DINT, 399
STRING_TO_INT, 399
STRING_TO_REAL, 399
STRINGERROR, 448
SUB, 257
SUB_***_***, 105, 173
SUB_***_TIME, 175
SUB_TIME, 259
Subtraction
  SUB, 257
  SUB_TIME, 259
Subtraction from tables
  SUB_***_***, 105
Subtraction of a duration from a date
  SUB_***_TIME, 175
Sum of table elements
  SUM_***, 109
  SUM_***, 109
  SWAP_***, 111

T
  Table conversion
    MOVE_***_***, 85
  Table functions
    ADD_***_***, 41
    AND_***_***, 45
    COPY_***_***, 49
    DIV_***_***, 53
    EQUAL_***_***, 57
    FIND_EQ_***_***, 61
    FIND_EQP_***_***, 63
    FIND_GT_***_***, 67
    FIND_LT_***_***, 69
    LENGTH_***_***, 73
    MAX_***_***, 75
    MIN_***_***, 77
    MOD_***_***_***, 79
    MOVE_***_***_***, 83, 85
    MUL_***_***_***, 87
    NOT_***_***, 91
    OCCUR_***_***, 93
    OR_***_***_***, 95
    ROL_***_***, 99
    ROR_***_***, 101
    SORT_***_***, 103
    SUB_***_***, 105
    SUM_ARINT, 109
    SWAP_***_***, 111
    XOR_***_***_***, 113

TAN, 263
TAN_REAL, 263
Tangent
  TAN, 263
  TAN_REAL, 263
TIME_AS_WORD, 401
TIME_TO_***, 403
TIME_TO_STRING, 405
Index

Timer & Counter
CTD, 317
CTD_***, 317
CTU, 321
CTU_***, 321
CTUD, 325
CTUD_***, 325
TOF, 329
TON, 331
TP, 333
TOD_TO_STRING, 407
TOF, 329
TON, 331
TP, 333
TRIGGER, 209
Type conversion
BIT_TO_BYTE, 341
BIT_TO_WORD, 345
BOOL_TO_***, 347
BYTE_AS_WORD, 349
BYTE_TO_***, 355
BYTE_TO_BIT, 351
DINT_AS_WORD, 365, 391
DINT_TO_***, 367
DWORD_TO_***, 375
INT_TO_***, 381
REAL_AS_WORD, 391
REAL_TO_***, 393
REAL_TRUNC_***, 397
TIME_AS_WORD, 401
TIME_TO_***, 403
TYPE_AS_WORD, 401
UDINT_AS_WORD, 409
UINT_TO_***, 411
UDINT_TO_***, 415
WORD_AS_BYTE, 419
WORD_AS_DINT, 421
WORD_AS_REAL, 423
WORD_AS_TIME, 425
WORD_AS_UDINT, 427
WORD_TO_***, 433
WORD_TO_BIT, 429

U
UDINT_AS_WORD, 409
UDINT_TO_***, 411
UINT_TO_***, 415
Unconditional FFB Call, 25
Up counter
CTU, 321
CTU_***, 321
Up/Down counter
CTUD, 325
CTUD_***, 325
UTWPORTADDR, 451

W
WDG, 447
WORD_AS_BYTE, 419
WORD_AS_DINT, 421
WORD_AS_REAL, 423
WORD_AS_TIME, 425
WORD_AS_UDINT, 427
WORD_TO_***, 433
WORD_TO_BIT, 429

X
XOR, 211
XOR_***_***, 113
XWAYNETWADDR, 451