Kollmorgen Automation Suite

KAS Reference Manual - PLC Library

Document Edition: V, December 2022

Valid for KAS Software Revision 4.00

Part Number: 959717

For safe and proper use, follow these instructions. Keep for future use.

KOLLMORGEN

1 Trademarks and Copyrights

Copyrights

Trademarks

- KAS and AKD are registered trademarks of Kollmorgen.
- Kollmorgen is part of the Altra Industrial Motion Company.
- EnDat is a registered trademark of Dr. Johannes Heidenhain GmbH
- EtherCAT is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH
- Ethernet/IP is a registered trademark of ODVA, Inc.
- Ethernet/IP Communication Stack: copyright (c) 2009, Rockwell Automation
- HIPERFACE is a registered trademark of Max Stegmann GmbH
- PROFINET is a registered trademark of PROFIBUS and PROFINET International (PI)
- SIMATIC is a registered trademark of SIEMENS AG
- Windows is a registered trademark of Microsoft Corporation
- PLCopen is an independent association providing efficiency in industrial automation.
- Codemeter is a registered trademark of WIBU-Systems AG.
- SyCon® is a registered trademark of Hilscher GmbH.

Kollmorgen Automation Suite is based on the work of:

- 7-zip (distributed under the terms of the LGPL and the BSD 3-clause licenses - see terms)
- The C++ Mathematical Expression Library (distributed under the MIT License)
- curl software library
- JsonCpp software (distributed under the MIT License – see terms)
- Mongoose software (distributed under the MIT License - see terms)
- Qt cross-platform SDK (distributed under the terms of the LGPL3; Qt source is available on KDN)
- Qwt project (distributed under the terms of the Qwt License)
- U-Boot, a universal boot loader is used by the AKD PDMM and PCMM (distributed under the terms of the GNU General Public License). The U-Boot source files, copyright notice, and readme are available on the distribution disk that is included with the AKD PDMM and PCMM.
- Zlib software library

All other product and brand names listed in this document may be trademarks or registered trademarks of their respective owners.

Disclaimer

The information in this document (Version V published on 12/7/2022) is believed to be accurate and reliable at the time of its release. Notwithstanding the foregoing, Kollmorgen assumes no responsibility for any damage or loss resulting from the use of this help, and expressly disclaims any liability or damages for loss of data, loss of use, and property damage of any kind, direct, incidental or consequential, in regard to or arising out of the performance or form of the materials presented herein or in any software programs that accompany this document.

All timing diagrams, whether produced by Kollmorgen or included by courtesy of the PLCopen organization, are provided with accuracy on a best-effort basis with no warranty, explicit or implied, by Kollmorgen. The user releases Kollmorgen from any liability arising out of the use of these timing diagrams.

2 Table of Contents

3 Programming Languages

This section provides information about the syntax, structure, and use of declarations and statements supported by the KAS IDE application language.

These are the available programming languages of the IEC 61131-3 standard:

- "Free Form Ladder Diagram (FFLD)" (➜ p. 47)
- "Function Block Diagram (FBD)" (➜ p. 41)
- "Instruction List (IL)" (➜ p. 42)
- "Sequential Function Chart (SFC)" (➜ p. 35)
- "Structured Text (ST)" (➜ p. 44)

A language must be selected for each program or User-Defined Function Block of the application.

NOTE

When using FFLD or FBD languages, review Use of ST Expressions in Graphic Language.

3.1 Sequential Function Chart (SFC)

The SFC language is a state diagram.

- Graphical steps are used to represent stable states.
- Transitions describe the conditions and events that lead to a change of state.
- Using SFC simplifies the programming of sequential operations because it saves a lot of variables and tests just for maintaining the program context.

IMPORTANT

Do not use SFC as a decision diagram.
Using a step as a point of decision and transitions as conditions in an algorithm must never appear in an SFC chart.
Using SFC as a decision language leads to poor performance and complicate charts.
ST must be preferred when programming a decision algorithm that has no sense in term of program state.

These are basic components of an SFC chart:

Chart	Programming
• SFC Steps • SFC Transitions • Create SFC Parallel Branches • Jump to an SFC Step	• Actions in an SFC Step • Timeout on an SFC Step • Condition of an SFC Transition

The KAS IDE fully supports SFC programming with several hierarchical levels of charts (e.g., a chart that controls another chart).

Working with a hierarchy of SFC charts is an easy and powerful way for managing complex sequences and saves performances at runtime.

See these sections for more information:

- "Hierarchy of SFC Programs" (➜ p. 40)
- "Control an SFC Child Program" (➜ p. 40)

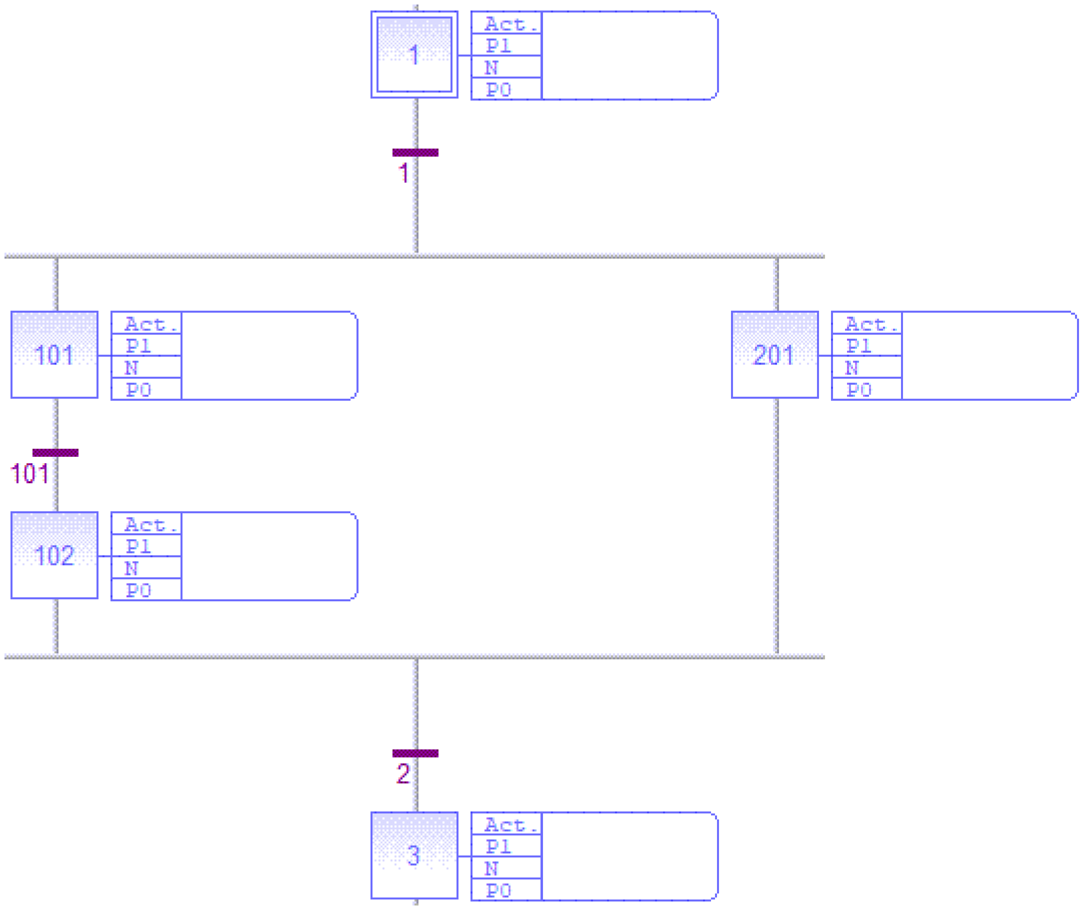
3.1.1 SFC Execution at Runtime

SFC programs are executed sequentially within a target cycle according to the order defined when entering programs in the hierarchy tree.

- A parent SFC program is executed before its children.
  - This implies that when a parent starts or stops a child, the corresponding actions in the child program are performed during the same cycle.
- In a chart, all valid transitions are evaluated first and then actions of active steps are performed.
  - The chart is evaluated from the left to the right and from the top to the bottom.

Example

Execution order:



- Evaluate transitions:
- Manage steps:

-

The initial steps define the initial status of the program when it is started.

- All top level (main) programs are started when the application starts.
- Child programs are explicitly started from action blocks within the parent programs.

The evaluation of transitions leads to changes of active steps, according to these rules:

- A transition is crossed if:
  - Its condition is TRUE **and** all steps linked to the top of the transition (before) are active.

- When a transition is crossed:
  - All steps linked to the top of the transition (before) are deactivated.
  - All steps linked to the bottom of the transition (after) are activated.

3.1.1.1 Divergence

- All conditions are considered as **exclusive**, according to a left-to-right priority order.
  - It means a transition is considered as FALSE if at least one of the transitions connected to the same divergence on its left side is TRUE.

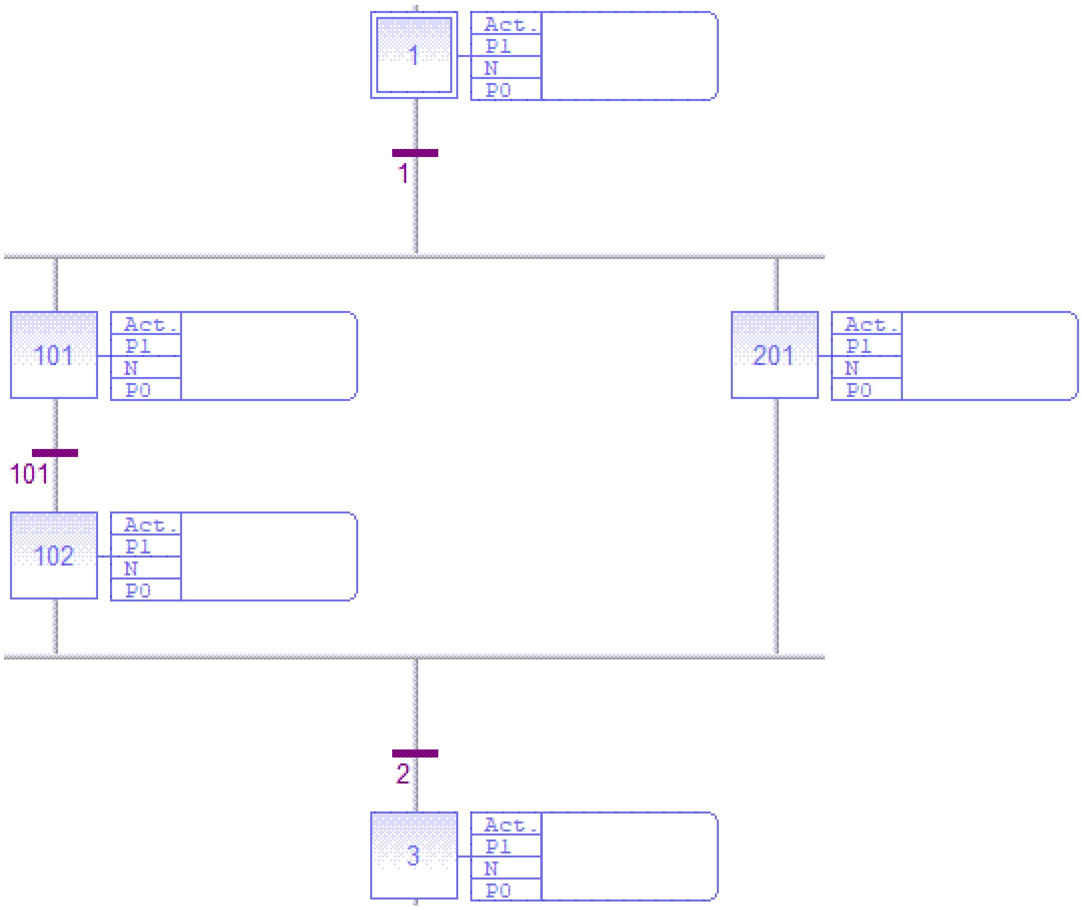
3.1.1.1.1 Order of Action Block Execution

For a given cycle, if a transition is:

- FALSE, the N-action blocks are evaluated for all active steps waiting on that transition.
- TRUE, the P0 action blocks are evaluated for all active steps waiting on that transition, followed by the P1 and N steps for all steps waiting on that transition.
  - The steps that were waiting on that transition are then marked as active.

Example

Using this SFC:



The order of action block execution for a given cycle is:

Incoming State	Evaluating Transition	If Transition is FALSE	If Transition is TRUE
First Cycle	N/A	[1] P1 [1] N	
Transition 1. Not yet TRUE.	Transition 1	[1] N	[1] P0 [101] P1 [101] N [201] P1 [201] N
Passed transition 1. Transition 101 not yet TRUE.	Transition 101	[101] N [201] N	[101] P0 [201] N [102] P1 [102] N
Passed transitions 1 and 101 Transition 2 not yet TRUE.	Transition 2	[201] N [102] N	[201] P0 [102] P0 [3] P1 [3] N

IMPORTANT

Execution of SFC in the IEC 61131-3 target is sampled according to the target cycles.
When a transition is crossed within a cycle, these steps are activated.

The evaluation of the chart continues in the next cycle.
If several consecutive transitions are TRUE within a branch, only one of them is crossed within one target cycle.

IMPORTANT

Some runtime systems may not support exclusivity of the transitions within an divergence.
See the OEM instructions for more information about SFC support.

3.1.2 Hierarchy of SFC Programs

Each SFC program can have one or more child programs.

- Child programs are written in SFC.
- They are started or stopped in the actions of the parent program.
- The number of hierarchy levels must not exceed 19.
- A child program can also have children.
  - When a child program is stopped, its children are also stopped.
  - When a child program is started, it must start its children.
  - A child program is controlled (started or stopped) from the action blocks of its parent program.
  - Designing a child program is:
    - a simple way to program an action block in SFC language.
    - very useful for designing a complex process and separate operations due to different aspects of the process.
      - Example: It is common to manage the execution modes in a parent program and to handle details of the process operations in child programs.

3.1.3 Control an SFC Child Program

Controlling a child program can be simply achieved by specifying the name of the child program as an action block in a step of its parent program.

These are possible qualifiers that can be applied to an action block for handling a child program:

`Child (N);`	Starts the child program when the step is activated and stops (kills) it when the step is deactivated.
`Child (S);`	Starts the child program when the step is activated. Initial steps of the child program are activated.
`Child (R);`	Stops (kills) the child program when the step is activated. All active steps of the child program are deactivated.

Alternatively, use these statements in an action block programmed in ST language.

In this table, "`prog`" represents the name of the child program:

`GSTART (prog);`	Starts the child program when the step is activated. Initial steps of the child program are activated.
`GKILL (prog);`	Stops (kills) the child program when the step is activated. All active steps of the child program are deactivated.
`GFREEZE (prog);`	Suspends the execution of a child program.
`GRST (prog);`	Restarts a program suspended by a `GFREEZE` command.

Use the `GSTATUS` function in expressions.

This function returns the current state of a child SFC program:

`GSTATUS (prog)`	Returns the current state of a child SFC program: 0: program is inactive 1: program is active 2: program is suspended

NOTE

When a child program is started by its parent program, it keeps the "inactive" status until it is executed (further in the cycle).
If a child program is started in an SFC chart, GSTATUS returns 1 (active) on the next cycle.

3.2 Function Block Diagram (FBD)

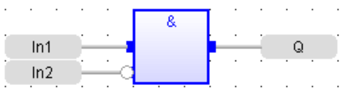
A function block diagram is a data flow between constant expressions or variables and operations represented by rectangular blocks.

- Operations can be basic operations, function calls, or function block calls.
- See Use of ST Expressions in Graphic Language for more information.
- The name of the operation or function, or the type of function block is written within the block rectangle.
- With a function block call, the name of the called instance is written in the header of the block rectangle.

Example

3.2.1 Data Flow

- The data flow represents values of any data type.
  - All connections must be from input and outputs points having the same data type.
- With a Boolean connection, use a connection link terminated by a small circle.
  - This indicates a Boolean **negation** of the data flow.



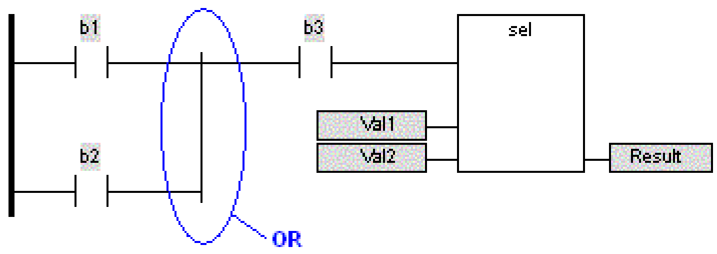
- The data flow must be understood from the left to the right and from the top to the bottom.
  - It is possible to use labels and jumps to change the default data flow execution.

3.2.2 FFLD Symbols

FFLD symbols can also be entered in FBD diagrams and linked to FBD objects.

- See these sections for information about components of the FFLD language:
  - "FFLD Contacts" (→ p. 49)
  - "FFLD Coils" (→ p. 50)
  - Power Rails
- Special vertical lines are available in FBD language for representing the merging of FFLD parallel lines.
  - Such vertical lines represent a OR operation between the connected inputs.

Example of an OR vertical line used in a FBD diagram:



See Also

- FBD Editor Keyboard Shortcuts
- Function Block Diagram (FBD) Editor
- Using the FBD Toolbar

3.3 Instruction List (IL)

This language is more appropriate when your algorithm refers to the Boolean algebra.

A program written in IL language is a list of *instructions*.

- Each instruction is written on one line of text.
- An instruction can have one or more *operands*.
- Operands are variables or constant expressions.
- Each instruction begins with a label, followed by a colon (:).
- Labels are used as destination for jump instructions.

KAS IDE allows you to mix ST and IL languages in textual program.

- ST Language is the default language.
- When you enter IL Language instructions, the program must be entered between `BEGIN_IL` and `END_IL` keywords.

Example

```
BEGIN_IL
FFLD    var1
ST      var2
END_IL
```

3.3.1 Comments

Comment text can be entered at the end of a line containing an instruction.

- Comment texts have no meaning for the execution of the program.
- Comment text must begin with `(*` and end with `*)`.
- Comments can be entered on empty lines (with no instruction) and on several lines (i.e., a comment text can include line breaks).
- Comment texts cannot be nested.

```
(* My comment *)
LD a
ST b    (* Store value in d *)
```

3.3.2 Data Flow

An IL Language complete statement is made of instructions for:

- first: evaluating an expression (called current result).
- then: use the current result for performing actions.

3.3.3 Evaluation of Expressions

The order of instructions in the program is the one used for evaluating expressions, unless parentheses are inserted.

This list is the available instructions for evaluation of expressions:

Instruction	Operand	Meaning
"Addition +" (➔ p. 89)	Numerical	Performs an addition of all inputs. Adds the operand and the current result.
AND ANDN &	Boolean	AND between the operand and the current result.
"Divide /" (➔ p. 90)	Numerical	Performs a division of all inputs. Divide the current result by the operand.
"EQ =" (➔ p. 107)	Numerical	Compares the current result with the operand.
"Assignment :=" (➔ p. 54)	Any type	Loads the operand in the current result.
Function Call	Functional Arguments	Calls a function.
"GE >=" (➔ p. 105)	Numerical	Compares the current result with the operand.
"GT >" (➔ p. 106)	Numerical	Compares the current result with the operand.
"LE <=" (➔ p. 110)	Numerical	Compares the current result with the operand.
"LT <" (➔ p. 111)	Numerical	Compares the current result with the operand.
"Multiply *" (➔ p. 99)	Numerical	Performs a multiplication of all inputs. Multiply the operand and the current result.
"NE <>" (➔ p. 109)	Numerical	Compares the current result with the operand.
OR / ORN	Boolean	Performs a logical OR of all inputs. OR between the operand and the current result.
"Parenthesis ( )" (➔ p. 68)		Changes the execution order.
"Subtraction -" (➔ p. 102)	Numerical	Performs a subtraction of all inputs. Subtract the operand from the current result.
"XOR / XORN" (➔ p. 87)	Boolean	XOR between the operand and the current result.

NOTE

Instructions suffixed by **N** uses the Boolean negation of the operand.

3.3.4 Actions

These instructions perform actions according to the value of current result.

Some of these instructions do not need a current result to be evaluated.

Instruction	Operand	Meaning
CAL	f. block	Calls a function block (no current result needed).
CALC	f. block	Calls a function block if the current result is TRUE.
CALNC / CALCN	f. block	Calls a function block if the current result is FALSE.
JMP	label	Jump to a label - no current result needed.
JMPC	label	Jump to a label if the current result is TRUE.
JMPNC / JMPCN	label	Jump to a label if the current result is FALSE.
R	Boolean	Sets the operand to FALSE if the current result is TRUE.
RET		Jump to the end of the current program - no current result needed.
RETC / RETNC / RETCN		Jump to the end of the current program if the current result is TRUE / FALSE.
S	Boolean	Sets the operand to TRUE if the current result is TRUE.
ST / STN	Any type	Stores the current result in the operand.

NOTE

Instructions suffixed by **N** uses the Boolean negation of the operand.

NOTE

An IL Language program cannot be called if there is no entry variable or if it's type is complex (e.g., array).

3.4 Structured Text (ST)

ST is a structured literal programming language.

- A ST program is a list of *statements*.
  - Each statement describes an action and must end with a semi-colon (`;`).
- The presentation of the text has no meaning for a ST program. You can insert blank characters and line breaks where you want in the program text.

3.4.1 Comments

Comment text can be entered anywhere in a ST program.

- Comment text:
  - Has no meaning for the execution of the program.
  - Must begin with "`(*`" and end with "`*)`".
  - Can be entered on several lines (i.e., a comment text can include line breaks).
  - Cannot be nested.

You can also use `//` to add a comment on a single line:

```
//My main comment
(* My comment *)
```

```
a := d + e;
(* A comment can also
be on several lines *)
b := d * e;

c := d - e; (* My comment *)
```

3.4.2 Expressions

Each statement describes an action and can include evaluation of complex expressions.

An expression is evaluated:

- From the left to the right.
- According to the default priority order of operators.
  - The default priority can be changed using parentheses "Parenthesis ( )" (→ p. 68).

Arguments of an expression can be:

- Declared Variables.
- Constant Expressions.
- Call a Function.

3.4.3 Statements

3.4.3.1 Basic Statements

These are the available basic statements that can be entered in a ST program:

- "Assignment :=" (→ p. 54) (assignment)
- Call a Function Block

3.4.3.2 Conditional Statements

These are the available conditional statements in ST language:

- "CASE OF ELSE END_CASE" (→ p. 57)

```
  Switch between enumerated statements according to an
expression.
  The selector can be any integer or a STRING.
CASE iChoice OF
0:
MyString := 'Nothing';
1 .. 2,5:
MyString := 'First case';
3,4:
MyString := 'Second case';
ELSE
MyString := 'Other case';
END_CASE;
```

- "IF THEN ELSE ELSIF END_IF" (➜ p. 63)

```
   Simple binary switch.
   One or several ELSIF are allowed.
IF a = b THEN
   c := 0;
ELSIF a < b THEN
   c := 1;
ELSE
   c := -1;
END_IF;
```

3.4.3.3 Loop Statements

These are the available statements for describing loops in ST language:

IMPORTANT

Loop instructions can lead to infinite loops that block the target cycle.
Never test the state of an input in the condition as the input will not be refreshed before the next cycle.

- "FOR TO BY END_FOR" (➜ p. 62)
  - Loops with FOR instructions are slow.
    Optimize your code by replacing such iterations with a WHILE statement.

```
  Iteration of statement execution.
  The BY statement is optional (default value is 1)
FOR iCount := 0 TO 100 BY 2 DO
MyVar := MyVar + 1;
END_FOR;
```

- "REPEAT UNTIL END_REPEAT" (➜ p. 69)

```
  Repeat a list of statements.
  Condition is evaluated on loop exit after the statements.
iCount := 0;
REPEAT
MyVar := MyVar + 1;
iCount := iCount + 1;
UNTIL iCount < 100 END_REPEAT;
```

- "WHILE DO END_WHILE" (➜ p. 73)

```
  Repeat a list of statements.
  Condition is evaluated on loop entry before the statements.
iCount := 0;
WHILE iCount < 100 DO
iCount := iCount +1;
MyVar := MyVar + 1;
END_WHILE;
```

3.4.3.4 Other Statements

These are some other statements in ST language:

- "WAIT / WAIT_TIME" (➜ p. 71) - Suspend the execution.
- "ON" (➜ p. 67) - Conditional execution of statements: provides a simpler syntax for checking the rising edge of a Boolean condition.

3.4.3.5 Helpful Features

This content is in the Copa-Data content but not ours - is it relevant for us?

There are some features to simplify the programming:

- A click on the "(" keyboard button automatically inserts the ")" symbol, if the previous word is an instance or a function name. The caret is put between the brackets.
- Hint: Function blocks and UDFBs cannot be used in ST Language without an instance name. If a word is located before a function block or a UDFB the feature is not available.
- A click on the "[" keyboard button automatically inserts the "]" symbol. The caret is put between the two [].

TIP

ST provides an automatic completion of typed words.
See Auto-completion of Wordsfor more information.

3.5 Free Form Ladder Diagram (FFLD)

A Ladder Diagram is a list of rungs.

- Each rung represents a Boolean data flow from a power rail on the left. The power rail represents the TRUE state. The data flow must be understood from the left to the right. Each symbol connected to the rung either changes the rung state or performs an operation.
- These are possible graphic items to be entered in FFLD diagrams:
  - Power Rails
  - Contacts and Coils
  - Operations, Functions and Function blocks, represented by rectangular blocks.
    - See Call a Function or Call a Function Block.
  - LABELS and Jumps JMP JMPC JMPNC JMPCN
  - Use of ST Expressions in Graphic Language

3.5.1 Use of EN Input and ENO Output for Blocks

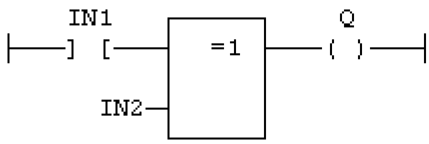
The rung state in a FFLD diagram is always Boolean.

- Blocks are connected to the rung with their first input and output.
  - This implies that special EN and ENO input and output are added to the block if its first input or output is not Boolean.
- The EN input is a condition.
  - It means that the operation represented by the block is not performed if the rung state (EN) is FALSE.
- The ENO output always represents the sane status as the EN input.
  - The rung state is not modified by a block having an ENO output.

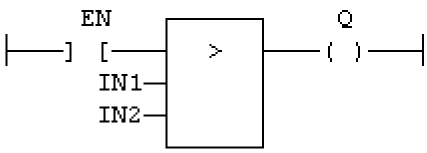
3.5.1.1 Examples

- This is an example of "XOR / XORN" (➜ p. 87) with Boolean inputs and outputs and requiring no EN or ENO pin.

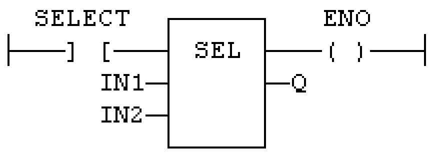
- First input is the rung.
- The rung is the output.



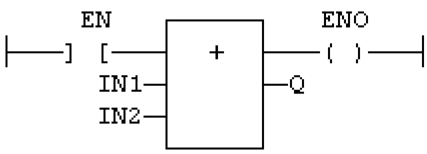
- This is an example of the "GT >" (→ p. 106) (greater than) with non-Boolean inputs and a Boolean output.
  - This block has an EN input in FFLD language.
  - The comparison is executed only if EN is TRUE.



- This is an example of the "SEL" (→ p. 131) with a first Boolean input but an integer output.
  - This block has an ENO output in FFLD language.
  - The input rung is the selector.
  - ENO has the same value as SELECT.



- This is an example of "Addition +" (→ p. 89) having only numerical arguments.
  - This block has both EN and ENO pins in FFLD language.
  - The addition is executed only if EN is TRUE.
  - ENO has the same value as EN.



3.5.2 Contacts and Coils

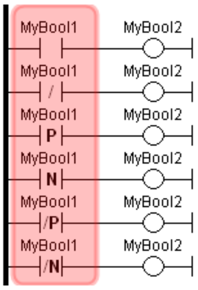
This is a list of the contact and coil types available:

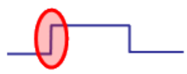
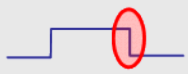
Contacts	Coils
Negative Transition -|N|-	Reset (Unlatch) -(R)-
Normally Closed -|/|-	De-energize -(/)-
Normally closed negative transition -|/N|-	Negative transition sensing coil -(N)-
Normally closed positive transition -|/P|-	Positive transition sensing coil -(P)-
Normally Open -| |-	Energize -( )-
Positive Transition -|P|-	Set (Latch) -(S)-

3.5.2.1 FFLD Contacts

Contacts are basic graphic elements of the FFLD language. A contact is associated with a Boolean variable which is displayed above the graphic symbol. A contact sets the state of the rung on its right-hand side, according to the value of the associated variable and the rung state on its left-hand side.

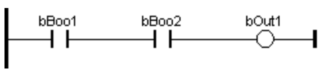
Below are the six possible contact symbols and how they change the flow:



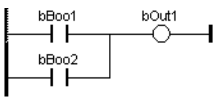
Contacts	Description
boolVariable -] [-	**Normal**: The flow on the right is the Boolean AND operation between: (1) the flow on the left and (2) the associated variable.
boolVariable -]/[-	**Negated**: The flow on the right is the Boolean AND operation between: (1) the flow on the left and (2) the negation of the associated variable.
boolVariable -]P[-	**Positive Transition**: The flow on the right is TRUE when the flow on the left is TRUE and the associated variable is TRUE and was FALSE the last time this contact was scanned (rising edge). 
boolVariable -]N[-	**Negative Transition**: The flow on the right is TRUE when the flow on the left is TRUE and the associated variable is FALSE and was TRUE last time this contact was scanned (falling edge). 
boolVariable -]/P[-	**Normally Closed Positive Transition**: The flow on the right is TRUE when the flow on the left is TRUE and the associated variable does not change from FALSE to TRUE from the last scan of this contact to this scan (NOT rising edge).
boolVariable -]/N[-	**Normally Closed Negative Transition**: The flow on the right is TRUE when the flow on the left is TRUE and the associated variable does not change from TRUE to FALSE from the last scan of this contact to this scan (NOT falling edge).

Serialized and Parallel contacts

Two serial normal contacts represent an AND operation.



Two contacts in parallel represent an OR operation.

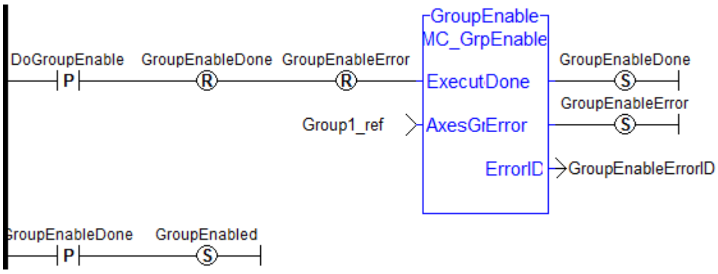


Transition Contacts

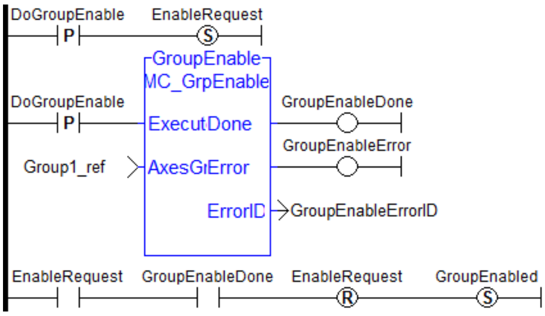
The transition contacts -|P|-, -|N -|/P|-, and -|/N|- compare the current state of the Boolean variable to the Boolean's state the last time the contact was scanned. This means that the Boolean variable could change states several times during a scan, but if it's back to the same state when the transition contact is scanned, the transition contact will not produce a TRUE. Also, some function blocks can complete immediately. Therefore a different approach, other than using transition contacts, is needed to determine if a function block completed successfully.

For example:

MC_GrpEnable executes and turns on its Done output immediately. In the following code, the GroupEnableDone positive transition contact will only provide a TRUE the first time MC_GrpEnable is executed. For all subsequent executions, the positive transition contact will not provide a TRUE since GroupEnableDone will be TRUE every time the contact is scanned.



To remedy this, the following code uses the SET and RESET of a Boolean (i.e. EnableRequest) to provide a way to detect each successful execution of the function block:



TIP

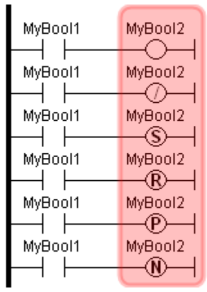
When a contact or coil is selected, you can press the **Spacebar** to change its type (normal, negated...).

When your application is running, you can select a contact and press the **Spacebar** to swap its value between TRUE and FALSE.

3.5.2.2 FFLD Coils

Coils are basic graphic elements of the FFLD language.

- A coil is associated with a Boolean variable displayed above the graphic symbol.
- A coil performs a change of the associated variable according to the flow on its left-hand side.
- The six possible coil symbols are:



<table>
<tr><th>Variable</th><th>Coils</th><th>Description</th></tr>
<tr><td>Negated</td><td><code>boolVariable
-(/)-</code></td><td>The associated variable is forced to the negation of the flow on the left of the coil.</td></tr>
<tr><td>Negative Transition</td><td><code>boolVariable
-(N)-</code></td><td>The associated variable is forced to TRUE if the flow on the left changes from TRUE to FALSE (and forced to FALSE in all other cases).</td></tr>
<tr><td>Normal</td><td><code>boolVariable
-( )-</code></td><td>The associated variable is forced to the value of the flow on the left of the coil.</td></tr>
<tr><td>Positive Transition</td><td><code>boolVariable
-(P)-</code></td><td>The associated variable is forced to TRUE if the flow on the left changes from FALSE to TRUE (and forced to FALSE in all other cases).</td></tr>
<tr><td>Reset</td><td><code>boolVariable
-(R)-</code></td><td>The associated variable is forced to FALSE if the flow on the left is TRUE.
No action if the rung state is FALSE.
Rules for Reset coil animation:

Power Flow on left is TRUE:

The horizontal lines are red
The variable above (R) is black
The R and the circle around the R are black

Power Flow on left is FALSE and variable above reset coil is NOT Energized (OFF)

The horizontal lines are black
The variable above (R) is black
The R and the circle around the R are black

Power Flow on left is FALSE and variable above reset coil is Energized (ON)

The horizontal lines are black
The variable above (R) is red
The R and the circle around the R are red

</td></tr>
</table>

Variable	Coils	Description

Set	`boolVariable`
	`-(S)-`
		The associated variable is forced to TRUE if the flow on the left is TRUE.
		No action if the flow is FALSE.
		Rules for Set coil animation:
		• Power Flow on left is TRUE:
		  ◦ The horizontal wires on either side of the (S) are red
		  ◦ The variable and the (S) are red
		• Power Flow on left is FALSE and the (S) variable is Energized (ON)
		  ◦ The horizontal lines on either sided of (S) are black
		  ◦ The variable and the (S) are red
		• In all other cases:
		  ◦ The horizontal wires are black
		  ◦ The variable and the (S) are black

TIP

When a contact or coil is selected, you can press the **Spacebar** to change its type (normal, negated...).

When your application is running, you can select a contact and press the **Spacebar** to swap its value between TRUE and FALSE.

IMPORTANT

Although coils are commonly put at the end, the rung can be continued after a coil.
The flow is **never changed** by a coil symbol.

4 PLC Standard Libraries

The following topics detail the set of programming features and standard blocks:

Note: Some other functions not documented here are reserved for diagnostics and special operations. Please contact your technical support for further information.

4.1 Basic Operations

4.1.1 Data Manipulation

These are the language features for basic data manipulation:

- "Assignment :=" (➔ p. 54)
- "Bit Access" (➔ p. 56)
- Call a Function
- Call a Function Block
- "Call a Sub-Program" (➔ p. 56)
- "CountOf" (➔ p. 59)
- "DEC" (➔ p. 60)
- "INC" (➔ p. 64)
- "MoveBlock" (➔ p. 65)
- "NEG -" (➔ p. 92) integer negation (unary operator)
- "Parenthesis ( )" (➔ p. 68)

4.1.2 Control Program Execution

4.1.2.1 Language Features

These are the language features to control program execution:

- Jumps JMP JMPC JMPNC JMPCN
- LABELS
- "RETURN RET RETC RETNC RETCN" (➔ p. 70)

4.1.2.2 Structured Statements

These are the structured statements to control program execution:

Statement	Description
"CASE OF ELSE END_CASE" (➔ p. 57)	Switch to one of various possible statements.
"EXIT" (➔ p. 61)	Exit from a loop instruction.
"FOR TO BY END_FOR" (➔ p. 62)	Execute iterations of statements.
"IF THEN ELSE ELSIF END_IF" (➔ p. 63)	Conditional execution of statements.
"ON" (➔ p. 67)	Conditional execution of statements.
"REPEAT UNTIL END_REPEAT" (➔ p. 69)	Repeat a list of statements.
"WAIT / WAIT_TIME" (➔ p. 71)	Suspends the execution of an ST program.
"WHILE DO END_WHILE" (➔ p. 73)	Repeat a list of statements while a condition is TRUE.

4.1.3 Assignment :=

PLCopen ✓

Operator - Variable assignment.

4.1.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	ANY				Any variable or complex expression.

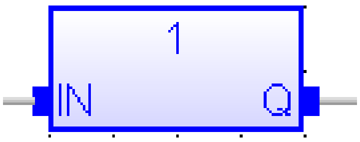
4.1.3.2 Outputs

Output	Data Type	Range	Unit	Description
Q	ANY			Forced variable.

4.1.3.3 Remarks

- The output variable and the input expression must have the same type.
- The forced variable cannot have the read-only attribute.
- In the FFLD and FBD languages, the 1 block is available to perform a 1 gain data copy (1 copy).
- In the IL language:
  - The FFLD instruction loads the first operand.
  - The ST instruction stores the current result into a variable.
    - The current result and the operand of ST must have the same type.
- Both FFLD and ST instructions can be modified by "N" in case of a Boolean operand for performing a Boolean negation.

4.1.3.4 FBD Language

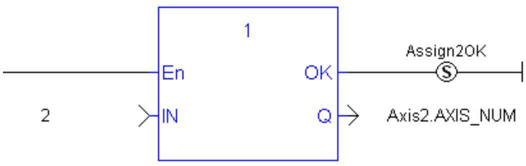


4.1.3.5 FFLD Language

- In the FFLD language, the input rung (EN) enables the assignment
- The output rung keeps the state of the input rung.

(* The copy is executed only if EN is TRUE. *)

(* ENO has the same value as EN. *)



4.1.3.6 IL Language

```
Op1: FFLD  IN    (* current result is: IN *)
        ST  Q     (* Q is: IN *)
        FFLDN IN1  (* current result is: NOT (IN1) *)
        ST  Q     (* Q is: NOT (IN1) *)
        FFLD  IN2  (* current result is: IN2 *)
        STN Q     (* Q is: NOT (IN2) *)
```

4.1.3.7 ST Language

```
Q := IN; (* copy IN into variable Q *)
          Q := (IN1 + (IN2 / IN 3)) * IN4; (* assign the result of a
complex expression *)
          result := SIN (angle); (* assign a variable with the result of
a function *)
          time := MyTon.ET; (* assign a variable with an output parameter
of a function block *)
```

See Also

Parenthesis ( )

4.1.4 Bit Access

You can directly specify a bit within an integer variable in expressions and diagrams using this notation:

```
Variable.BitNo
```

Where:

`Variable`: is the name of an integer variable.

`BitNo`: is the number of the bit in the integer.

The variable can have one of these data types:

- SINT, USINT, BYTE (8-bits from .0 to .7)
- INT, UINT, WORD (16-bits from .0 to .15)
- DINT, UDINT, DWORD (32-bits from .0 to 31)
- LINT, ULINT, LWORD (64-bits from .0 to .63)

0 (zero) always represents the less significant bit.

4.1.5 Differences Between Functions and Function Blocks

It is important to clearly understand what is different between functions and function blocks.

- A **Function** is called once and it performs an action.
  - This is synchronous.
- A **Function Block** (FB) is an instance that has its own set of data.
  - An FB maintains its own, internal machine state and often has an output to indicate when the work is done.
  - An FB is asynchronous.
  - The best way to work with a function block is to call it during multiple scan.
    - This triggers the action the first time, then monitor the status of this action, especially via the Done output.

See Also

- Call a Function
- Call a Function Block

4.1.6 Call a Sub-Program

A sub-program is called by another program.

- Unlike function blocks, local variables of a sub-program are not instantiated and you do not need to declare instances.
- A call to a sub-program processes the block algorithm using the specified input parameters.
- Output parameters can then be accessed.

4.1.6.1 FBD and FFLD Languages

To call a sub-program in FBD or FFLD languages, insert the block in the diagram and connect its inputs and outputs.

4.1.6.2 IL Language

To call a sub-program in the IL language, you must use the CAL instruction with the name of the sub-program, followed by the input parameters written between parentheses and separated by comas.

Alternatively, the CALC, CALCN or CALNC conditional instructions can be used:

```
CAL    Calls the sub-program
CALC   Calls the sub-program if the current result is TRUE
CALNC  Calls the sub-program if the current result is FALSE
CALCN  same as CALNC
```

Example

```
Op1: CAL  MySubProg (i1, i2)
FFLD  MySubProg.Q1
ST  Res1
FFLD  MySubProg.Q2
ST  Res2
```

4.1.6.3 ST Language

To call a sub-program in ST, you must specify its name, followed by the input parameters written between parentheses and separated by comas.

To have access to an output parameter, use the name of the sub-program followed by a dot . and the name of the parameter:

```
MySubProg (i1, i2); (* calls the sub-program *)
Res1 := MySubProg.Q1;
Res2 := MySubProg.Q2;
```

Alternatively, if a sub-program has one and only one output parameter, it can be called as a function in ST language:

```
Res := MySubProg (i1, i2);
```

4.1.7 CASE OF ELSE END_CASE

Statement - Switch to one of various possible statements.

4.1.7.1 Syntax

```
CASE <DINT expression> OF
<value> :
    <statements>
<value> , <value> :
    <statements>;
<value> .. <value> :
    <statements>;
ELSE
    <statements>
END_CASE;
```

4.1.7.2 Remarks

- All enumerated values correspond to the evaluation of the DINT expression and **are possible cases** in the execution of the statements.
- The statements specified after the ELSE keyword are executed if the expression takes a value which is not enumerated in the switch.
- For each case, you must specify either:
  - a value.
  - a list of possible values separated by comas (,).
  - a range of values specified by a "min .. max" interval.

- You must enter space characters before and after the ".." separator.

4.1.7.3 FBD Language

Not available.

4.1.7.4 FFLD Language

Not available.

4.1.7.5 IL Language

Not available.

4.1.7.6 ST Language

```
(* This example check first prime numbers: *)
CASE iNumber OF
0 :
   Alarm := TRUE;
   AlarmText := '0 gives no result';
1 .. 3, 5 :
   bPrime := TRUE;
4, 6 :
   bPrime := FALSE;
ELSE
   Alarm := TRUE;
   AlarmText := 'I don't know after 6 !';
END_CASE;
```

See Also

- EXIT
- "FOR TO BY END_FOR" (→ p. 62)

- "IF THEN ELSE ELSIF END_IF" (➜ p. 63)
- "REPEAT UNTIL END_REPEAT" (➜ p. 69)
- "WHILE DO END_WHILE" (➜ p. 73)

4.1.8 CountOf

PLCopen ✓

Function - Returns the number of items in an array.

4.1.8.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ARR	Any				Declared array.

4.1.8.2 Outputs

Output	Data Type	Range	Unit	Description
Q	DINT			Total number of items in the array.

4.1.8.3 Remarks

- The input must be an array and can have any data type.
- This function is particularly useful to avoid writing directly the actual size of an array in a program.
  - This keeps the program independent from the declaration.

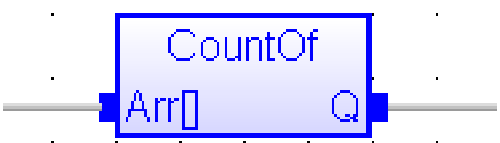
Example

```
FOR i := 1 TO CountOf (MyArray) DO
   MyArray[i-1] := 0;
END_FOR;
```

Examples

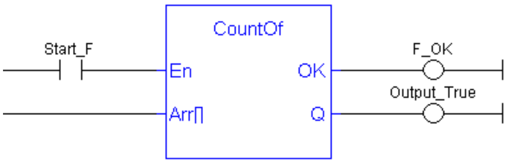
Array	Return
`Arr1 [ 0..9 ]`	10
`Arr2 [ 0..4 , 0..9 ]`	50

4.1.8.4 FBD Language



4.1.8.5 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.
  - The function is executed only if EN is TRUE.



4.1.8.6 IL Language

Not available.

4.1.8.7 ST Language

```
Q := CountOf (ARR);
```

4.1.9 DEC

Function - Decrease a numerical variable.

4.1.9.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	ANY				Numerical variable (increased after call).

4.1.9.2 Outputs

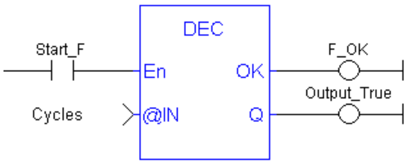
Output	Data Type	Range	Unit	Description
Q	ANY			Decreased value.

4.1.9.3 Remarks

- This function is particularly designed for "ST Language" (→ p. 61).
  - It allows simplified writing as assigning the result of the function is not mandatory.
- When the function is called, the variable connected to the `IN` input is decreased and copied to Q.
  - All data types are supported except BOOL and STRING: for these types, the output is the copy of `IN`.
- For real values, a variable is decreased by 1.0.
- For time values, a variable is decreased by 1 ms.
- The `IN` input must be directly connected to a variable.
  - It cannot be a constant or complex expression.

4.1.9.4 FBD Language

4.1.9.5 FFLD Language



4.1.9.6 IL Language

Not available.

4.1.9.7 ST Language

```
IN := 2;
Q := DEC (IN);
(* now: IN = 1 ; Q = 1 *)
DEC (IN); (* simplified call *)
```

4.1.10 EXIT

Statement - Exit from a loop instruction.

4.1.10.1 Remarks

- The EXIT statement indicates that the current loop (FOR, REPEAT, or WHILE) must be finished.
- The execution continues after the END_FOR, END_REPEAT, or END_WHILE keyword or the loop where the EXIT is.
- EXIT quits only one loop and cannot be used to exit at the same time several levels of nested loops.

IMPORTANT

Loop instructions can lead to infinite loops that block the target cycle.

4.1.10.2 FBD Language

Not available.

4.1.10.3 FFLD Language

Not available.

4.1.10.4 IL Language

Not available.

4.1.10.5 ST Language

```
(* This program searches for the first non null item of an array: *)
iFound = -1; (* means: not found *)
FOR iPos := 0 TO (iArrayDim - 1) DO
   IF iPos <> 0 THEN
      iFound := iPos;
      EXIT;
   END_IF;
END_FOR;
```

See Also

- "CASE OF ELSE END_CASE" (➜ p. 57)
- "FOR TO BY END_FOR" (➜ p. 62)
- "IF THEN ELSE ELSIF END_IF" (➜ p. 63)
- "REPEAT UNTIL END_REPEAT" (➜ p. 69)
- "WHILE DO END_WHILE" (➜ p. 73)

4.1.11 FOR TO BY END_FOR

Statement - Execute iterations of statements.

4.1.11.1 Syntax

```
    FOR <index> := <minimum> TO <maximum>
BY <step> DO
        <statements>
    END_FOR;

index = DINT internal variable used as index.
minimum = DINT expression: initial value for index.
maximum = DINT expression: maximum allowed value for index.
step = DINT expression: increasing step of index after each iteration
(default is 1) .
```

4.1.11.2 Remarks

- The `BY <step>` statement can be omitted.
- The default value for the step is 1.

4.1.11.3 FBD Language

Not available.

4.1.11.4 FFLD Language

Not available.

4.1.11.5 IL Language

Not available.

4.1.11.6 ST Language

```
iArrayDim := 10;

(* resets all items of the array to 0 *)
FOR iPos := 0 TO (iArrayDim - 1) DO
   MyArray[iPos] := 0;
END_FOR;

(* set all items with odd index to 1 *)
FOR iPos := 1 TO 9 BY 2 DO
   MyArray[ipos] := 1;
END_FOR;
```

See Also

- "CASE OF ELSE END_CASE" (→ p. 57)
- "EXIT" (→ p. 61)
- "IF THEN ELSE ELSIF END_IF" (→ p. 63)
- "REPEAT UNTIL END_REPEAT" (→ p. 69)
- "REPEAT UNTIL END_REPEAT" (→ p. 69)
- "WHILE DO END_WHILE" (→ p. 73)

4.1.12 IF THEN ELSE ELSIF END_IF

Statement - Conditional execution of statements.

4.1.12.1 Syntax

```
IF <BOOL expression> THEN
     <statements>
ELSIF <BOOL expression> THEN
     <statements>
ELSE
     <statements>
END_IF;
```

4.1.12.2 Remarks

- The IF statement is available in ST only.
- The execution of the statements is conditioned by a Boolean expression.
- ELSIF and ELSE statements are optional.
- There can be several ELSIF statements.

4.1.12.3 FBD Language

Not available.

4.1.12.4 FFLD Language

Not available.

4.1.12.5 IL Language

Not available.

4.1.12.6 ST Language

```
(* simple condition *)

IF bCond THEN
   Q1 := IN1;
   Q2 := TRUE;
END_IF;

(* binary selection *)
IF bCond THEN
   Q1 := IN1;
   Q2 := TRUE;
   ELSE
      Q1 := IN2;
      Q2 := FALSE;
```

```
END_IF;

(* enumerated conditions *)
IF bCond1 THEN
   Q1 := IN1;
   ELSIF bCond2 THEN
      Q1 := IN2;
      ELSIF bCond3 THEN
         Q1 := IN3;
         ELSE
            Q1 := IN4;
END_IF;
```

See Also

- "CASE OF ELSE END_CASE" (➔ p. 57)
- "EXIT" (➔ p. 61)
- "FOR TO BY END_FOR" (➔ p. 62)
- "REPEAT UNTIL END_REPEAT" (➔ p. 69)
- "WHILE DO END_WHILE" (➔ p. 73)

4.1.13 INC

PLCopen ✓

Function - Increase a numerical variable.

4.1.13.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	ANY				Numerical variable (increased after call).

4.1.13.2 Outputs

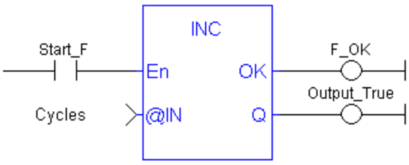
Output	Data Type	Range	Unit	Description
Q	ANY			Increased value.

4.1.13.3 Remarks

- This function is particularly designed for "ST Language" (➔ p. 65).
  - It allows simplified writing as assigning the result of the function is not mandatory.
- When the function is called, the variable connected to the `IN` input is decreased and copied to Q.
  - All data types are supported except BOOL and STRING: for these types, the output is the copy of `IN`.
- For real values, a variable is decreased by 1.0.
- For time values, a variable is decreased by 1 ms.
- The `IN` input must be directly connected to a variable.
  - It cannot be a constant or complex expression.

4.1.13.4 FBD Language

4.1.13.5 FFLD Language



4.1.13.6 IL Language

Not available.

4.1.13.7 ST Language

```
IN := 1;
Q := INC (IN);
(* now: IN = 2 ; Q = 2 *)
```

```
INC (IN); (* simplified call *)
```

4.1.14 MoveBlock

PLCopen ✓

Function - Move/Copy items of an array.

4.1.14.1 Inputs

Input	Data Type	Range	Unit	Default	Description
DST	ANY (*)				Array containing the destination of the copy.
NB	DINT				Number of items to be copied.
PosDST	DINT				Index of the destination in DST.
PosSRC	DINT				Index of the first character in SRC.
SRC	ANY (*)				Array containing the source of the copy.

(*) SRC and DST cannot be a STRING.

4.1.14.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL			TRUE if successful.

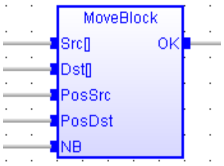
4.1.14.3 Remarks

- Arrays of string are not supported by this function.
- The function copies a number (NB) of consecutive items starting at the PosSRC index in SRC array to PosDST position in DST array.

- SRC and DST can be the same array.
- In this case, the function avoids lost items when source and destination areas overlap.

- This function verifies array bounds and is always safe.
  - The function returns TRUE if successful.
  - It returns FALSE if input positions and number do not fit the bounds of SRC and DST arrays.

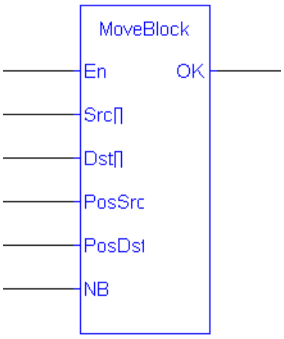
4.1.14.4 FBD Language



4.1.14.5 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.

(* The function is executed only if EN is TRUE. *)



4.1.14.6 IL Language

Not available.

4.1.14.7 ST Language

```
OK := MOVEBLOCK (SRC, DST, PosSRS, PosDST, NB);
```

4.1.15 NEG -

PLCopen

Operator - Performs a negation of the input. (unary operator)

4.1.15.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	ANY				Numeric value.

4.1.15.2 Outputs

Output	Data Type	Range	Unit	Description
Q	ANY			Negation of the input.

4.1.15.3 Truth Table

IN	Q
0	0
1	-1
-123	123

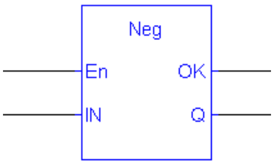
4.1.15.4 Remarks

- In FBD and FFLD language, the block **NEG** can be used.

4.1.15.5 FBD Language

4.1.15.6 FFLD Language

- In the FFLD language, the conversion is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.
- The negation is executed only if EN is TRUE.
- ENO keeps the same value as EN.



4.1.15.7 IL Language

Not available.

4.1.15.8 ST Language

- In the ST language, "-" can be followed by a complex Boolean expression between parentheses.
  - The output data type must be the same as the input data type.

```
Q := -IN;
Q := - (IN1 + IN2);
```

4.1.16 ON

Statement - Conditional execution of statements.

4.1.16.1 Syntax

```
ON <BOOL expression> DO
   <statements>
END_DO;
```

4.1.16.2 Remarks

- The `ON` instruction provides a simpler syntax for checking the rising edge of a Boolean condition.
- Statements within the `ON` structure are executed only when the Boolean expression rises from FALSE to TRUE.
- The `ON` instruction avoids systematic use of the R_TRIG function block or other "last state" flags.
- The `ON` syntax is available in any program or sub-program.
- This statement is an extension to the standard and is NOT IEC 61131-3 compliant.

⚠CAUTION

This instruction is NOT UDFB safe.
Do not use inside UDFBs.

4.1.16.3 FBD Language

Not available.

4.1.16.4 FFLD Language

Not available.

4.1.16.5 IL Language

Not available.

4.1.16.6 ST Language

```
(* This example counts the rising edges of variable bIN *)
ON bIN DO
   diCount := diCount + 1;
END_DO;
```

4.1.17 Parenthesis ( )

Operator - Force the evaluation order in a complex expression.

4.1.17.1 Remarks

- Parentheses are used in ST and IL languages for changing the default evaluation order of various operations within a complex expression.
- Example: The default evaluation of `2 * 3 + 4` expression in ST language gives a result of 10 because `*` operator has the highest priority.
  - Changing the expression as `2 * ( 3 + 4)` gives a result of 14.
- Parentheses can be nested in a complex expression.

These are the default evaluation priority order for ST language operations:

Priority	Operation	Description
1	- NOT	Unary operators
2	* /	Multiply / Divide
3	+ -	Add / Subtract

Priority	Operation	Description
4	< > <= >= = <>	Comparisons
5	& AND	Boolean And
6	OR	Boolean Or
7	XOR	Exclusive OR

- In the IL language:
  - The default order is the sequence of instructions.
    - Each new instruction modifies the current result sequentially.
  - The opening parenthesis "(" is written between the instruction and its operand.
    - The closing parenthesis ")" must be written alone as an instruction without operand.

4.1.17.2 FBD Language

Not available.

4.1.17.3 FFLD Language

Not available.

4.1.17.4 IL Language

```
Op1: FFLD(  IN1
     ADD(  IN2
     MUL   IN3
     )
     SUB   IN4
     )
     ST    Q     (* Q is: (IN1 + (IN2 * IN3) -  IN4) *)
```

4.1.17.5 ST Language

```
Q := (IN1 + (IN2 / IN 3)) * IN4;
```

See Also

Assignment :=

4.1.18 REPEAT UNTIL END_REPEAT

Statement - Repeat a list of statements.

4.1.18.1 Syntax

```
    REPEAT
        <statements>
    UNTIL <BOOL expression>
    END_REPEAT;
```

4.1.18.2 Remarks

- The statements between `REPEAT` and `UNTIL` are executed until the Boolean expression is TRUE.
- The condition is evaluated **after** the statements are executed.
  - Statements are executed at least once.

IMPORTANT

Loop instructions can lead to infinite loops that block the target cycle.
Never test the state of an input in the condition because the input is not refreshed before the next cycle.

4.1.18.3 FBD Language

Not available.

4.1.18.4 FFLD Language

Not available.

4.1.18.5 IL Language

Not available.

4.1.18.6 ST Language

```
iPos := 0;
REPEAT
   MyArray[iPos] := 0;
   iNbCleared := iNbCleared + 1;
   iPos := iPos + 1;
UNTIL iPos = iMax
END_REPEAT;
```

See Also

- "CASE OF ELSE END_CASE" (→ p. 57)
- "EXIT" (→ p. 61)
- "FOR TO BY END_FOR" (→ p. 62)
- "IF THEN ELSE ELSIF END_IF" (→ p. 63)
- "WHILE DO END_WHILE" (→ p. 73)

4.1.19 RETURN RET RETC RETNC RETCN

Statement - Jump to the end of the program.

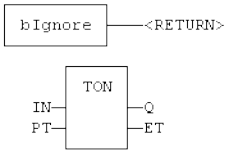
4.1.19.1 Remarks

- When used within an action block of a SFC step, the RETURN statement jumps to the end of the action block.

4.1.19.2 FBD Language

- In the FBD language, the return statement is represented by the <RETURN> symbol.
  - The input of the symbol must be connected to a valid Boolean signal.
  - The jump is performed only if the input is TRUE.

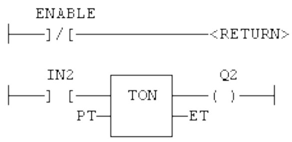
(* In this example, the TON block will not be called if bIgnore is TRUE. *)



4.1.19.3 FFLD Language

- In the FFLD language, the <RETURN> symbol is used as a coil at the end of a rung.
  - The jump is performed only if the rung state is TRUE.

(* In this example all the networks above 5 are skipped if ENABLE is FALSE *)



4.1.19.4 IL Language

These are the meanings of possible instructions:

- `RET` Jump to the end always.
- `RETC` Jump to the end if the current result is TRUE.
- `RETNC` Jump to the end if the current result is FALSE.
- `RETCN` Same as `RETNC`.

```
Start: FFLD   IN1
       RETC          (* Jump to the end if IN1 is TRUE *)

       FFLD   IN2    (* these instructions are not executed *)
       ST     Q2      (* if IN1 is TRUE *)
       RET           (* Jump to the end unconditionally *)

       FFLD   IN3    (* these instructions are never executed *)
       ST     Q3
```

4.1.19.5 ST Language

```
IF NOT bEnable THEN
   RETURN;
END_IF;
(* the rest of the program is not executed if bEnable is FALSE *)
```

See Also

- Jumps JMP JMPC JMPNC JMPCN
- LABELS

4.1.20 WAIT / WAIT_TIME

Statement - Suspends the execution of an ST program.

4.1.20.1 Syntax

```
WAIT<BOOL expression> ;
```

```
WAIT_TIME<TIME expression> ;
```

4.1.20.2 Remarks

- The `WAIT` statement:
  - Provides an easy way to program a state machine.
    - This avoids the use of complex `CASE` structures.
  - Verifies the attached Boolean expression and takes these actions:
    - If the expression is TRUE, the program continues normally.
    - If the expression is FALSE, then the execution of the program is suspended up to the **next PLC cycle**. The Boolean expression will be checked again during next cycles until it becomes TRUE. The execution of other programs is not affected.
- The `WAIT_TIME` statement suspends the execution of the program for the specified duration.
  - The execution of other programs is not affected.

These instructions are available in ST language only and have no correspondence in other languages.

- The WAIT and WAIT_TIME instructions:
  - Cannot be called in a User-Defined Function Block (UDFB).
    - The use of WAIT or WAIT_TIME in a UDFB provokes a compile error.
  - Can be called in a sub-program.
    - However, it can lead to some unsafe situation if the same sub program is called from various programs.
  - Do not support re-entrancy.
    - Avoiding this situation is the responsibility of the programmer.
    - The compiler outputs some warning messages if a sub-program containing a WAIT or WAIT_TIME instruction is called from more than one program.
  - Must not be called from ST parts of SFC programs.
    - This makes no sense as SFC is already a state machine.
    - The use of WAIT or WAIT_TME in SFC or in a sub-program called from SFC provokes a compile error.
  - Are not available when the code is compiled through a "C" compiler.
    - Using "C" code generation with a program containing a WAIT or WAIT_TIME instruction provokes an error during post-compiling.
- This statement is an extension to the standard and is NOT IEC 61131-3 compliant.

⚠CAUTION

This instruction is NOT UDFB safe.
Do not use inside UDFBs.

4.1.20.3 FBD Language

Not available.

4.1.20.4 FFLD Language

Not available.

4.1.20.5 IL Language

Not available.

4.1.20.6 ST Language

```
(* use of WAIT with different kinds of BOOL expressions *)
WAIT BoolVariable;
WAIT (diLevel > 100) AND NOT bAlarm;
WAIT SubProgCall ();

(* use of WAIT_TIME with different kinds of TIME expressions *)
WAIT_TIME t#2s;
WAIT_TIME TimeVariable;
```

4.1.21 WHILE DO END_WHILE

Statement - Repeat a list of statements while a condition is TRUE.

4.1.21.1 Syntax

```
    WHILE <BOOL expression> DO
        <statements>
    END_WHILE;
```

4.1.21.2 Remarks

- The statements between DO and END_WHILE are executed while the Boolean expression is TRUE.
- The condition is evaluated **before** the statements are executed.
- If the condition is FALSE when WHILE is first reached, statements are never executed.

IMPORTANT

Loop instructions can lead to infinite loops that block the target cycle.
Never test the state of an input in the condition because the input is not refreshed before the next cycle.

4.1.21.3 FBD Language

Not available.

4.1.21.4 FFLD Language

Not available.

4.1.21.5 IL Language

Not available.

4.1.21.6 ST Language

```
iMax := 10;
WHILE iPos < iMax DO
   MyArray[iPos]:=0;
```

```
    iPos +:=1;
END_WHILE;
```

See Also

- "CASE OF ELSE END_CASE" (➜ p. 57)
- "EXIT" (➜ p. 61)
- "FOR TO BY END_FOR" (➜ p. 62)
- "IF THEN ELSE ELSIF END_IF" (➜ p. 63)
- "REPEAT UNTIL END_REPEAT" (➜ p. 69)

4.2 Boolean Operations

PLCopen ✓

4.2.1 Standard Operators

These are the standard operators for managing Booleans.

Operator	Description
AND ANDN &	Performs a logical AND of all inputs.
NOT	Performs a Boolean negation of the input.
OR / ORN	Performs a logical OR of all inputs.
"QOR" (➜ p. 78)	Counts the number of TRUE inputs.
R	Force a Boolean output to FALSE.
S	Force a Boolean output to TRUE.
XOR / XORN	Performs an exclusive OR of all inputs.

4.2.2 Available Blocks

These are the available blocks for managing Boolean signals:

Block	Description
f_trig	Falling pulse detection.
FlipFlop	Flipflop bistable.
r_trig	Rising pulse detection.
RS	Reset dominant bistable.
sema	Semaphore.
SR	Set dominant bistable.

4.2.3 FlipFlop

PLCopen ✓

Function Block - Flipflop bistable.

4.2.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	BOOL				Swap command (on rising edge).
RST	BOOL				Reset to FALSE.

4.2.3.2 Outputs

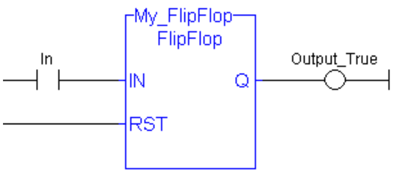
Output	Data Type	Range	Unit	Description
Q	BOOL			Output.

4.2.3.3 Remarks

- The output is systematically reset to FALSE if RST is TRUE.
- The output changes on each rising edge of the IN input, if RST is FALSE.

4.2.3.4 FBD Language

4.2.3.5 FFLD Language



4.2.3.6 IL Language

```
(* MyFlipFlop is declared as an instance of FLIPFLOP function block: *)
Op1: CAL
MyFlipFlop (IN, RST)
     FFLD
MyFlipFlop.Q
     ST  Q1
```

4.2.3.7 ST Language

```
(* MyFlipFlop is declared as an instance of FLIPFLOP function block: *)
MyFlipFlop (IN, RST);
Q := MyFlipFlop.Q;
```

See Also

R

S

SR

4.2.4 f_trig

PLCopen

Function Block - Falling pulse detection.

4.2.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
CLK	BOOL				Boolean signal.

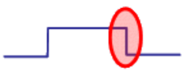
4.2.4.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL			TRUE when the input changes from TRUE to FALSE.

4.2.4.3 Truth Table

CLK	CCLK (prev)	Q
0	0	0
0	1	1
1	0	0
1	1	0

4.2.4.4 Remarks

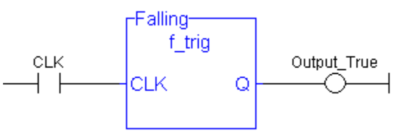


- It is recommended to use declared instances of R_TRIG or F_TRIG function blocks.
  - This is to avoid contingencies during an Online Change.

4.2.4.5 FBD Language

4.2.4.6 FFLD Language

- In the FFLD language,]P[and]N[contacts can be used.



4.2.4.7 IL Language

```
(* MyTrigger is declared as an instance of F_TRIG function block *)
Op1: CAL MyTrigger (CLK)
```

```
MyTrigger.Q
Q
```

4.2.4.8 ST Language

```
(* MyTrigger is declared as an instance of F_TRIG function block. *)
MyTrigger (CLK);
Q := MyTrigger.Q;
```

See Also

r_trig

4.2.5 NOT

Operator - Performs a Boolean negation of the input.

4.2.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	BOOL				Boolean value.

4.2.5.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL			Boolean negation of the input.

4.2.5.3 Truth Table

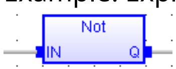
IN	Q
0	1
1	0

4.2.5.4 Remarks

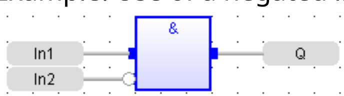
4.2.5.5 FBD Language

- In the FBD language, the block "NOT" can be used.
  - Alternatively, you can use a link terminated by a "o" negation.

Example: Explicit use of the NOT block:



Example: Use of a negated link: Q is IN1 AND NOT IN2:



4.2.5.6 FFLD Language

- In the FFLD language, negated contacts and coils can be used.

Example: Negated contact: Q is: IN1 AND NOT IN2:

```
     IN1      IN2       Q
|---] [------]\[------( )---|
```

Example: Negated coil: Q is NOT (IN1 AND IN2):

```
     IN1      IN2       Q
|---] [------] [------(\)---|
```

4.2.5.7 IL Language

- In the IL language, the **N** modifier can be used with instructions FFLD, AND, OR, XOR and ST.
  - It represents a negation of the operand.

```
Op1: FFLDN IN1
     OR   IN2
     ST   Q    (* Q is equal to: (NOT IN1) OR IN2 *)
Op2: FFLD  IN1
     AND IN2
     STN Q    (* Q is equal to: NOT (IN1 AND IN2) *)st
```

4.2.5.8 ST Language

- In the ST language, NOT can be followed by a complex Boolean expression between parentheses.

```
Q := NOT IN;
Q := NOT (IN1 OR IN2);
```

See Also

- AND ANDN &
- OR / ORN
- XOR / XORN

4.2.6 QOR

Operator - Counts the number of TRUE inputs.

4.2.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN1 INn	BOOL				Boolean inputs.

4.2.6.2 Outputs

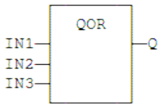
Output	Data Type	Range	Unit	Description
Q	DINT			Number of inputs being TRUE.

4.2.6.3 Remarks

- The block accepts a non-fixed number of inputs.

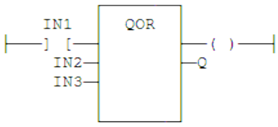
4.2.6.4 FBD Language

- The block can have a maximum of 16 inputs.



4.2.6.5 FFLD Language

- The block can have a maximum of 16 inputs.



4.2.6.6 IL Language

```
Op1: LD   IN1
     QOR   IN2, IN3
     ST   Q
```

4.2.6.7 ST Language

```
Q := QOR (IN1, IN2);
Q := QOR (IN1, IN2, IN3, IN4, IN5, IN6);
```

4.2.7 R

Operator - Force a Boolean output to FALSE.

4.2.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description
RESET	BOOL				Condition.

4.2.7.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL			Output to be forced.

4.2.7.3 Truth Table

RESET	Q (prev)	Q
0	0	0
0	1	1
1	0	0

RESET	Q (prev)	Q
1	1	0

4.2.7.4 Remarks

None

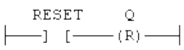
4.2.7.5 FBD Language

- In the FBD language, RS and SR function blocks are preferred.
  - Use "RS" (➔ p. 80) or "SR" (➔ p. 86) function blocks.
  - (S) and (R) coils can be used.

4.2.7.6 FFLD Language

- In the FFLD languages, they are represented by (S) and (R) coils.

Example: Use of R coil:



4.2.7.7 IL Language

- In the IL language, S and R operators are available as standard instructions.

```
Op1: FFLD  RESET
     R    Q     (* Q is forced to FALSE if RESET is TRUE *)
                (* Q is unchanged if RESET is FALSE *)
```

4.2.7.8 ST Language

Not available.

See Also

- "RS" (➔ p. 80)
- "S" (➔ p. 83)
- "SR" (➔ p. 86)

4.2.8 RS

Function Block - Reset dominant bistable.

4.2.8.1 Inputs

Input	Data Type	Range	Unit	Default	Description
SET	BOOL				Condition for forcing to TRUE.
RESET1	BOOL				Condition for forcing to FALSE. Highest priority command.

4.2.8.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL			Output to be forced.

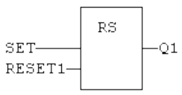
4.2.8.3 Truth Table

SET	RESET1	Q1 prev	Q1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

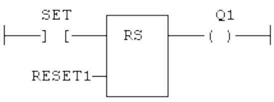
4.2.8.4 Remarks

- The output is unchanged when both inputs are FALSE.
- When both inputs are TRUE, the output is forced to FALSE. (reset dominant)

4.2.8.5 FBD Language



4.2.8.6 FFLD Language



4.2.8.7 IL Language

```
(* MyRS is declared as an instance of RS function block *)
Op1: CAL MyRS (SET, RESET1)
     FFLD  MyRS.Q1
     ST   Q1
```

4.2.8.8 ST Language

```
(* MyRS is declared as an instance of RS function block *)
MyRS (SET, RESET1);
Q1 := MyRS.Q1;
```

See Also

- "R" (→ p. 79)
- "RS" (→ p. 80)
- "SR" (→ p. 86)

4.2.9 r_trig

PLCopen

Function Block - Rising pulse detection.

4.2.9.1 Inputs

Input	Data Type	Range	Unit	Default	Description
CLK	BOOL				Boolean signal.

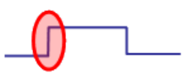
4.2.9.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL			TRUE when the input changes from FALSE to TRUE.

4.2.9.3 Truth Table

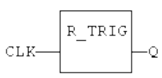
CLK	CCLK (prev)	Q
0	0	0
0	1	0
1	0	1
1	1	0

4.2.9.4 Remarks



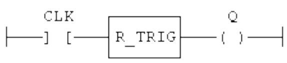
- It is recommended to use declared instances of R_TRIG or F_TRIG function blocks.
  - This is to avoid contingencies during an Online Change.

4.2.9.5 FBD Language



4.2.9.6 FFLD Language

- In the FFLD language,]P[and]N[contacts can be used.
- The input signal is the rung.
- The rung is the output.



4.2.9.7 IL Language

```
(* MyTrigger is declared as an instance of R_TRIG function block *)
Op1: CAL MyTrigger (CLK)
FFLD  MyTrigger.Q
ST  Q
```

4.2.9.8 ST Language

```
(* MyTrigger is declared as an instance of R_TRIG function block *)
MyTrigger (CLK);
Q := MyTrigger.Q;
```

See Also

- "f_trig" (→ p. 76)

4.2.10 S

Operator - Force a Boolean output to TRUE.

4.2.10.1 Inputs

Input	Data Type	Range	Unit	Default	Description
SET	BOOL				Condition.

4.2.10.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL			Output to be forced.

4.2.10.3 Truth Table

SET	Q prev	Q
0	0	0
0	1	1
1	0	1
1	1	1

4.2.10.4 Remarks

None

4.2.10.5 FBD Language

- In the FBD language, RS and SR function blocks are preferred.
  - Use "RS" (→ p. 80) or "SR" (→ p. 86) function blocks.
  - (S) and (R) coils can be used.

4.2.10.6 FFLD Language

- In the FFLD languages, they are represented by (S) and (R) coils.

Example: Use of S coil:

```
     SET      Q
|---] [-----(S)---|
```

4.2.10.7 IL Language

- In the IL language, S and R operators are available as standard instructions.

```
Op1: FFLD  SET
     S    Q     (* Q is forced to TRUE if SET is TRUE *)
                (* Q is unchanged if SET is FALSE *)
```

4.2.10.8 ST Language

Not available.

See Also

- "R" (→ p. 79)
- "RS" (→ p. 80)
- "SR" (→ p. 86)

4.2.11 sema

PLCopen

Function Block - Semaphore.

4.2.11.1 Inputs

Input	Data Type	Range	Unit	Default	Description
CLAIM	BOOL				Takes the semaphore.
RELEASE	BOOL				Releases the semaphore.

4.2.11.2 Outputs

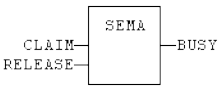
Output	Data Type	Range	Unit	Description
BUSY	BOOL			TRUE if semaphore is busy.

4.2.11.3 Remarks

The function block implements this algorithm:

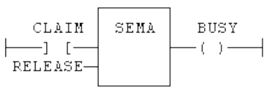
```
BUSY := mem;
if CLAIM then
   mem := TRUE;
else if RELEASE then
   BUSY := FALSE;
   mem := FALSE;
end_if;
```

4.2.11.4 FBD Language



4.2.11.5 FFLD Language

- In the FFLD language, the input rung is the CLAIM command.
  - The output rung is the BUSY output signal.



4.2.11.6 IL Language

```
(* MySema is a declared instance of SEMA function block *)
Op1: CAL MySema (CLAIM, RELEASE)
     FFLD  MyBlinker.BUSY
     ST   BUSY
```

4.2.11.7 ST Language

```
(* MySema is a declared instance of SEMA function block *)
MySema (CLAIM, RELEASE);
BUSY := MyBlinker.BUSY;
```

4.2.12 SR

PLCopen

Function Block - Set dominant bistable.

4.2.12.1 Inputs

Input	Data Type	Range	Unit	Default	Description
SET1	BOOL				Condition for forcing to TRUE. Highest priority command.
RESET	BOOL				Condition for forcing to FALSE.

4.2.12.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL			Output to be forced.

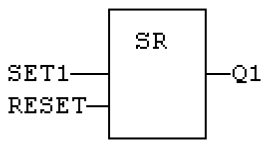
4.2.12.3 Truth Table

SET1	RESET	Q1 prev	Q1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

4.2.12.4 Remarks

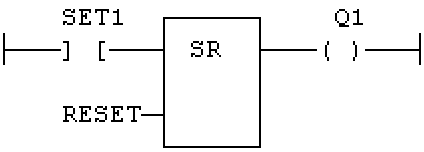
- The output is unchanged when both inputs are FALSE.
- When both inputs are TRUE, the output is forced to FALSE. (set dominant)

4.2.12.5 FBD Language



4.2.12.6 FFLD Language

- The SET1 command is the rung.
  - The rung is the output.



4.2.12.7 IL Language

```
(* MySR is declared as an instance of SR function block *)
Op1: CAL MySR (SET1, RESET)
     FFLD  MySR.Q1
     ST  Q1
```

4.2.12.8 ST Language

```
(* MySR is declared as an instance of SR function block *)
MySR (SET1, RESET);
Q1 := MySR.Q1;
```

See Also

- "R" (→ p. 79)
- "RS" (→ p. 80)
- "S" (→ p. 83)

4.2.13 XOR / XORN

Operator - Performs an exclusive OR of all inputs.

4.2.13.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN1	BOOL				First Boolean input.
IN2	BOOL				Second Boolean input.

4.2.13.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL			Exclusive OR of all inputs.

4.2.13.3 Truth Table

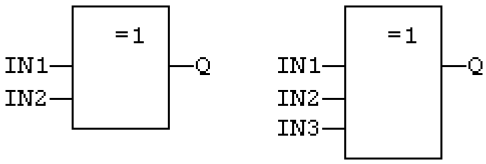
IN1	IN2	Q
0	0	0

IN1	IN2	Q
0	1	1
1	0	1
1	1	0

4.2.13.4 Remarks

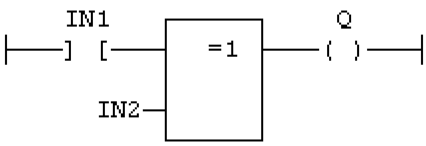
- The block is called **=1** in FBD and FFLD languages.

4.2.13.5 FBD Language



4.2.13.6 FFLD Language

- The first input is the rung.
- The rung is the output.



4.2.13.7 IL Language

- In the IL language, the XOR instruction performs an exclusive OR between the current result and the operand.
  - The current result must be Boolean.
  - The XORN instruction performs an exclusive between the current result and the Boolean negation of the operand.

```
Op1: FFLD   IN1
     XOR  IN2
     ST   Q   (* Q is equal to: IN1 XOR IN2 *)
Op2: FFLD   IN1
     XORN IN2
     ST   Q   (* Q is equal to: IN1 XOR (NOT IN2) *)
```

4.2.13.8 ST Language

```
Q := IN1 XOR IN2;
Q := IN1 XOR IN2 XOR IN3;
```

See Also

- AND ANDN &
- "NOT" (→ p. 77)
- OR / ORN

4.3 Arithmetic Operations

PLCopen ✓

4.3.1 Standard Operators

These are the standard operators that perform arithmetic operations:

Operator	Description
"Addition +" (➔ p. 89)	Performs an addition of all inputs.
"Divide /" (➔ p. 90)	Performs a division of all inputs.
"Multiply *" (➔ p. 99)	Performs a multiplication of all inputs.
NEG -	Performs a negation of the input. (unary operator)
"Subtraction -" (➔ p. 102)	Performs a subtraction of all inputs.

4.3.2 Standard Functions

These are the standard functions that perform arithmetic operations:

Function	Description
limit	Limits a numeric value between low and high bounds.
max	Get the maximum of two integers.
min	Get the minimum of two integers.
mod / modLR / modR	Calculation of modulo.
odd	Test if an integer is odd.
"SetWithin" (➔ p. 101)	Force a value when inside an interval.

4.3.3 Addition +

Operator - Performs an addition of all inputs.

4.3.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN1	ANY				First input.
IN2	ANY				Second input.

4.3.3.2 Outputs

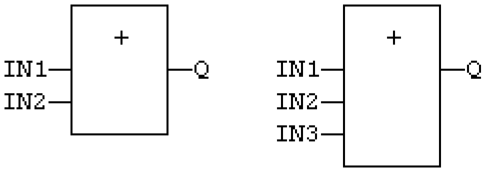
Output	Data Type	Range	Unit	Description
Q	ANY			Result: IN1 + IN2.

4.3.3.3 Remarks

- All inputs and the output must have the same type.
- The addition can be used with strings.
  - The result is the concatenation of the input strings.

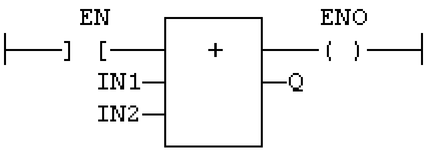
4.3.3.4 FBD Language

- In the FBD language, the block can have a maximum of 32 inputs.



4.3.3.5 FFLD Language

- In the FFLD language, the input rung (EN) enables the operation.
  - The output rung (ENO) keeps the same value as the input rung.



4.3.3.6 IL Language

- In the IL language, the **ADD** instruction performs an addition between the current result and the operand.
  - The current result and the operand must have the same type.

```
Op1: FFLD  IN1
     ADD IN2
     ST  Q    (* Q is equal to: IN1 + IN2 *)
Op2: FFLD  IN1
     ADD IN2
     ADD IN3
     ST  Q    (* Q is equal to: IN1 + IN2 + IN3 *)
```

4.3.3.7 ST Language

```
Q := IN1 + IN2;
MyString := 'He' + 'll ' + 'o';    (* MyString is equal to 'Hello' *)
```

See Also

- Divide /
- Multiply *
- Subtraction -

4.3.4 Divide /

Operator - Performs a division of all inputs.

4.3.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN1	ANY_NUM				First input.
IN2	ANY_NUM				Second input.

4.3.4.2 Outputs

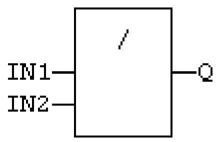
Output	Data Type	Range	Unit	Description
Q	ANY_NUM			Result: IN1 / IN2.

4.3.4.3 Remarks

- All inputs and the output must have the same type.

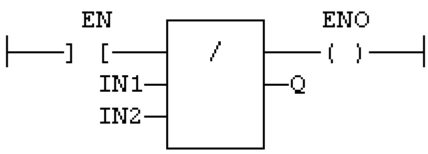
4.3.4.4 FBD Language

- In the FBD language, the block can have a maximum of 32 inputs.



4.3.4.5 FFLD Language

- In the FFLD language, the input rung (EN) enables the operation.
  - The output rung (ENO) keeps the same value as the input rung.



4.3.4.6 IL Language

- In the IL language, the **DIV** instruction performs a division between the current result and the operand.
  - The current result and the operand must have the same type.

```
Op1: FFLD  IN1
     DIV IN2
     ST  Q     (* Q is equal to: IN1 / IN2 *)
Op2: FFLD  IN1
     DIV IN2
     DIV IN3
     ST  Q     (* Q is equal to: IN1 / IN2 / IN3 *)
```

4.3.4.7 ST Language

```
Q := IN1 / IN2;
```

See Also

- Addition +
- Multiply *
- Subtraction -

4.3.5 NEG -

PLCopen

Operator - Performs a negation of the input. (unary operator)

4.3.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	ANY				Numeric value.

4.3.5.2 Outputs

Output	Data Type	Range	Unit	Description
Q	ANY			Negation of the input.

4.3.5.3 Truth Table

IN	Q
0	0
1	-1
-123	123

4.3.5.4 Remarks

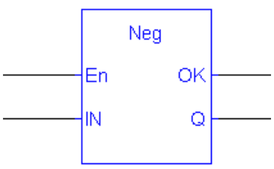
- In FBD and FFLD language, the block **NEG** can be used.

4.3.5.5 FBD Language

4.3.5.6 FFLD Language

- In the FFLD language, the conversion is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.

- The negation is executed only if EN is TRUE.
- ENO keeps the same value as EN.



4.3.5.7 IL Language

Not available.

4.3.5.8 ST Language

- In the ST language, "-" can be followed by a complex Boolean expression between parentheses.
  - The output data type must be the same as the input data type.

```
Q := -IN;
Q := - (IN1 + IN2);
```

4.3.6 limit

PLCopen ✓

Function - Limits a numeric value between low and high bounds.

4.3.6.1 Inputs

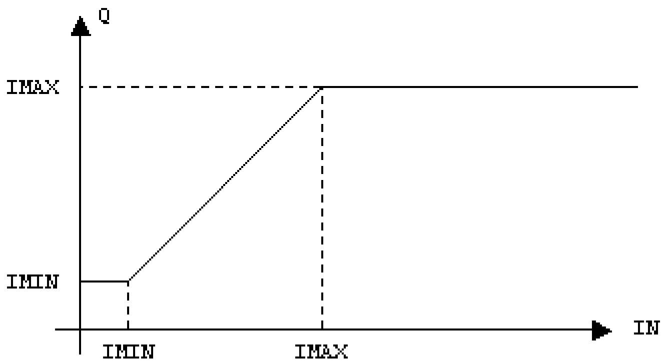
Input	Data Type	Range	Unit	Default	Description
IMIN	DINT				Low bound.
IN	DINT				Input value.
IMAX	DINT				High bound.

4.3.6.2 Outputs

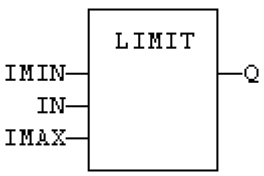
Output	Data Type	Range	Unit	Description
Q	DINT			IMIN if IN < IMIN; IMAX if IN > IMAX; IN otherwise.

4.3.6.3 Remarks

4.3.6.3.1 Function Diagram



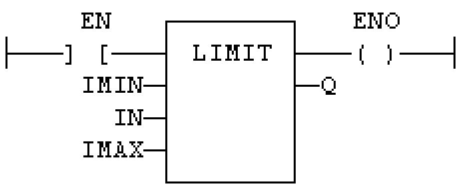
4.3.6.4 FBD Language



4.3.6.5 FFLD Language

- In the FFLD language, the input rung (EN) enables the operation.
  - The output rung keeps the state of the input rung.

- The comparison is executed only if EN is TRUE.
- ENO has the same value as EN.



4.3.6.6 IL Language

- In the IL language, the first input must be loaded before the function call.
  - Other inputs are operands of the function, separated by a coma.

```
Op1: LD    IMIN
     LIMIT IN, IMAX
     ST    Q
```

4.3.6.7 ST Language

```
Q := LIMIT (IMIN, IN, IMAX);
```

See Also

- max
- min
- mod / modLR / modR
- odd

4.3.7 max

PLCopen

Function - Get the maximum of two integers.

4.3.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN1	ANY				First input.
IN2	ANY				Second input.

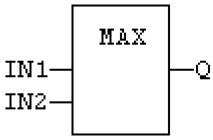
4.3.7.2 Outputs

Output	Data Type	Range	Unit	Description
Q	ANY			IN1 if IN1 > IN2; IN2 otherwise.

4.3.7.3 Remarks

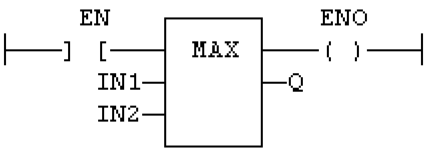
None

4.3.7.4 FBD Language



4.3.7.5 FFLD Language

- In the FFLD language, the input rung (EN) enables the operation.
  - The output rung keeps the state of the input rung.
  - The comparison is executed only if EN is TRUE.
  - ENO has the same value as EN.



4.3.7.6 IL Language

- In the IL language, the first input must be loaded before the function call.
  - The second input is the operand of the function.

```
Op1: LD   IN1
     MAX IN2
     ST   Q     (* Q is the maximum of IN1 and IN2 *)
```

4.3.7.7 ST Language

```
Q := MAX (IN1, IN2);
```

See Also

- "limit" (→ p. 93)
- "min" (→ p. 95)
- "mod / modLR / modR" (→ p. 96)
- "odd" (→ p. 100)

4.3.8 min

PLCopen ✓

Function - Get the minimum of two integers.

4.3.8.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN1	ANY				First input.
IN2	ANY				Second input.

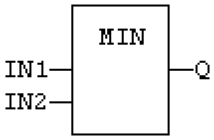
4.3.8.2 Outputs

Output	Data Type	Range	Unit	Description
Q	ANY			IN1 if IN1 < IN2; IN2 otherwise.

4.3.8.3 Remarks

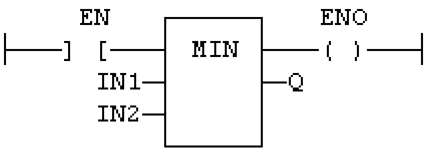
None

4.3.8.4 FBD Language



4.3.8.5 FFLD Language

- In the FFLD language, the input rung (EN) enables the operation.
  - The output rung keeps the state of the input rung.
  - The comparison is executed only if EN is TRUE.
  - ENO has the same value as EN.



4.3.8.6 IL Language

- In the IL language, the first input must be loaded before the function call.
  - The second input is the operand of the function.

```
Op1: LD   IN1
     MIN  IN2
     ST   Q     (* Q is the minimum of IN1 and IN2 *)
```

4.3.8.7 ST Language

```
Q := MIN (IN1, IN2);
```

See Also

- "limit" (→ p. 93)
- "max" (→ p. 94)
- "mod / modLR / modR" (→ p. 96)
- "odd" (→ p. 100)

4.3.9 mod / modLR / modR

PLCopen ✓

Function - Calculation of modulo.

The modulo is calculated as:

```
Q = IN - Trunc(IN/BASE) * BASE
```

where *Trunc(x)* calculates the truncated (rounded toward zero) value of x.

NOTE

If BASE = 0, then Q will return 0.
If IN = 0, then Q will return 0.

4.3.9.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	mod = DINT modR = REAL modLR = LREAL				Input value.
BASE	mod = DINT modR = REAL modLR = LREAL				Base of the modulo.

4.3.9.2 Outputs

Output	Data Type	Range	Unit	Description
Q	mod = DINT modR = REAL modLR = LREAL			Modulo: rest of the integer division (IN / BASE).

4.3.9.3 Examples

MOD Examples

IN	BASE	Q
11	7	4
7	7	0
5	7	5
0	7	0
-5	7	-5
-7	7	0
-11	7	-4
11	-7	4
7	-7	0
5	-7	5
0	-7	0
-5	-7	-5

IN	BASE	Q
-7	-7	0
-11	-7	-4
5	0	0

MODR / MODLR Examples

IN	BASE	Q
9.00	5.75	3.25
5.75	5.75	0
2.50	5.75	2.50
0	5.75	0
-2.50	5.75	-2.5
-5.75	5.75	0
-9.00	5.75	-3.25
9.00	-5.75	3.25
5.75	-5.75	0
2.50	-5.75	2.50
0	-5.75	0
-2.50	-5.75	-2.5
-5.75	-5.75	0
-9.00	-5.75	-3.25
4.25	0	0

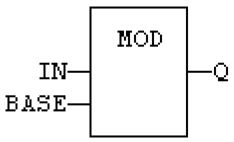
4.3.9.4 Remarks

None

The MOD family of functions can return negative numbers. By adding BASE to the result, the modulo value may be forced to a positive number if the range is required to be [0, BASE). An example of how to accomplish this in ST code follows.

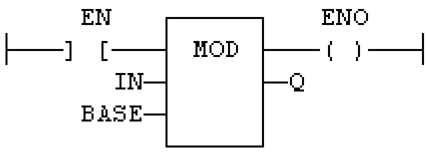
```
q := MOD(x, base);
IF q < 0 THEN
   q := q + base;
END_IF;
```

4.3.9.5 FBD Language



4.3.9.6 FFLD Language

- In the FFLD language, the input rung (EN) enables the operation.
  - The output rung keeps the state of the input rung.
  - The comparison is executed only if EN is TRUE.
  - ENO has the same value as EN.



4.3.9.7 IL Language

- In the IL language, the first input must be loaded before the function call.
  - The second input is the operand of the function.

```
Op1: LD  IN
     MOD BASE
     ST  Q     (* Q is the rest of integer division: IN / BASE *)
```

4.3.9.8 ST Language

```
Q := MOD (IN, BASE);
```

See Also

- "limit" (➔ p. 93)
- "max" (➔ p. 94)
- "min" (➔ p. 95)
- "odd" (➔ p. 100)

4.3.10 Multiply *

Operator - Performs a multiplication of all inputs.

4.3.10.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN1	ANY_NUM				First input.
IN2	ANY_NUM				Second input.

4.3.10.2 Outputs

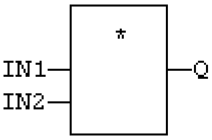
Output	Data Type	Range	Unit	Description
Q	ANY_NUM			Result: IN1 * IN2.

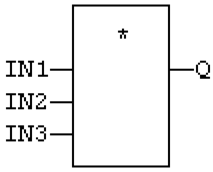
4.3.10.3 Remarks

- All inputs and the output must have the same type.

4.3.10.4 FBD Language

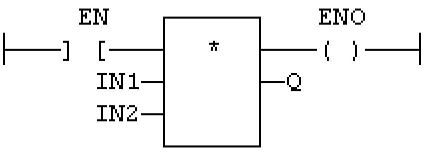
- In the FBD language, the block can have a maximum of 32 inputs.





4.3.10.5 FFLD Language

- The multiplication is executed only if EN is TRUE.
- In the FFLD language, the input rung (EN) enables the operation.
  - The output rung (ENO) keeps the same value as the input rung.
- ENO is equal to EN.



4.3.10.6 IL Language

- In the IL language, the **MUL** instruction performs a multiplication between the current result and the operand.
  - The current result and the operand must have the same type.

```
Op1: FFLD  IN1
     MUL IN2
     ST  Q     (* Q is equal to: IN1 * IN2 *)
Op2: FFLD  IN1
     MUL IN2
     MUL IN3
     ST  Q     (* Q is equal to: IN1 * IN2 * IN3 *)
```

4.3.10.7 ST Language

```
Q := IN1 * IN2;
```

See Also

- Addition +
- Divide /
- Subtraction -

4.3.11 odd

PLCopen

Function - Test if an integer is odd.

4.3.11.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	DINT				Input value.

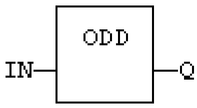
4.3.11.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL			• TRUE if IN is odd. • FALSE if IN is even.

4.3.11.3 Remarks

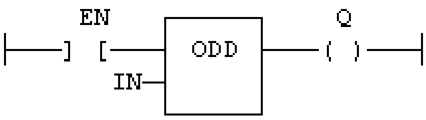
None

4.3.11.4 FBD Language



4.3.11.5 FFLD Language

- In the FFLD language, the input rung (EN) enables the operation.
  - The output rung keeps the state of the input rung.
  - The function is executed only if EN is TRUE.



4.3.11.6 IL Language

- In the IL language, the first input must be loaded before the function call.
  - The second input is the operand of the function.

```
Op1: LD   IN
     ODD
     ST   Q     (* Q is TRUE if IN is odd. *)
```

4.3.11.7 ST Language

```
Q := ODD (IN);
```

See Also

- "limit" (→ p. 93)
- "max" (→ p. 94)
- min
- "mod / modLR / modR" (→ p. 96)

4.3.12 SetWithin

- PLCopen ✓

Function - Force a value when inside an interval.

4.3.12.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	ANY				Input.
MIN	ANY				Low limit of the interval.
MAX	ANY				High limit of the interval.
VAL	ANY				Value to apply when inside the interval.

4.3.12.2 Outputs

Output	Data Type	Range	Unit	Description
Q	ANY			Result.

4.3.12.3 Truth Table

In	Q
IN < MIN	IN
IN > MAX	IN
MIN < IN < MAX	VAL

4.3.12.4 Remarks

- The output is forced to VAL when the IN value is within the [MIN ... MAX] interval.
- It is set to IN when outside the interval.

4.3.13 Subtraction -

Operator - Performs a subtraction of all inputs.

4.3.13.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN1	ANY_NUM / TIME				First input.
IN2	ANY_NUM / TIME				Second input.

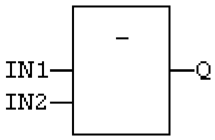
4.3.13.2 Outputs

Output	Data Type	Range	Unit	Description
Q	ANY_NUM / TIME			Result: IN1 - IN2.

4.3.13.3 Remarks

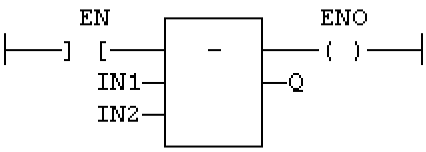
- All inputs and the output must have the same type.

4.3.13.4 FBD Language



4.3.13.5 FFLD Language

- In the FFLD language, the input rung (EN) enables the operation.
  - The output rung (ENO) keeps the same value as the input rung.
  - The subtraction is executed only if EN is TRUE.
  - ENO is equal to EN.



4.3.13.6 IL Language

- In the IL language, the **SUB** instruction performs a subtraction between the current result and the operand.
  - The current result and the operand must have the same type.

```
Op1: FFLD  IN1
     SUB IN2
     ST  Q     (* Q is equal to: IN1 - IN2 *)
Op2: FFLD  IN1
     SUB IN2
     SUB IN3
     ST  Q     (* Q is equal to: IN1 - IN2 - IN3 *)
```

4.3.13.7 ST Language

```
Q := IN1 - IN2;
```

See Also

- Addition +
- Divide /
- Multiply *

4.4 Comparison Operations

These are the standard operators and blocks that perform comparisons:

Operator	Description
LT <	Less than
LE <=	Less than or equal to

Operator	Description
NE <>	Not equal to
EQ =	Is equal to
GT >	Greater than
GE >=	Greater than or equal to
4.4.1 CMP	Detailed comparison

4.4.1 CMP

PLCopen ✓

Function Block - Comparison with detailed outputs for integer inputs.

4.4.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN1	DINT				First value.
IN2	DINT				Second value.

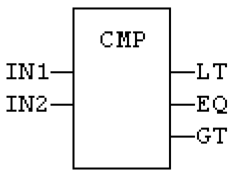
4.4.1.2 Outputs

Output	Data Type	Range	Unit	Description
EQ	BOOL			TRUE if IN1 = IN2.
GT	BOOL			TRUE if IN1 > IN2.
LT	BOOL			TRUE if IN1 < IN2.

4.4.1.3 Remarks

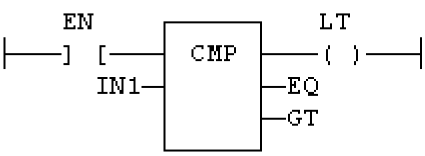
None

4.4.1.4 FBD Language



4.4.1.5 FFLD Language

- In the FFLD language, the rung input (EN) validates the operation.
  - The rung output is the result of **LT** (lower than) comparison).
  - The comparison is executed only if EN is TRUE.



4.4.1.6 IL Language

```
(* MyCmp is declared as an instance of CMP function block *)
Op1: CAL MyCmp (IN1, IN2)
     LD  MyCmp.LT
     ST  bLT
     LD  MyCmp.EQ
     ST  bEQ
     LD  MyCmp.GT
     ST  bGT
```

4.4.1.7 ST Language

```
(* MyCmp is declared as an instance of CMP function block. *)
MyCMP (IN1, IN2);
bLT := MyCmp.LT;
bEQ := MyCmp.EQ;
bGT := MyCmp.GT;
```

See Also

- EQ =
- GE >=
- GT >
- LE <=
- LT <
- NE <>

4.4.2 GE >=

PLCopen ✓

Operator - Tests if first input is greater than or equal to second input.

4.4.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN1	ANY_NUM				First input.
IN2	ANY_NUM				Second input.

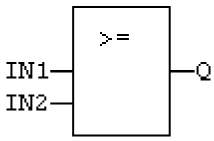
4.4.2.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL			TRUE if IN1 >= IN2.

4.4.2.3 Remarks

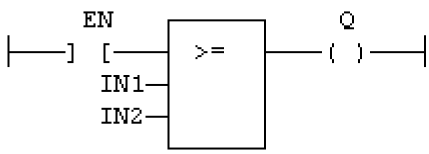
- Both inputs must have the same type.
- Comparisons can be used with strings.
  - In this case, the lexical order is used for comparing the input strings.
  - Example: ABC is less than ZX; ABCD is greater than ABC.

4.4.2.4 FBD Language



4.4.2.5 FFLD Language

- In the FFLD language, the input rung (EN) enables the operation.
  - The output rung is the result of the comparison.
  - The comparison is executed only if EN is TRUE.



4.4.2.6 IL Language

- In the IL language, the GE instruction performs the comparison between the current result and the operand.
  - The current result and the operand must have the same type.

```
Op1: FFLD  IN1
     GE   IN2
     ST   Q     (* Q is true if IN1 >= IN2 *)
```

4.4.2.7 ST Language

```
Q := IN1 >= IN2;
```

See Also

- CMP
- EQ =
- GT >
- LE <=
- LT <
- NE <>

4.4.3 GT >

Operator - Test if first input is greater than second input.

4.4.3.1 Inputs

`IN1 : ANY` First input.
`IN2 : ANY` Second input.

4.4.3.2 Outputs

`Q : BOOL` TRUE if IN1 > IN2

4.4.3.3 Remarks

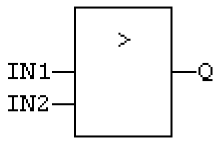
Both inputs must have the same type. In FFLD language, the input rung (EN) enables the operation, and the output rung is the result of the comparison. In IL language, the GT instruction performs the comparison between the current result and the operand. The current result and the operand must have the same type.

Comparisons can be used with strings. In that case, the lexical order is used for comparing the input strings. For instance, "ABC" is less than "ZX" ; "ABCD" is greater than "ABC".

4.4.3.4 ST Language

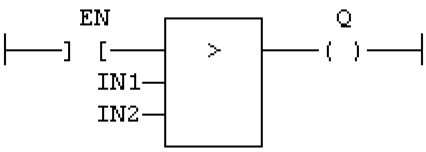
```
Q := IN1 > IN2;
```

4.4.3.5 FBD Language



4.4.3.6 FFLD Language

(* The comparison is executed only if EN is TRUE. *)



4.4.3.7 IL Language

```
Op1: FFLD  IN1
     GT   IN2
     ST   Q     (* Q is true if IN1 > IN2 *)
```

See Also

- CMP
- EQ =
- GE >=
- LE <=
- LT <
- NE <>

4.4.4 EQ =

PLCopen

Operator - Test if first input is equal to second input.

4.4.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN1	ANY				First input.
IN2	ANY				Second input.

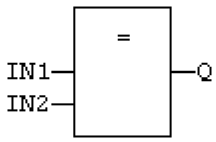
4.4.4.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL			TRUE if IN1 = IN2.

4.4.4.3 Remarks

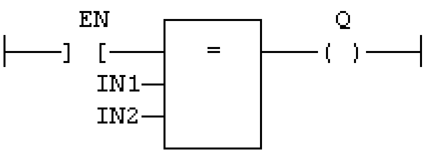
- Both inputs must have the same type.
- Comparisons can be used with strings.
  - In that case, the lexical order is used for comparing the input strings.
  - Example: ABC is less than ZX; ABCD is greater than ABC.
- Equality comparisons cannot be used with TIME variables.
  - The reason is that the timer actually has the resolution of the target cycle and test can be unsafe as some values can never be reached.

4.4.4.4 FBD Language



4.4.4.5 FFLD Language

- In the FFLD language, the input rung (EN) enables the operation.
  - The output rung is the result of the comparison.
  - The comparison is executed only if EN is TRUE.



4.4.4.6 IL Language

- In the IL language, the EQ instruction performs the comparison between the current result and the operand.
  - The current result and the operand must have the same type.

```
Op1: FFLD  IN1
     EQ   IN2
     ST   Q     (* Q is true if IN1 = IN2 *)
```

4.4.4.7 ST Language

```
Q := IN1 = IN2;
```

See Also

- CMP
- GE >=
- GT >
- LE <=
- LT <
- NE <>

4.4.5 NE <>

PLCopen ✓

Operator - Test if first input is not equal to second input.

4.4.5.1 Inputs

`IN1 : ANY`	First input.
`IN2 : ANY`	Second input.

4.4.5.2 Outputs

`Q : BOOL`	TRUE if IN1 is not equal to IN2.

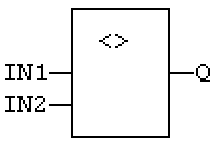
4.4.5.3 Remarks

Both inputs must have the same data type. In FFLD language, the input rung (EN) enables the operation, and the output rung is the result of the comparison. In IL language, the NE instruction performs the comparison between the current result and the operand. The current result and the operand must have the same data type.

Comparisons can be used with strings. In that case, the lexical order is used for comparing the input strings. For instance, "ABC" is less than "ZX" ; "ABCD" is greater than "ABC".

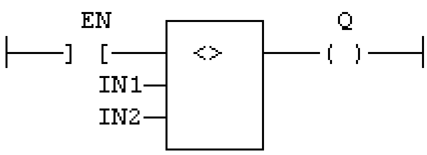
Equality comparisons cannot be used with TIME variables.The reason is that the timer has the resolution of the target cycle and test can be unsafe as some values can never be reached.

4.4.5.4 FBD Language



4.4.5.5 FFLD Language

(* The comparison is executed only if EN is TRUE. *)



4.4.5.6 IL Language

```
Op1: FFLD  IN1
     NE  IN2
     ST  Q    (* Q is true if IN1 is not equal to IN2 *)
```

4.4.5.7 ST Language

```
Q := IN1 <> IN2;
```

See Also

- CMP
- EQ =
- GE >=
- GT >
- LE <=
- LT <

4.4.6 LE <=

PLCopen ✓

Operator - Test if first input is less than or equal to second input.

4.4.6.1 Inputs

`IN1 : ANY`	First input.
`IN2 : ANY`	Second input.

4.4.6.2 Outputs

`Q : BOOL`	TRUE if IN1 <= IN2.

4.4.6.3 Remarks

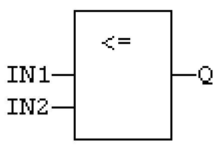
Both inputs must have the same type. In FFLD language, the input rung (EN) enables the operation, and the output rung is the result of the comparison. In IL language, the LE instruction performs the comparison between the current result and the operand. The current result and the operand must have the same type.

Comparisons can be used with strings. In that case, the lexical order is used for comparing the input strings. For instance, "ABC" is less than "ZX" ; "ABCD" is greater than "ABC".

4.4.6.4 ST Language

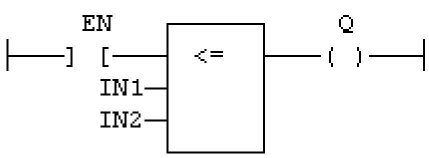
Q := IN1 <= IN2;

4.4.6.5 FBD Language



4.4.6.6 FFLD Language

(* The comparison is executed only if EN is TRUE. *)



4.4.6.7 IL Language

```
Op1: FFLD  IN1
     LE  IN2
     ST  Q    (* Q is true if IN1 <= IN2 *)
```

See Also

- CMP
- EQ =
- GE >=
- GT >
- LT <
- NE <>

4.4.7 LT <

PLCopen

Operator - Test if first input is less than second input.

4.4.7.1 Inputs

`IN1 : ANY`	First input.
`IN2 : ANY`	Second input.

4.4.7.2 Outputs

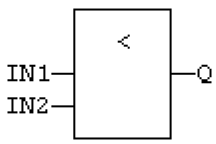
`Q : BOOL`	TRUE if IN1 < IN2.

4.4.7.3 Remarks

Both inputs must have the same data type. In FFLD language, the input rung (EN) enables the operation, and the output rung is the result of the comparison. In IL language, the LT instruction performs the comparison between the current result and the operand. The current result and the operand must have the same datatype.

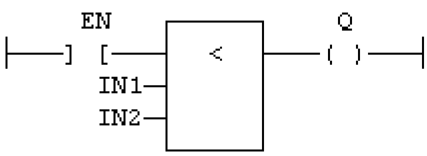
Comparisons can be used with strings. In that case, the lexical order is used for comparing the input strings. For instance, "ABC" is less than "ZX" ; "ABCD" is greater than "ABC".

4.4.7.4 FBD Language



4.4.7.5 FFLD Language

(* The comparison is executed only if EN is TRUE. *)



4.4.7.6 IL Language

```
Op1: FFLD  IN1
     LT   IN2
     ST   Q     (* Q is true if IN1 < IN2 *)
```

4.4.7.7 ST Language

```
Q := IN1 < IN2;
```

See Also

- CMP
- EQ =
- GE >=
- GT >
- LE <=
- NE <>

4.5 Type Conversion Functions

These are the standard functions for converting a data into another data type:

Function	Description
"any_to_bool" (➜ p. 112)	Converts to Boolean.
"any_to_dint / any_to_udint" (➜ p. 114)	Converts to integer (32-bit - default).
"any_to_int / any_to_uint" (➜ p. 115)	Converts to 16-bit integer.
"any_to_lint / any_to_ulint" (➜ p. 116)	Converts to long (64-bit) integer.
"any_to_lreal" (➜ p. 117)	Converts to double precision real.
"any_to_real" (➜ p. 118)	Converts to real.
"any_to_sint / any_to_usint" (➜ p. 121)	Converts to small (8-bit) integer.
"any_to_string" (➜ p. 122)	Converts to character string.
"any_to_time" (➜ p. 120)	Converts to time.
"NUM_TO_STRING" (➜ p. 123)	Converts a number to a string.

These are the standard functions performing conversions in BCD format (*):

Function	Description
"bcd_to_bin" (➜ p. 124)	Converts a BCD value to a binary value.
"bin_to_bcd" (➜ p. 125)	Converts a binary value to a BCD value.

(*) BCD conversion functions may not be supported by all targets.

4.5.1 any_to_bool

Operator - Converts the input into Boolean value.

4.5.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	ANY				Input value.

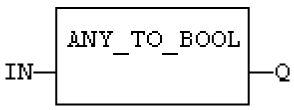
4.5.1.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL			Value converted to Boolean.

4.5.1.3 Remarks

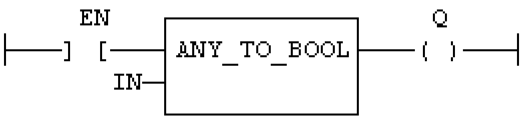
- For DINT, REAL, and TIME input data types, the result is FALSE if the input is 0 (zero).
  - The result is TRUE in all other cases.
- For STRING inputs, the output is TRUE if the input string is not empty.
  - The output is FALSE if the string is empty.

4.5.1.4 FBD Language



4.5.1.5 FFLD Language

- In the FFLD language, the conversion is executed only if the input rung (EN) is TRUE.
  - The output rung is the result of the conversion.
  - The output rung is FALSE if the EN is FALSE.



4.5.1.6 IL Language

- In the IL Language, the any_to_bool function converts the current result.

```
Op1: FFLD  IN
     ANY_TO_BOOL
     ST   Q
```

4.5.1.7 ST Language

```
Q := ANY_TO_BOOL (IN);
```

See Also

- any_to_dint / any_to_udint
- any_to_int / any_to_uint
- any_to_lint / any_to_ulint
- any_to_lreal
- any_to_real
- any_to_sint / any_to_usint

- any_to_string
- any_to_time

4.5.2 any_to_dint / any_to_udint

Operator - Converts the input into integer value.

Can be unsigned with `any_to_udint`.

4.5.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	ANY				Input value.

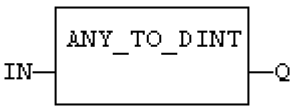
4.5.2.2 Outputs

Output	Data Type	Range	Unit	Description
Q	DINT			Value converted to a signed double integer. (32-bit).
Q	UDINT			Value converted to an unsigned double integer. (32-bit).

4.5.2.3 Remarks

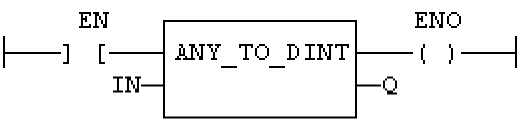
- For BOOL input data types, the output is 0 (zero) or 1.
- For REAL input data type, the output is the integer part of the input real.
- For TIME input data types, the result is the number of milliseconds.
- For STRING input data types, the output is the number represented by the string or 0 (zero) if the string does not represent a valid number.

4.5.2.4 FBD Language



4.5.2.5 FFLD Language

- In the FFLD language, the conversion is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.



4.5.2.6 IL Language

- In the IL Language, the `any_to_udint` converts the current result.

```
Op1: FFLD  IN
     ANY_TO_DINT
     ST   Q
```

4.5.2.7 ST Language

```
Q := ANY_TO_DINT (IN);
```

See Also

- any_to_bool
- any_to_int / any_to_uint
- any_to_lint / any_to_ulint
- any_to_lreal
- any_to_real
- any_to_sint / any_to_usint
- any_to_string
- any_to_time

4.5.3 any_to_int / any_to_uint

Operator - Converts the input into 16-bit integer value.

Can be unsigned with `any_to_uint`.

4.5.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	ANY				Input value.

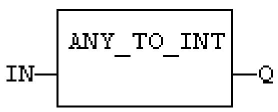
4.5.3.2 Outputs

Output	Data Type	Range	Unit	Description
Q	INT			Value converted to a signed integer. (16-bit).
Q	UINT			Value converted to an unsigned integer. (16-bit).

4.5.3.3 Remarks

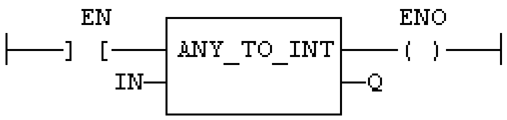
- For BOOL input data types, the output is 0 (zero) or 1.
- For REAL input data type, the output is the integer part of the input real.
- For TIME input data types, the result is the number of milliseconds.
- For STRING input data types, the output is the number represented by the string or 0 (zero) if the string does not represent a valid number.

4.5.3.4 FBD Language



4.5.3.5 FFLD Language

- In the FFLD language, the conversion is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.



4.5.3.6 IL Language

- In the IL Language, the `any_to_int` converts the current result.

```
Op1: FFLD  IN
     ANY_TO_INT
     ST  Q
```

4.5.3.7 ST Language

```
Q := ANY_TO_INT (IN);
```

See Also

- any_to_bool
- any_to_dint / any_to_udint
- any_to_lint / any_to_ulint
- any_to_lreal
- any_to_real
- any_to_sint / any_to_usint
- any_to_string
- any_to_time

4.5.4 any_to_lint / any_to_ulint

PLCopen

Operator - Converts the input into long (64-bit) integer value.

Can be unsigned with `any_to_ulint`.

4.5.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	ANY				Input value.

4.5.4.2 Outputs

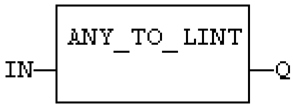
Output	Data Type	Range	Unit	Description
Q	LINT			Value converted to long (64-bit) integer.
Q	ULINT			Value converted to long (64-bit) unsigned integer.

4.5.4.3 Remarks

- For BOOL input data types, the output is 0 (zero) or 1.
- For REAL input data type, the output is the integer part of the input real.
- For TIME input data types, the result is the number of milliseconds.

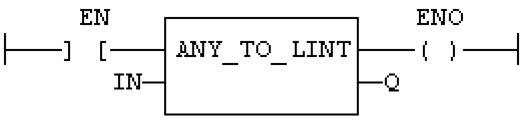
- For STRING input data types, the output is the number represented by the string or 0 (zero) if the string does not represent a valid number.

4.5.4.4 FBD Language



4.5.4.5 FFLD Language

- In the FFLD language, the conversion is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.



4.5.4.6 IL Language

- In IL Language, the `any_to_lint` converts the current result.

```
Op1: FFLD  IN
     ANY_TO_LINT
     ST  Q
```

4.5.4.7 ST Language

```
Q := ANY_TO_LINT (IN);
```

See Also

- any_to_bool
- any_to_dint / any_to_udint
- any_to_int / any_to_uint
- any_to_lreal
- any_to_real
- any_to_sint / any_to_usint
- any_to_string
- any_to_time

4.5.5 any_to_lreal

Operator - Converts the input into double precision floating point real value.

4.5.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	ANY				Input value.

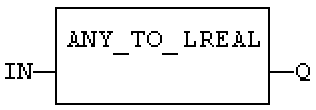
4.5.5.2 Outputs

Output	Data Type	Range	Unit	Description
Q	LREAL			Value converted to double precision floating point real.

4.5.5.3 Remarks

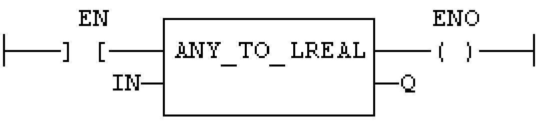
- For BOOL input data types, the output is 0.0 or 1.0.
- For DINT input data types, the output is the same number.
- For TIME input data types, the result is the number of milliseconds.
- For STRING input data types, the output is the number represented by the string or 0.0 if the string does not represent a valid number.

4.5.5.4 FBD Language



4.5.5.5 FFLD Language

- In the FFLD language, the conversion is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.



4.5.5.6 IL Language

- In IL Language, the `any_to-lreal` converts the current result.

```
Op1: FFLD  IN
     ANY_TO_LREAL
     ST   Q
```

4.5.5.7 ST Language

```
Q := ANY_TO_LREAL (IN);
```

See Also

- any_to_bool
- any_to_dint / any_to_udint
- any_to_int / any_to_uint
- any_to_lint / any_to_ulint
- any_to_real
- any_to_sint / any_to_usint
- any_to_string
- any_to_time

4.5.6 any_to_real

PLCopen

Operator - Converts the input into single precision floating point real value.

4.5.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	ANY				Input value.

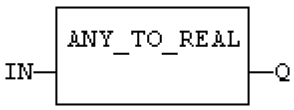
4.5.6.2 Outputs

Output	Data Type	Range	Unit	Description
Q	REAL			Value converted to single precision floating point real.

4.5.6.3 Remarks

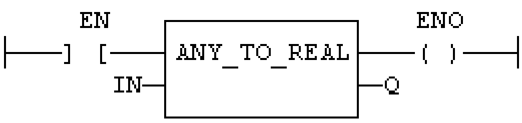
- For BOOL input data types, the output is 0.0 or 1.0.
- For DINT input data types, the output is the same number.
- For TIME input data types, the result is the number of milliseconds.
- For STRING input data types, the output is the number represented by the string or 0.0 if the string does not represent a valid number.

4.5.6.4 FBD Language



4.5.6.5 FFLD Language

- In the FFLD language, the conversion is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.



4.5.6.6 IL Language

- In the IL Language, the `any_to_real` converts the current result.

```
Op1: FFLD   IN
     ANY_TO_REAL
     ST   Q
```

4.5.6.7 ST Language

```
Q := ANY_TO_REAL (IN);
```

See Also

- any_to_bool
- any_to_dint / any_to_udint
- any_to_int / any_to_uint

- any_to_lint / any_to_ulint
- any_to_lreal
- any_to_sint / any_to_usint
- any_to_string
- any_to_time

4.5.7 any_to_time

PLCopen

Operator - Converts the input into time value.

4.5.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	ANY				Input value.

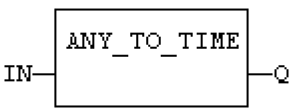
4.5.7.2 Outputs

Output	Data Type	Range	Unit	Description
Q	TIME			Value converted to time.

4.5.7.3 Remarks

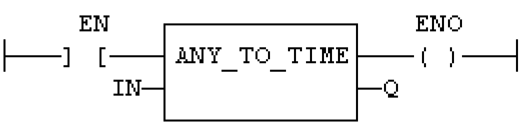
- For BOOL input data types, the output is t#0ms or t#1ms.
- For DINT or REAL input data type, the output is the time represented by the input number as a number of milliseconds.
- For STRING input data types, the output is the time represented by the string or t#0ms if the string does not represent a valid time.

4.5.7.4 FBD Language



4.5.7.5 FFLD Language

- In the FFLD language, the conversion is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.



4.5.7.6 IL Language

- In the IL Language, the `any_to_time` converts the current result.

```
Op1: FFLD  IN
ANY_TO_TIME
ST  Q
```

4.5.7.7 ST Language

```
Q := ANY_TO_TIME (IN);
```

See Also

- any_to_bool
- any_to_dint / any_to_udint
- any_to_int / any_to_uint
- any_to_lint / any_to_ulint
- any_to_lreal
- any_to_real
- any_to_sint / any_to_usint
- any_to_string

4.5.8 any_to_sint / any_to_usint

Operator - Converts the input into a short (8 bit) integer value.

Can be unsigned with `any_to_usint`.

4.5.8.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	ANY				Input value.

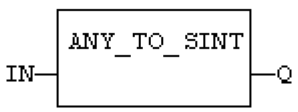
4.5.8.2 Outputs

Output	Data Type	Range	Unit	Description
Q	SINT			Value converted to a signed short integer. (8-bit).
Q	USINT			Value converted to an unsigned short integer. (8-bit).

4.5.8.3 Remarks

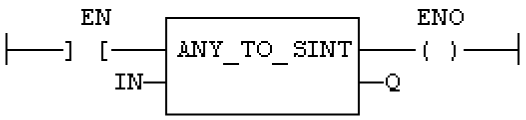
- For BOOL input data types, the output is 0 (zero) or 1.
- For REAL input data type, the output is the integer part of the input real.
- For TIME input data types, the result is the number of milliseconds.
- For STRING input data types, the output is the number represented by the string or 0 (zero) if the string does not represent a valid number.

4.5.8.4 FBD Language



4.5.8.5 FFLD Language

- In the FFLD language, the conversion is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.



4.5.8.6 IL Language

- In the IL Language, the `any_to_sint` converts the current result.

```
Op1: FFLD  IN
     ANY_TO_SINT
     ST  Q
```

4.5.8.7 ST Language

```
Q := ANY_TO_SINT (IN);
```

See Also

- any_to_bool
- any_to_dint / any_to_udint
- any_to_int / any_to_uint
- any_to_lint / any_to_ulint
- any_to_lreal
- any_to_real
- any_to_string
- any_to_time

4.5.9 any_to_string

ANY_TO_STRING

PLCopen ✓

Operator - Converts the input into string value.

4.5.9.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	ANY				Input value.

4.5.9.2 Outputs

Output	Data Type	Range	Unit	Description
Q	STRING			Value converted to a string.

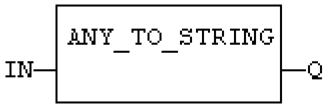
4.5.9.3 Remarks

- For BOOL input data types, the output is
  - 0 for FALSE.
  - 1 for TRUE.

- For DINT, REAL, or TIME input data types, the output is the string representation of the input

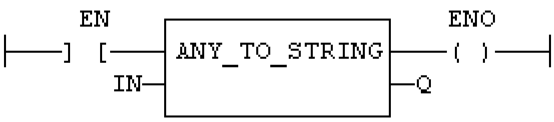
number.
- This is a number of milliseconds for TIME inputs.

4.5.9.4 FBD Language



4.5.9.5 FFLD Language

- In the FFLD language, the conversion is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.



4.5.9.6 IL Language

- In the IL language, the `any_to_string` function converts the current result.

```
Op1: FFLD  IN
     ANY_TO_STRING
     ST  Q
```

4.5.9.7 ST Language

```
Q := ANY_TO_STRING (IN);
```

See Also

- any_to_bool
- any_to_dint / any_to_udint
- any_to_int / any_to_uint
- any_to_lint / any_to_ulint
- any_to_lreal
- any_to_real
- any_to_sint / any_to_usint
- any_to_time

4.5.10 NUM_TO_STRING

PLCopen ✓

Function- Converts a number into string value.

4.5.10.1 Inputs

IN : ANY Input number.
WIDTH : DINT Length of the output string (see remarks).
DIGITS : DINT Number of digits after decimal point.

4.5.10.2 Outputs

Q : STRING Value converted to string.

4.5.10.3 Remarks

This function converts any numerical value to a string. Unlike the ANY_TO_STRING function, it allows you to specify a length and a number of digits after the decimal points.

- If **WIDTH** is 0, the string is formatted with the necessary length.

```
Q := NUM_TO_STRING (1.333333, 0, 2);    (* Q is '1.33' *)
```

- If **WIDTH** is greater than 0, the string is completed with leading blank characters in order to match the value of WIDTH.

```
Q := NUM_TO_STRING (123.4, 8, 2);    (* Q is '  123.40' *)
```

- If **WIDTH** is greater than 0, the string is completed with trailing blank characters in order to match the value of WIDTH.

```
Q := NUM_TO_STRING (123.4, -8, 2);    (* Q is '123.40  ' *)
```

- If **DIGITS** is 0 then neither decimal part nor decimal point are added.

```
Q := NUM_TO_STRING (1.333333, 3, 0);    (* Q is '  1' *)
```

- If **DIGITS** is greater than 0, the corresponding number of decimal digits are added. '0' digits are added if necessary

```
Q := NUM_TO_STRING (1.333333, 0, 1);    (* Q is '1.3' *)
```

- If the value is too long for the specified width, then the string is filled with '*' characters.

```
Q := NUM_TO_STRING (1234, 3, 0);    (* Q is '***' *)
```

4.5.11 bcd_to_bin

PLCopen

Function - Converts a BCD (Binary Coded Decimal) value to a binary value.

4.5.11.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	DINT				Integer value in BCD.

4.5.11.2 Outputs

Output	Data Type	Range	Unit	Description
Q	DINT			Value converted to integer or 0 (zero) if IN is not a valid positive BCD value.

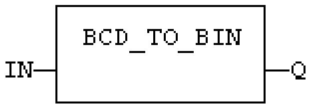
4.5.11.2.1 Truth Table

IN	Q
-2	0 (invalid)
0	0
16 (16#10)	10
15 (16#0F)	0 (invalid)

4.5.11.3 Remarks

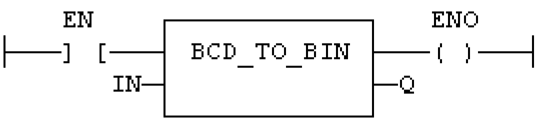
- The input must:
  - be positive.
  - represent a valid BCD value.

4.5.11.4 FBD Language



4.5.11.5 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.



4.5.11.6 IL Language

- In the IL language, the input must be loaded in the current result before calling the function.

```
Op1: LD          IN
     BCD_TO_BIN
     ST          Q
```

4.5.11.7 ST Language

```
Q := BCD_TO_BIN (IN);
```

See Also

bin_to_bcd

4.5.12 bin_to_bcd

PLCopen

Function - Converts a binary value to a BCD (Binary Coded Decimal) value.

4.5.12.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	DINT				Integer value.

4.5.12.2 Outputs

Output	Data Type	Range	Unit	Description
Q	DINT			Value converted to BCD or 0 if IN is less than 0.

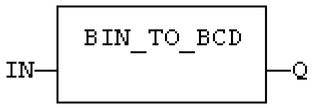
4.5.12.3 Truth Table

IN	Q
-2	0 (invalid)
0	0
10	16 (16#10)
22	34 (16#22)

4.5.12.4 Remarks

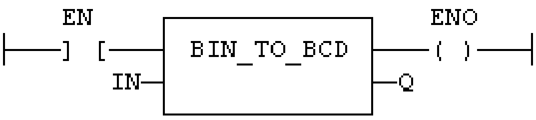
- The input must be positive.

4.5.12.5 FBD Language



4.5.12.6 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.



4.5.12.7 IL Language

- In the IL language, the input must be loaded in the current result before calling the function.

```
Op1: LD          IN
     BIN_TO_BCD
     ST          Q
```

4.5.12.8 ST Language

```
Q := BIN_TO_BCD (IN);
```

See Also

bcd_to_bin

4.6 Selectors

These are the standard functions that perform data selection:

Function	Description
"SEL" (➜ p. 131)	2 integer inputs
"MUX4" (➜ p. 127)	4 integer inputs
"MUX8" (➜ p. 129)	8 integer inputs

4.6.1 MUX4

PLCopen ✓

Function - Select one of the inputs - 4 inputs.

4.6.1.1 Inputs

Inputs	Data Type	Description
K	DINT	Selection command.
IN0	ANY	First input.
IN1	ANY	Second input.
IN2	ANY	Third input.
IN3	ANY	Last input.

4.6.1.2 Outputs

Output	Data Type	Description
Q	ANY	IN0 or IN1 ... or IN3 depending on K. See "Truth Table" (➜ p. 127).

4.6.1.3 Truth Table

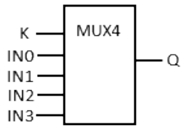
K	Q
0	IN0
1	IN1
2	IN2
3	IN3
Other	0

4.6.1.4 Remarks

- In FFLD language, the input rung (EN) enables the selection.
  - The output rung keeps the state of the input rung.

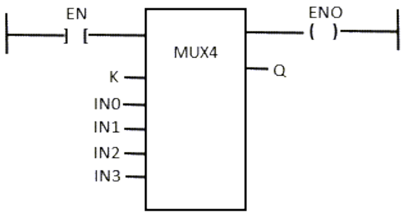
- In IL language, the first parameter (selector) must be loaded in the current result before calling the function.
  - Other inputs are operands of the function separated by comas.

4.6.1.5 FBD Language



4.6.1.6 FFLD Language

(* the selection is performed only if EN is TRUE. *)
(* ENO has the same value as EN. *)



4.6.1.7 IL Language

```
Op1: LD   SELECT
     MUX4 IN1, IN2, IN3, IN4
     ST   Q
```

4.6.1.8 ST Language

```
Q := MUX4 (K, IN0, IN1, IN2, IN3);
```

See Also

- "MUX8" (➜ p. 129)
- "SEL" (➜ p. 131)

4.6.2 MUX8

PLCopen

Function - Select one of the inputs - 8 inputs.

4.6.2.1 Inputs

Inputs	Data Type	Description
K	DINT	Selection command.
IN0	ANY	First input.
IN1	ANY	Second input.
IN2	ANY	Third input.
IN3	ANY	Fourth input.
IN4	ANY	Fifth input.
IN5	ANY	Sixt input.
IN6	ANY	Seventh input.
IN7	ANY	Last input.

4.6.2.2 Outputs

Output	Data Type	Description
Q	ANY	IN0 or IN1 ... or IN7 depending on K. See "Truth Table" (→ p. 129).

4.6.2.3 Truth Table

K	Q
0	IN0
1	IN1
2	IN2
3	IN3
4	IN4
5	IN5
6	IN6
7	IN7
Other	0

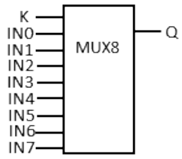
4.6.2.4 Remarks

- In FFLD language, the input rung (EN) enables the selection.
  - The output rung keeps the state of the input rung.
- In IL language, the first parameter (selector) must be loaded in the current result before calling the function.
  - Other inputs are operands of the function separated by comas.

4.6.2.5 ST Language

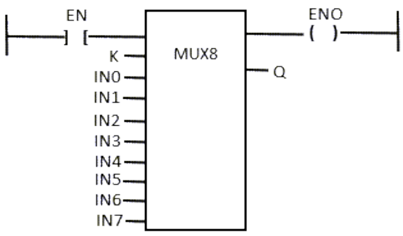
```
Q := MUX8 (K, IN0, IN1, IN2, IN3, IN4, IN5, IN6, IN7);
```

4.6.2.6 FBD Language



4.6.2.7 FFLD Language

(* the selection is performed only if EN is TRUE *)
(* ENO has the same value as EN *)



4.6.2.8 IL Language

Not available.

```
Op1: LD   SELECT
     MUX8 IN1, IN2, IN3, IN4, IN5, IN6, IN7, IN8
     ST   Q
```

See Also

- "MUX4" (→ p. 127)
- "SEL" (→ p. 131)

4.6.3 SEL

Function - Select one of the inputs - 2 inputs.

4.6.3.1 Inputs

Inputs	Data Type	Description
G	BOOL	Selection command.
IN0	ANY	First input.
IN1	ANY	Second input.

4.6.3.2 Outputs

Output	Data Type	Description
Q	ANY	• IN0 if G is FALSE • IN1 if G is TRUE

4.6.3.3 Truth Table

SELECT	Q
0	IN0
1	IN1

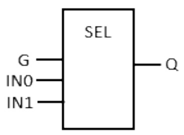
4.6.3.4 Remarks

- In FFLD language, the selector command is the input rung.
  - The output rung keeps the same state as the input rung.
- In IL language, the first parameter (selector) must be loaded in the current result before calling the function.
  - Other inputs are operands of the function separated by comas.

4.6.3.5 ST Language

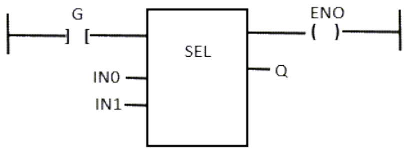
```
Q := SEL (G, IN0, IN1);
```

4.6.3.6 FBD Language



4.6.3.7 FFLD Language

(* the input rung is the selector *)
(* ENO has the same value as SELECT *)



4.6.3.8 IL Language

```
Op1: LD   SELECT
     SEL IN1, IN2
     ST   Q
```

See Also

- "MUX4" (→ p. 127)
- "MUX8" (→ p. 129)

4.7 Registers

4.7.1 Standard Functions

These are the standard functions for managing 8- to 32-bit registers:

Function	Description
"rol" (➜ p. 147)	Rotate bits of a register to the left.
"ror" (➜ p. 149)	Rotate bits of a register to the right.
"shl" (➜ p. 154)	Shift bits of a register to the left.
"shr" (➜ p. 155)	Shift bits of a register to the right.

4.7.2 Advanced Function

This is the advanced function for register manipulation:

Function	Description
"MBshift" (➜ p. 142)	Multi-byte shift / rotate.

4.7.3 Bit-to-Bit Functions

These functions enable bit-to-bit operations on a 8- to 32-bit integers:

Function	Description
"and_mask" (➜ p. 134)	Performs a bit-to-bit Boolean AND between two integer values.
"not_mask" (➜ p. 144)	Performs a bit-to-bit Boolean negation of an integer value.
"or_mask" (➜ p. 145)	Performs a bit-to-bit Boolean OR between two integer values.
"xor_mask" (➜ p. 160)	Performs a bit to bit exclusive OR between two integer values.

4.7.4 Pack / Unpack Functions

These functions enable pack / unpack 8-, 16-, and 32-bit registers:

Function	Description
HiByte	Get the highest byte of a word.
HiWord	Get the highest word of a double word.
LoByte	Get the lowest byte of a word.

Function	Description
LoWord	Get the lowest word of a double word.
MakeDWord	Builds a double word as the concatenation of two words.
MakeWord	Builds a word as the concatenation of two bytes.
PACK8	Pack bits in a byte.
UNPACK8	Extract bits from a byte.

4.7.5 Bit Access

These functions provide bit access in 8- to 32-bit integers:

Function	Description
SetBit	Set a bit in an integer register.
TestBit	Test a bit of an integer register.

4.7.6 Deprecated Functions

These functions have been deprecated.

- They are available for backwards compatibility only.
- The previous functions should be used for all current and future development.

AND_WORD	AND_BYTE
NOT_WORD	NOT_BYTE
OR_WORD	OR_BYTE
ROLB	RORB
ROLW	RORW
SHLB	SHRB
SHLW	SHRW
XOR_WORD	XOR_BYTE

4.7.7 and_mask

PLCopen

Function - Performs a bit-to-bit Boolean AND between two integer values.

4.7.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	ANY				First input.
MSK	ANY				Second input. (AND mask)

4.7.7.2 Outputs

Output	Data Type	Range	Unit	Description
Q	ANY			AND mask between IN and MSK inputs.

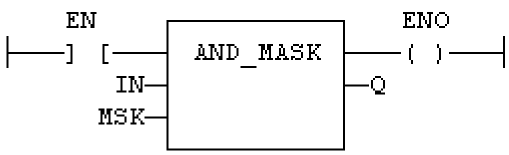
4.7.7.3 Remarks

- Arguments can be signed or unsigned integers from 8- to 32-bits.

4.7.7.4 FBD Language

4.7.7.5 FFLD Language

- In the FFLD language, the input rung (EN) enables the operation.
  - The output rung (ENO) keeps the same value as the input rung.
  - The function is executed only if EN is TRUE.
  - ENO is equal to EN.



4.7.7.6 IL Language

- In the IL language, the first parameter (IN) must be loaded in the current result before calling the function.
  - The other input is the operands of the function.

```
Op1: LD       IN
     AND_MASK MSK
     ST       Q
```

4.7.7.7 ST Language

```
Q := AND_MASK (IN, MSK);
```

See Also

- not_mask
- or_mask
- xor_mask

4.7.8 HiByte

PLCopen

Function - Get the highest byte of a word.

4.7.8.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	UINT				16-bit register.

4.7.8.2 Outputs

Output	Data Type	Range	Unit	Description
Q	USINT			Highest significant byte.

4.7.8.3 Remarks

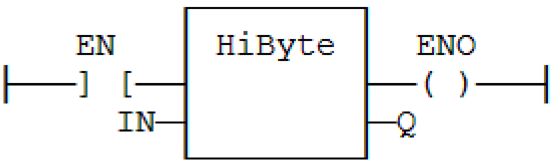
None

4.7.8.4 FBD Language

4.7.8.5 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.

- The function is executed only if EN is TRUE.
- ENO keeps the same value as EN.



4.7.8.6 IL Language

- In the IL language, the input must be loaded in the current result before calling the function.

```
Op1: LD  IN
     HIBYTE
     ST  Q
```

4.7.8.7 ST Language

```
Q := HIBYTE (IN);
```

See Also

- HiWord
- LoByte
- LoWord

- MakeDWord
- MakeWord

4.7.9 LoByte

PLCopen

Function - Get the lowest byte of a word.

4.7.9.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	UINT				16-bit register.

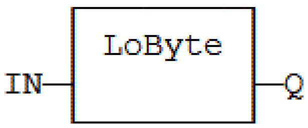
4.7.9.2 Outputs

Output	Data Type	Range	Unit	Description
Q	USINT			Lowest significant byte.

4.7.9.3 Remarks

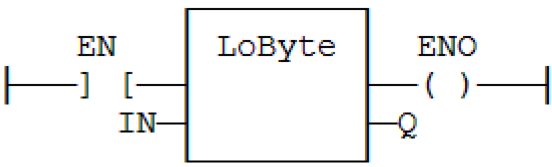
None

4.7.9.4 FBD Language



4.7.9.5 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.
- The function is executed only if EN is TRUE.
- ENO keeps the same value as EN.



4.7.9.6 IL Language

- In the IL language, the input must be loaded in the current result before calling the function.

```
Op1: LD   IN
     LOBYTE
     ST   Q
```

4.7.9.7 ST Language

```
Q := LOBYTE (IN);
```

See Also

- HiByte
- HiWord
- LoWord
- MakeDWord
- MakeWord

4.7.10 HiWord

PLCopen

Function - Get the highest word of a double word.

4.7.10.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	UDINT				32-bit register.

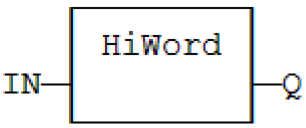
4.7.10.2 Outputs

Output	Data Type	Range	Unit	Description
Q	UINT			Highest significant word.

4.7.10.3 Remarks

None

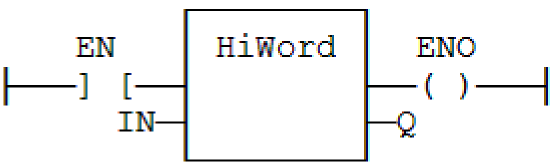
4.7.10.4 FBD Language



4.7.10.5 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.

- The function is executed only if EN is TRUE.
- ENO keeps the same value as EN.



4.7.10.6 IL Language

- In the IL language, the input must be loaded in the current result before calling the function.

```
Op1: LD   IN
     HIWORD
     ST   Q
```

4.7.10.7 ST Language

```
Q := HIWORD (IN);
```

See Also

- HiByte
- LoByte
- LoWord
- MakeDWord
- MakeWord

4.7.11 LoWord

PLCopen ✓

Function - Get the lowest word of a double word.

4.7.11.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	UDINT				32-bit register.

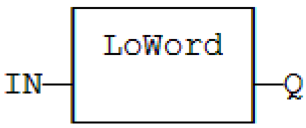
4.7.11.2 Outputs

Output	Data Type	Range	Unit	Description
Q	UINT			Lowest significant word.

4.7.11.3 Remarks

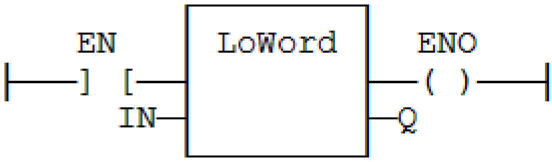
None

4.7.11.4 FBD Language



4.7.11.5 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.
- The function is executed only if EN is TRUE.
- ENO keeps the same value as EN.



4.7.11.6 IL Language

- In the IL language, the input must be loaded in the current result before calling the function.

```
Op1: LD   IN
     LOWORD
     ST   Q
```

4.7.11.7 ST Language

```
Q := LOWORD (IN);
```

See Also

- HiByte
- HiWord
- LoByte
- MakeDWord
- MakeWord

4.7.12 MakeDWord

Function - Builds a double word as the concatenation of two words.

4.7.12.1 Inputs

Input	Data Type	Range	Unit	Default	Description
HI	USINT				Highest significant word.
LO	USINT				Lowest significant word.

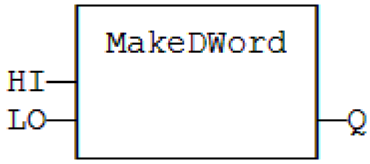
4.7.12.2 Outputs

Output	Data Type	Range	Unit	Description
Q	UINT			32-bit register.

4.7.12.3 Remarks

None

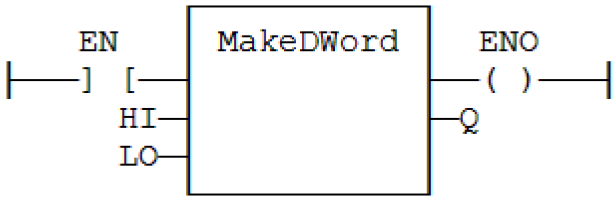
4.7.12.4 FBD Language



4.7.12.5 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.

- The function is executed only if EN is TRUE.
- ENO keeps the same value as EN.



4.7.12.6 IL Language

- In the IL language, the input must be loaded in the current result before calling the function.

```
Op1: LD        HI
     MAKEDWORD LO
     ST        Q
```

4.7.12.7 ST Language

```
Q := MAKEDWORD (HI, LO);
```

See Also

- HiByte
- HiWord
- LoByte
- LoWord
- MakeWord

4.7.13 MakeWord

PLCopen

Function - Builds a word as the concatenation of two bytes.

4.7.13.1 Inputs

Input	Data Type	Range	Unit	Default	Description
HI	USINT				Highest significant byte.
LO	USINT				Lowest significant byte.

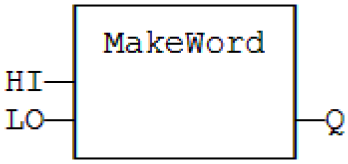
4.7.13.2 Outputs

Output	Data Type	Range	Unit	Description
Q	UINT			16-bit register.

4.7.13.3 Remarks

None

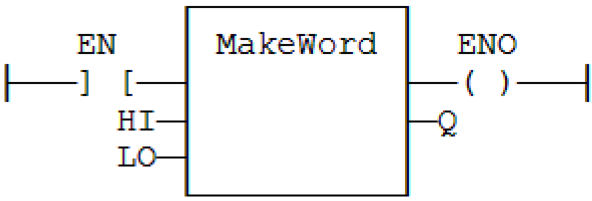
4.7.13.4 FBD Language



4.7.13.5 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.

- The function is executed only if EN is TRUE.
- ENO keeps the same value as EN.



4.7.13.6 IL Language

- In the IL language, the input must be loaded in the current result before calling the function.

```
Op1: LD       HI
     MAKEWORD LO
     ST       Q
```

4.7.13.7 ST Language

```
Q := MAKEWORD (HI, LO);
```

See Also

- HiByte
- HiWord
- LoByte
- LoWord
- MakeDWord

4.7.14 MBshift

PLCopen

Function - Multi-byte shift / rotate.

4.7.14.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Buffer	SINT / USINT				Array of bytes.
Pos	DINT				Base position in the array.
NbByte	DINT				Number of bytes to be shifted or rotated.
NbShift	DINT				Number of shifts or rotations.
ToRight	BOOL	TRUE, FALSE			• TRUE for right. • FALSE for left.
Rotate	BOOL	TRUE, FALSE			• TRUE for rotate. • FALSE for shift.
InBit	BOOL	TRUE, FALSE			Bit to be introduced in a shift.

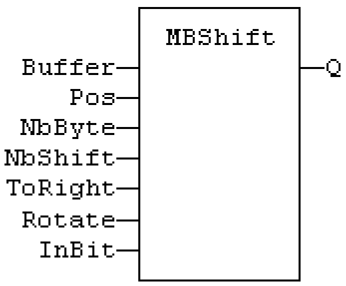
4.7.14.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL			TRUE if successful.

4.7.14.3 Remarks

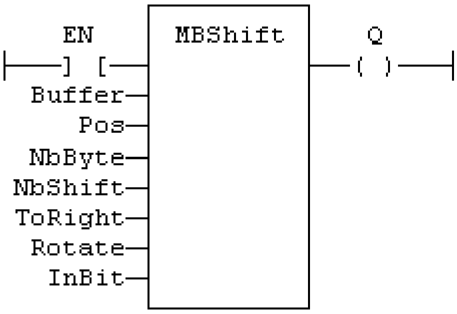
- Use the **ToRight** argument to specify a shift to the left (FALSE) or to the right (TRUE).
- Use the **Rotate** argument to specify either a shift (FALSE) or a rotation (TRUE).
- In case of a shift, the **InBit** argument specifies the value of the bit that replaces the last shifted bit.

4.7.14.4 FBD Language



4.7.14.5 FFLD Language

- In the FFLD language, the input rung (EN) enables the operation.
- The rung output is the result ("Q").
- The function is called only if EN is TRUE.



4.7.14.6 IL Language

Not available.

4.7.14.7 ST Language

```
Q := MBShift (Buffer, Pos, NbByte, NbShift, ToRight,
Rotate, InBit);
```

4.7.15 not_mask

PLCopen ✓

Function - Performs a bit-to-bit Boolean negation of an integer value.

4.7.15.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	ANY				Integer input.

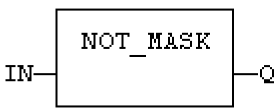
4.7.15.2 Outputs

Output	Data Type	Range	Unit	Description
Q	ANY			Bit to bit negation of the input.

4.7.15.3 Remarks

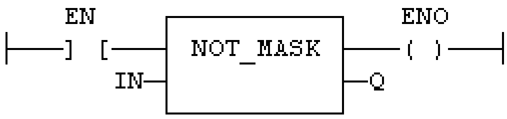
- Arguments can be signed or unsigned integers from 8- to 32-bits.

4.7.15.4 FBD Language



4.7.15.5 FFLD Language

- In the FFLD language, the input rung (EN) enables the operation.
  - The output rung keeps the state of the input rung.
  - The function is executed only if EN is TRUE.
  - ENO has the same value as EN.



4.7.15.6 IL Language

- In the IL language, the first parameter (IN) must be loaded in the current result before calling the function.
  - The other input is the operands of the function.

```
Op1: LD        IN
     NOT_MASK
     ST        Q
```

4.7.15.7 ST Language

```
Q := NOT_MASK (IN);
```

See Also

- and_mask
- or_mask
- xor_mask

4.7.16 or_mask

PLCopen ✓

Function - Performs a bit-to-bit Boolean OR between two integer values.

4.7.16.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	ANY				First input.
MSK	ANY				Second input. (OR mask)

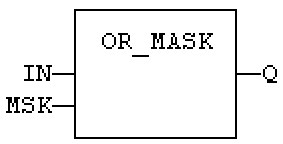
4.7.16.2 Outputs

Output	Data Type	Range	Unit	Description
Q	ANY			OR mask between IN and MSK inputs.

4.7.16.3 Remarks

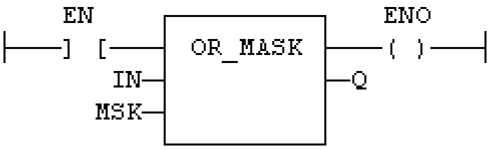
- Arguments can be signed or unsigned integers from 8- to 32-bits.

4.7.16.4 FBD Language



4.7.16.5 FFLD Language

- In the FFLD language, the input rung (EN) enables the operation.
  - The output rung keeps the state of the input rung.
  - The function is executed only if EN is TRUE.
  - ENO has the same value as EN.



4.7.16.6 IL Language

- In the IL language, the first parameter (IN) must be loaded in the current result before calling the function.
  - The other input is the operands of the function.

```
Op1: LD       IN
     OR_MASK  MSK
     ST       Q
```

4.7.16.7 ST Language

```
Q := OR_MASK (IN, MSK);
```

See Also

- and_mask
- not_mask
- xor_mask

4.7.17 PACK8

PLCopen

Function - Pack bits in a byte.

4.7.17.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN0	BOOL				Less significant bit.
IN7	BOOL				Highest significant bit.

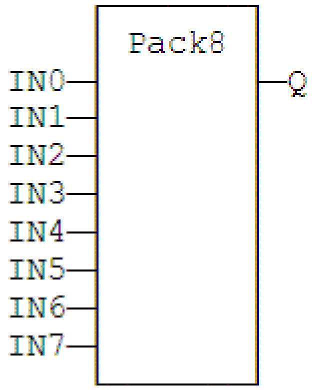
4.7.17.2 Outputs

Output	Data Type	Range	Unit	Description
Q	USINT			Byte built with input bits.

4.7.17.3 Remarks

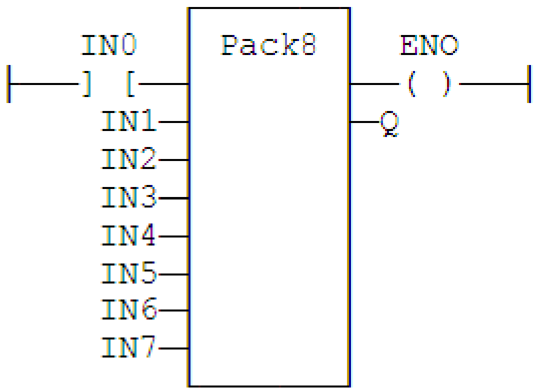
None

4.7.17.4 FBD Language



4.7.17.5 FFLD Language

- In the FFLD language, the input rung is the IN0 input.
  - The output rung (ENO) keeps the same value as the input rung.
- ENO keeps the same value as EN.



4.7.17.6 IL Language

- In the IL language, the input must be loaded in the current result before calling the function.

```
Op1: LD    IN0
     PACK8 IN1, IN2, IN3, IN4, IN5, IN6, IN7
     ST    Q
```

4.7.17.7 ST Language

```
Q := PACK8 (IN0, IN1, IN2, IN3, IN4, IN5, IN6, IN7);
```

See Also

UNPACK8

4.7.18 rol

Function - Rotate bits of a register to the left.

4.7.18.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	ANY				Register.
NBR	DINT				Number of rotations (each rotation is 1 bit).

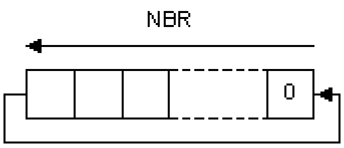
4.7.18.2 Outputs

Output	Data Type	Range	Unit	Description
Q	ANY			Rotated register.

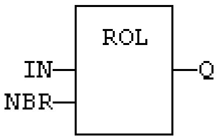
4.7.18.3 Remarks

- Arguments can be signed or unsigned integers from 8- to 32-bits.

4.7.18.3.1 Diagram

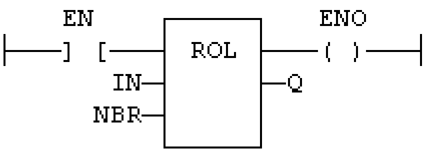


4.7.18.4 FBD Language



4.7.18.5 FFLD Language

- In the FFLD language, the input rung (EN) enables the operation.
  - The output rung keeps the state of the input rung.
  - The rotation is executed only if EN is TRUE.
  - ENO keeps the same value as EN.



4.7.18.6 IL Language

- In the IL language, the first input must be loaded before the function call.
  - The second input is the operand of the function.

```
Op1: LD  IN
     ROL NBR
     ST  Q
```

4.7.18.7 ST Language

```
Q := ROL (IN, NBR);
```

See Also

- "ror" (→ p. 149)
- "shl" (→ p. 154)
- "shr" (→ p. 155)
- These other links were in the Copa-Data topic but not ours - should they be added: SHRb ROLb SHLw SHRw ROLw

4.7.19 ror

PLCopen ✓

Function - Rotate bits of a register to the right.

4.7.19.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	ANY				Register.
NBR	ANY				Number of rotations (each rotation is 1 bit).

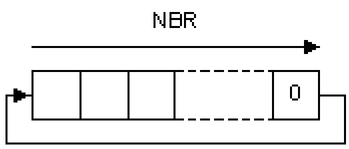
4.7.19.2 Outputs

Output	Data Type	Range	Unit	Description
Q	ANY			Rotated register.

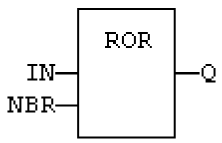
4.7.19.3 Remarks

- Arguments can be signed or unsigned integers from 8- to 32-bits.

4.7.19.3.1 Diagram



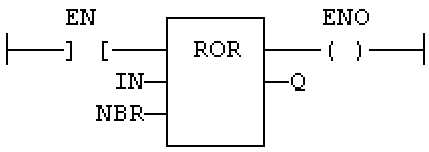
4.7.19.4 FBD Language



4.7.19.5 FFLD Language

- In the FFLD language, the input rung (EN) enables the operation.

- The output rung keeps the state of the input rung.
- The rotation is executed only if EN is TRUE.
- ENO keeps the same value as EN.



4.7.19.6 IL Language

- In the IL language, the first input must be loaded before the function call.
  - The second input is the operand of the function.

```
Op1: LD  IN
     ROR NBR
     ST  Q
```

4.7.19.7 ST Language

```
Q := ROR (IN, NBR);
```

See Also

- "rol" (➜ p. 147)
- "shl" (➜ p. 154)
- "shr" (➜ p. 155)
- These other links were in the Copa-Data topic but not ours - should they be added: SHRb ROLb SHLw SHRw ROLw

4.7.20 RORb / ROR_SINT / ROR_USINT / ROR_BYTE

Function - Rotate bits of a register to the right.

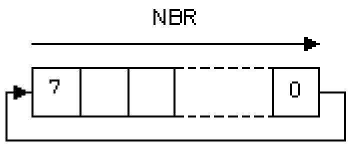
4.7.20.1 Inputs

`IN : SINT` 8 bit register
`NBR : SINT` Number of rotations (each rotation is 1 bit)

4.7.20.2 Outputs

`Q : SINT` Rotated register

4.7.20.3 Diagram



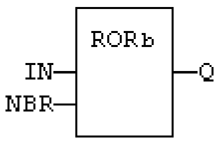
4.7.20.4 Remarks

In FFLD language, the input rung (EN) enables the operation, and the output rung keeps the state of the input rung. In IL language, the first input must be loaded before the function call. The second input is the operand of the function.

4.7.20.5 ST Language

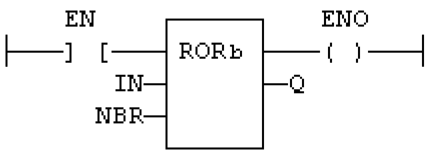
Q := RORb (IN, NBR);

4.7.20.6 FBD Language



4.7.20.7 FFLD Language

(* The rotation is executed only if EN is TRUE *)
(* ENO has the same value as EN *)



4.7.20.8 IL Language

```
Op1: FFLD   IN
     RORb NBR
     ST   Q
```

4.7.20.9 See also

SHL SHR ROL ROR SHLb SHRb ROLb SHLw SHRw ROLw RORw

4.7.21 RORw / ROR_INT / ROR_UINT / ROR_WORD

Function - Rotate bits of a register to the right.

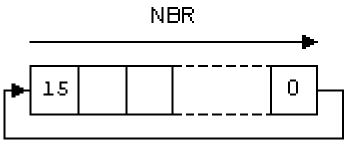
4.7.21.1 Inputs

`IN : INT` 16 bit register
`NBR : INT` Number of rotations (each rotation is 1 bit)

4.7.21.2 Outputs

`Q : INT` Rotated register

4.7.21.3 Diagram



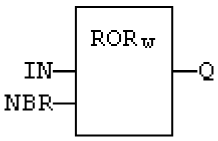
4.7.21.4 Remarks

In FFLD language, the input rung (EN) enables the operation, and the output rung keeps the state of the input rung. In IL language, the first input must be loaded before the function call. The second input is the operand of the function.

4.7.21.5 ST Language

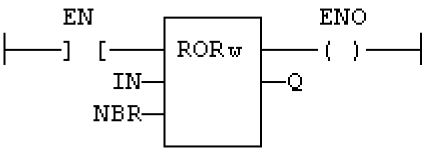
Q := RORw (IN, NBR);

4.7.21.6 FBD Language



4.7.21.7 FFLD Language

(* The rotation is executed only if EN is TRUE *)
(* ENO has the same value as EN *)



4.7.21.8 IL Language

```
Op1: FFLD  IN
     RORw NBR
     ST   Q
```

4.7.21.9 See also

SHL SHR ROL ROR SHLb SHRb ROLb RORb SHLw SHRw ROLw

4.7.22 SetBit

PLCopen

Function - Set a bit in an integer register.

4.7.22.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	ANY				8- to 64-bit integer register.
BIT	DINT				Bit number (0 = less significant bit).
VAL	BOOL				Bit value to apply.

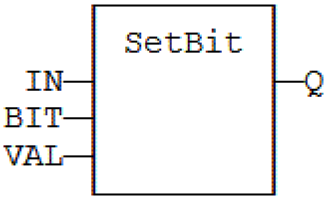
4.7.22.2 Outputs

Output	Data Type	Range	Unit	Description
Q	ANY			Modified register.

4.7.22.3 Remarks

- Types LINT, LREAL, REAL, STRING, and TIME are not supported for IN and Q.
- IN and Q must have the same type.
- In case of invalid arguments (e.g., bad bit number or invalid input type), the function returns the value of IN without modification.

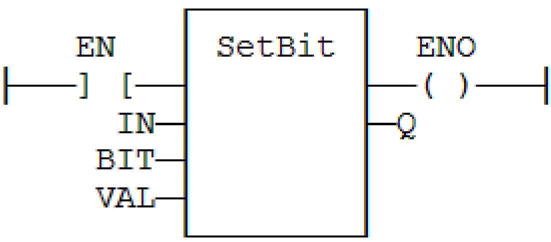
4.7.22.4 FBD Language



4.7.22.5 FFLD Language

- In the FFLD language, the conversion is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.

- The function is executed only if EN is TRUE.
- ENO keeps the same value as EN.



4.7.22.6 IL Language

Not available.

4.7.22.7 ST Language

```
Q := SETBIT (IN, BIT, VAL);
```

See Also

"TestBit" (➜ p. 158)

4.7.23 shl

PLCopen ✓

Function - Shift bits of a register to the left.

4.7.23.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	ANY				Register.
NBS	ANY				Number of shifts (each shift is 1 bit).

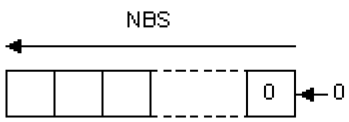
4.7.23.2 Outputs

Output	Data Type	Range	Unit	Description
Q	ANY			Shifted register.

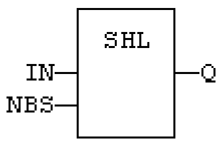
4.7.23.3 Remarks

- Arguments can be signed or unsigned integers from 8- to 32-bits.

4.7.23.3.1 Diagram

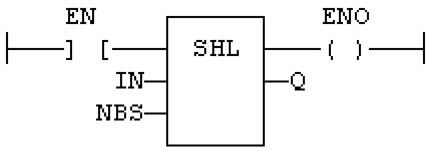


4.7.23.4 FBD Language



4.7.23.5 FFLD Language

- In the FFLD language, the input rung (EN) enables the operation.
  - The output rung keeps the state of the input rung.
  - The shift is executed only if EN is TRUE.
  - ENO has the same value as EN.



4.7.23.6 IL Language

- In the IL language, the first input must be loaded before the function call.
  - The second input is the operand of the function.

```
Op1: LD  IN
     SHL NBS
     ST Q
```

4.7.23.7 ST Language

```
Q := SHL (IN, NBS);
```

See Also

- "rol" (→ p. 147)
- "ror" (→ p. 149)
- "shr" (→ p. 155)
- These other links were in the Copa-Data topic but not ours - should they be added: SHRb ROLb SHLw SHRw ROLw

4.7.24 shr

PLCopen

Function - Shift bits of a register to the right.

4.7.24.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	ANY				Register.
NBS	ANY				Number of shifts (each shift is 1 bit).

4.7.24.2 Outputs

Output	Data Type	Range	Unit	Description
Q	ANY			Shifted register.

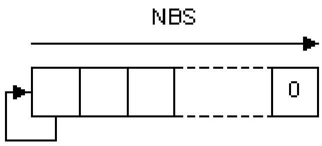
4.7.24.3 Remarks

- Arguments can be signed or unsigned integers from 8- to 32-bits.

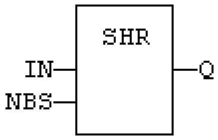
4.7.24.3.1 Diagram

- these bullets are in the Copa-Data topic but not ours - should we add it?
- If the option **SHR: do not duplicate the most significant bit** is checked in the Project settings

/ Advanced box, then the most significant bit is set to FALSE.
- If the option is not checked, then the most significant bit is duplicated:

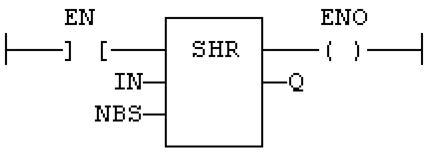


4.7.24.4 FBD Language



4.7.24.5 FFLD Language

- In the FFLD language, the input rung (EN) enables the operation.
  - The output rung keeps the state of the input rung.
  - The shift is executed only if EN is TRUE.
  - ENO has the same value as EN.



4.7.24.6 IL Language

- In the IL language, the first input must be loaded before the function call.
  - The second input is the operand of the function.

```
Op1: LD  IN
     SHR NBS
     ST  Q
```

4.7.24.7 ST Language

```
Q := SHR (IN, NBS);
```

See Also

- "rol" (➔ p. 147)
- "ror" (➔ p. 149)
- "shl" (➔ p. 154)
- These other links were in the Copa-Data topic but not ours - should they be added: SHRb ROLb SHLw SHRw ROLw

4.7.25 SWAB

PLCopen ✓

Function - Swap the bytes of an integer.

4.7.25.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	ANY	No range	N/A	No default	Any signed or unsigned integer.

4.7.25.2 Outputs

Output	Data Type	Range	Unit	Description
Q	ANY	No range	N/A	Swapped value.

4.7.25.3 Remarks

Supported data types are:

- DINT
- DWORD
- INT
- LINT
- LWORD
- SINT
- UDINT
- UINT
- ULINT
- USINT
- WORD

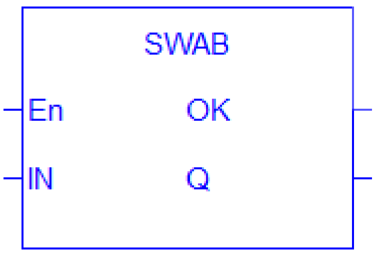
- SINT and USINT inputs result in the same output value because they are only 1 byte wide.
- If the function is called for another data type, the output takes the value of the input.

Examples

Type	IN	Q
WORD	16#1A2B	16#2B1A
DWORD	16#1A2B3C4D	16#4D3C2B1A

4.7.25.4 FBD Language

4.7.25.5 FFLD Language



4.7.25.6 IL Language

Not available.

4.7.25.7 ST Language

```
swappedValue := SWAB(value);
```

4.7.26 TestBit

PLCopen ✓

Function - Test a bit of an integer register.

4.7.26.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	ANY				8- to 64-bit integer register.
BIT	DINT				Bit number (0 = less significant bit).

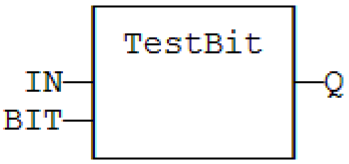
4.7.26.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL			Bit value.

4.7.26.3 Remarks

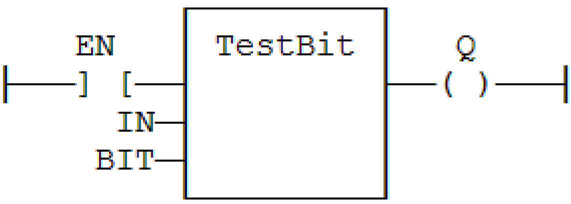
- Types LINT, LREAL, REAL, STRING, and TIME are not supported for IN and Q.
- IN and Q must have the same type.
- In case of invalid arguments (e.g., bad bit number or invalid input type), the function returns FALSE.

4.7.26.4 FBD Language



4.7.26.5 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung is the output of the function.
  - The function is executed only if EN is TRUE.



4.7.26.6 IL Language

Not available.

4.7.26.7 ST Language

```
Q := TESTBIT (IN, BIT);
```

See Also

"SetBit" (→ p. 153)

4.7.27 UNPACK8

PLCopen ✓

Function Block - Extract bits from a byte.

4.7.27.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	USINT				8-bit register.

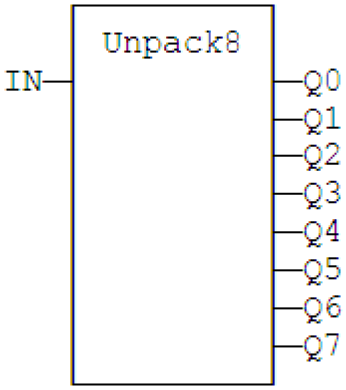
4.7.27.2 Outputs

Output	Data Type	Range	Unit	Description
Q0	BOOL			Less significant bit.
Q7	BOOL			Highest significant bit.

4.7.27.3 Remarks

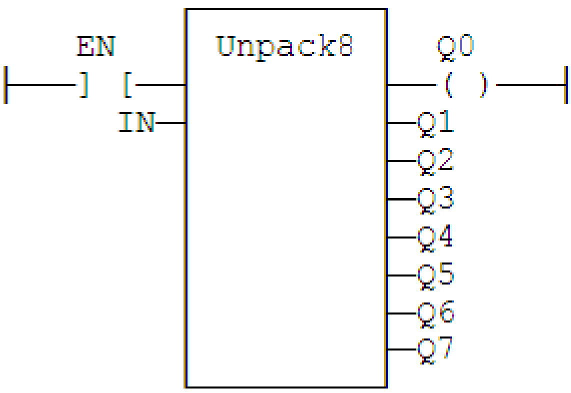
- The operation is executed only in the input rung (EN) is TRUE.

4.7.27.4 FBD Language



4.7.27.5 FFLD Language

- In the FFLD language, the output rung is the Q0 output.
- The operation is performed if EN = TRUE.



4.7.27.6 IL Language

```
(* MyUnpack is a declared instance of the UNPACK8 function block *)
Op1: CAL MyUnpack (IN)
     FFLD  MyUnpack.Q0
     ST   Q0
     (* ... *)
     FFLD  MyUnpack.Q7
     ST   Q7
```

4.7.27.7 ST Language

```
(* MyUnpack is a declared instance of the UNPACK8 function block *)
MyUnpack (IN);
Q0 := MyUnpack.Q0;
Q1 := MyUnpack.Q1;
Q2 := MyUnpack.Q2;
Q3 := MyUnpack.Q3;
Q4 := MyUnpack.Q4;
Q5 := MyUnpack.Q5;
Q6 := MyUnpack.Q6;
Q7 := MyUnpack.Q7;
```

See Also

"PACK8" (→ p. 146)

4.7.28 xor_mask

PLCopen ✓

Function - Performs a bit to bit exclusive OR between two integer values.

4.7.28.1 Inputs

Input	Data Type	Description
IN	ANY	First input.
MSK	ANY	Second input. (XOR mask)

4.7.28.2 Outputs

Output	Data Type	Description
Q	ANY	Exclusive OR mask between IN and MSK inputs.

4.7.28.3 Remarks

- Arguments can be signed or unsigned integers from 8- to 32-bits.

4.7.28.4 FBD Language

4.7.28.5 FFLD Language

- In the FFLD language, the input rung (EN) enables the operation.
  - The output rung keeps the state of the input rung.
  - The function is executed only if EN is TRUE.
  - ENO has the same value as EN.

4.7.28.6 IL Language

- In the IL language, the input must be loaded in the current result before calling the function.
  - The other input is the operands of the function.

```
Op1: LD       IN
     XOR_MASK MSK
     ST       Q
```

4.7.28.7 ST Language

```
Q := XOR_MASK (IN, MSK);
```

See Also

- "and_mask" (➔ p. 134)
- "not_mask" (➔ p. 144)
- "or_mask" (➔ p. 145)

4.8 Counters

These are the standard blocks for managing counters:

Function Block	Description
"CTD / CTDr" (➔ p. 161)	Down Counter
"CTU / CTUr" (➔ p. 163)	Up counter
"CTUD / CTUDr" (➔ p. 164)	Up / Down Counter

4.8.1 CTD / CTDr

PLCopen ✓

Function Block - Down counter.

4.8.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
CD	BOOL				Enable counting. Counter is decreased on each call when CD is TRUE.
LOAD	BOOL				Re-load command. Counter is set to PV when called with LOAD to TRUE.
PV	DINT				Programmed maximum value.

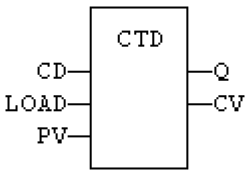
4.8.1.2 Outputs

Output	Data Type	Range	Unit	Description
CV	DINT			Current value of the counter.
Q	BOOL			TRUE when counter is empty (i.e., when CV = 0).

4.8.1.3 Remarks

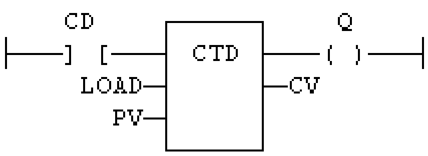
- The counter is empty (CV = 0) when the application starts.
- The counter does not include a pulse detection for CD input.
- Use the R_TRIG or F_TRIG function block for counting pulses of CD input signal.
- CTUr, CTDr, CTUDr function blocks operate exactly as other counters, except that all Boolean inputs (CU, CD, RESET, LOAD) have an implicit rising edge detection included.

4.8.1.4 FBD Language



4.8.1.5 FFLD Language

- In the FFLD language, CD is the input rung.
  - The output rung is the Q output.



4.8.1.6 IL Language

```
(* MyCounter is a declared instance of CTD function block. *)
Op1: CAL    MyCounter (CD, LOAD, PV)
FFLD      MyCounter.Q
ST      Q
```

```
LD      MyCounter.CV
    CV
```

4.8.1.7 ST Language

```
(* MyCounter is a declared instance of CTD function block. *)
MyCounter (CD, LOAD, PV);
Q := MyCounter.Q;
CV := MyCounter.CV;
```

See Also

- CTU / CTUr
- CTUD / CTUDr

4.8.2 CTU / CTUr

PLCopen ✓

Function Block - Up counter.

4.8.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
CU	BOOL				Enable counting. Counter is increased on each call when CU is TRUE.
PV	DINT				Programmed maximum value.
RESET	BOOL				Reset command. Counter is reset to 0 when called with RESET to TRUE.

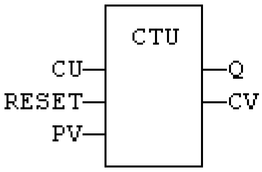
4.8.2.2 Outputs

Output	Data Type	Range	Unit	Description
CV	DINT			Current value of the counter.
Q	BOOL			TRUE when counter is full (i.e., when CV = PV).

4.8.2.3 Remarks

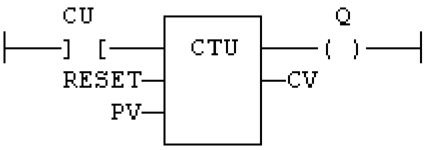
- The counter is empty (CV = 0) when the application starts.
- The counter does not include a pulse detection for CU input.
- Use the R_TRIG or F_TRIG function block for counting pulses of CU input signal.
- CTUr, CTDr, CTUDr function blocks operate exactly as other counters, except that all Boolean inputs (CU, CD, RESET, LOAD) have an implicit rising edge detection included.

4.8.2.4 FBD Language



4.8.2.5 FFLD Language

- In the FFLD language, CU is the input rung.
  - The output rung is the Q output.



4.8.2.6 IL Language

```
(* MyCounter is a declared instance of CTU function block. *)
Op1: CAL    MyCounter (CU, RESET, PV)
FFLD      MyCounter.Q
ST      Q
FFLD      MyCounter.CV
ST      CV
```

4.8.2.7 ST Language

```
(* MyCounter is a declared instance of CTU function block. *)
MyCounter (CU, RESET, PV);
Q := MyCounter.Q;
CV := MyCounter.CV;
```

See Also

- CTD / CTDr
- CTUD / CTUDr

4.8.3 CTUD / CTUDr

Function Block - Up/down counter.

4.8.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
CD	BOOL				Enable counting. Counter is decreased on each call when CD is TRUE.

Input	Data Type	Range	Unit	Default	Description
CU	BOOL				Enable counting. Counter is increased on each call when CU is TRUE.
LOAD	BOOL				Re-load command. Counter is set to PV when called with LOAD to TRUE.
PV	DINT				Programmed maximum value.
RESET	BOOL				Reset command. Counter is reset to 0 when called with RESET to TRUE.

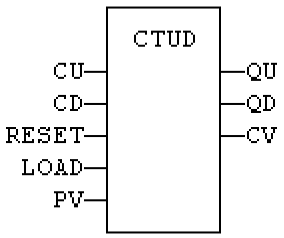
4.8.3.2 Outputs

Output	Data Type	Range	Unit	Description
CV	DINT			Current value of the counter.
QD	BOOL			TRUE when counter is empty (i.e., when CV = 0).
QU	BOOL			TRUE when counter is full (i.e., when CV = PV).

4.8.3.3 Remarks

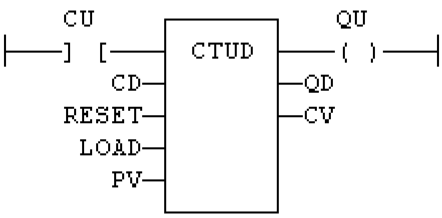
- The counter is empty (CV = 0) when the application starts.
- The counter does not include a pulse detection for CU and CD inputs.
- Use the R_TRIG or F_TRIG function blocks for counting pulses of CU or CD input signals.
- CTUr, CTDr, CTUDr function blocks operate exactly as other counters, except that all Boolean inputs (CU, CD, RESET, LOAD) have an implicit rising edge detection included.

4.8.3.4 FBD Language



4.8.3.5 FFLD Language

- In FFLD language, CU is the input rung.
  - The output rung is the QU output.



4.8.3.6 IL Language

```
(* MyCounter is a declared instance of CTUD function block. *)
Op1: CAL    MyCounter (CU, CD, RESET, LOAD, PV)
FFLD      MyCounter.QU
ST      QU
FFLD      MyCounter.QD
ST      QD
FFLD      MyCounter.CV
ST      CV
```

4.8.3.7 ST Language

```
(* MyCounter is a declared instance of CTUD function block. *)
MyCounter (CU, CD, RESET, LOAD, PV);
QU := MyCounter.QU;
QD := MyCounter.QD;
CV := MyCounter.CV;
```

See Also

"CTU / CTUr" (→ p. 163)

4.9 Timers

These are the standard functions for managing timers:

Function	Description
"BLINK" (→ p. 166)	Blinker
"BlinkA" (→ p. 167)	Asymmetric blinker
"PLS" (→ p. 169)	Pulse signal generator
"TMD" (→ p. 171)	Down-counting stop watch
"TMU / TMUsec" (→ p. 173)	Up-counting stop watch (seconds)
"TOF / TOFR" (→ p. 175)	Off timer
"TON" (→ p. 177)	On timer
"TP / TPR" (→ p. 178)	Pulse timer

4.9.1 BLINK PLCopen ✓

Function Block - Blinker.

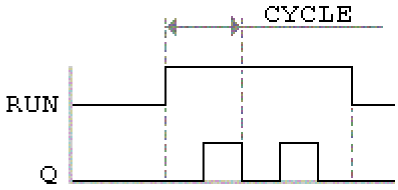
4.9.1.1 Inputs

`RUN : BOOL` Enabling command
`CYCLE : TIME` Blinking period

4.9.1.2 Outputs

`Q : BOOL` Output blinking signal

4.9.1.3 Time diagram



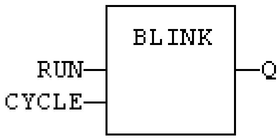
4.9.1.4 Remarks

The output signal is FALSE when the RUN input is FALSE. The CYCLE input is the complete period of the blinking signal. In FFLD language, the input rung is the IN command. The output rung is the Q output signal.

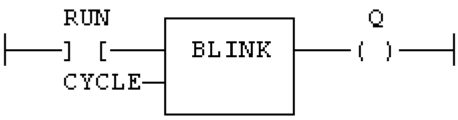
4.9.1.5 ST Language

```
(* MyBlinker is a declared instance of BLINK function block *)
MyBlinker (RUN, CYCLE);
Q := MyBlinker.Q;
```

4.9.1.6 FBD Language



4.9.1.7 FFLD Language



4.9.1.8 IL Language

```
(* MyBlinker is a declared instance of BLINK function block *)
Op1: CAL MyBlinker (RUN, CYCLE)
    FFLD  MyBlinker.Q
    ST  Q
```

(missing or bad snippet)

TON TOF TP

4.9.2 BlinkA

PLCopen ✓

Function Block - Asymmetric blinker.

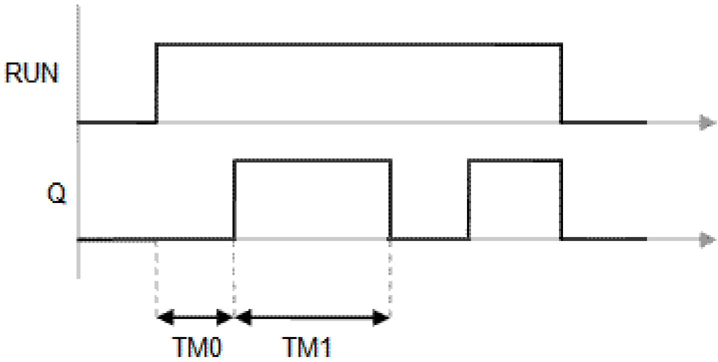
4.9.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
RUN	BOOL				Enabling command.
TM0	TIME				Duration of FALSE state on output.
TM1	TIME				Duration of TRUE state on output.

4.9.2.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL			Output blinking signal.

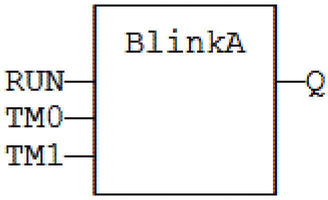
4.9.2.3 Time Diagram



4.9.2.4 Remarks

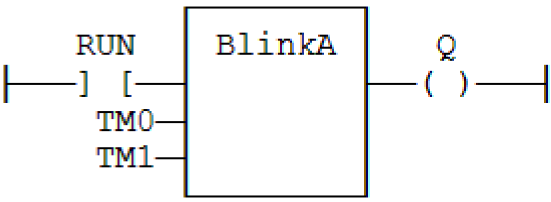
- The output signal is FALSE when the RUN input is FALSE.

4.9.2.5 FBD Language



4.9.2.6 FFLD Language

- In the FFLD language, the input rung is the IN command.
  - The output rung is the Q output.



4.9.2.7 IL Language

```
(* MyBlinker is a declared instance of BLINKA function block *)
Op1: CAL MyBlinker (RUN, TM0, TM1)
     FFLD  MyBlinker.Q
     ST   Q
```

4.9.2.8 ST Language

```
(* MyBlinker is a declared instance of BLINKA function block. *)
MyBlinker (RUN, TM0, TM1);
Q := MyBlinker.Q;
```

See Also

- TOF / TOFR
- TON
- TP / TPR

4.9.3 PLS

Function Block - Pulse signal generator

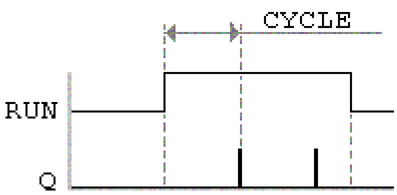
4.9.3.1 Inputs

`RUN : BOOL` Enabling command.
`CYCLE : TIME` Signal period.

4.9.3.2 Outputs

`Q : BOOL` Output pulse signal.

4.9.3.3 Time diagram



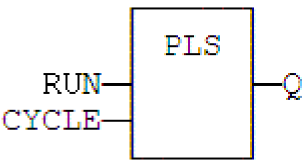
4.9.3.4 Remarks

On every period, the output is set to TRUE during one cycle only. In FFLD language, the input rung is the IN command. The output rung is the Q output signal.

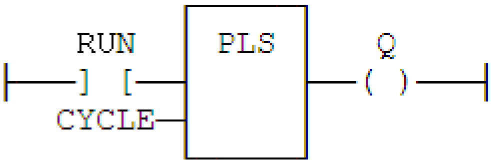
4.9.3.5 ST Language

```
(* MyPLS is a declared instance of PLS function block. *)
MyPLS (RUN, CYCLE);
Q := MyPLS.Q;
```

4.9.3.6 FBD Language



4.9.3.7 FFLD Language


RUN
PLS
Q
CYCLE

4.9.3.8 IL Language

```
(* MyPLS is a declared instance of PLS function block. *)
Op1: CAL MyPLS (RUN, CYCLE)
     FFLD  MyPLS.Q
     ST   Q
```

See Also

- TOF / TOFR
- TON
- TP / TPR

4.9.4 Sig_Gen

Function Block - Generator of pseudo-analogical Signal

4.9.4.1 Inputs

`RUN : BOOL` Enabling command

`PERIOD : TIME` Signal period

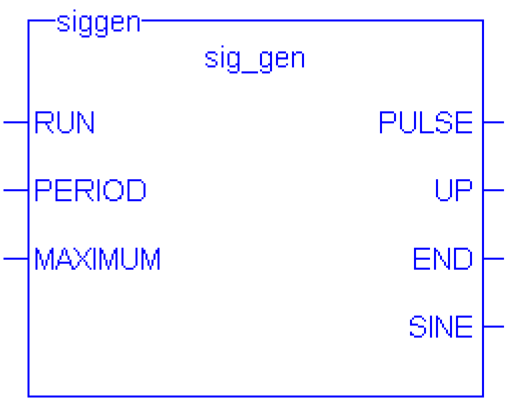
`MAXIMUM : DINT` Maximum growth during the signal period

4.9.4.2 Outputs

This FB generates signals of the four following types:

- `PULSE:` blinking at each period
- `UP    :` growing according max * period
- `END   :` pulse after max * period
- `SINE  :` sine curve

4.9.4.3 FFLD Language



4.9.5 TMD

Function Block - Down-counting stop watch.

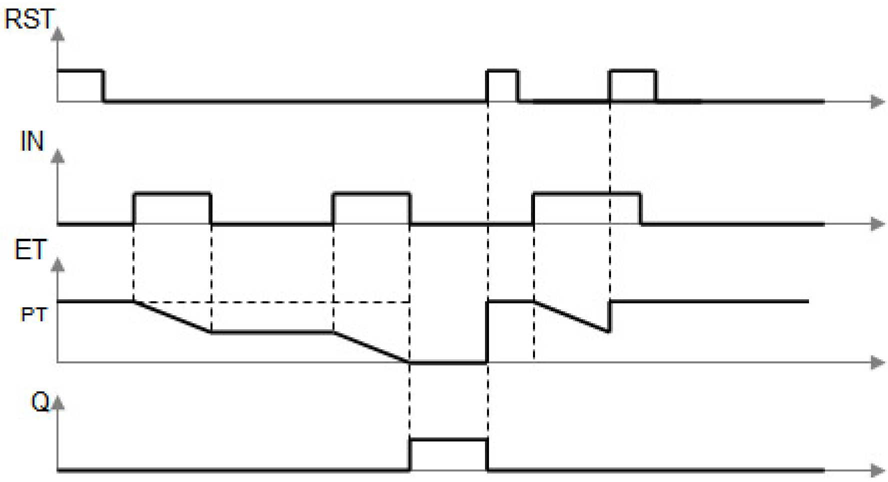
4.9.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	BOOL				The time counts when this input is TRUE.
RST	BOOL				Timer is reset to 0 (zero) when this input is TRUE.
PT	TIME				Programmed time.

4.9.5.2 Outputs

Input	Data Type	Range	Unit	Description
Q	BOOL			Timer elapsed output signal.
ET	TIME			Elapsed time.

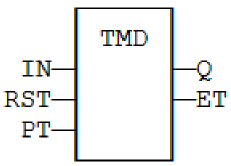
4.9.5.3 Time Diagram



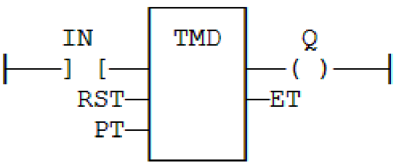
4.9.5.4 Remarks

- The timer counts up when the IN input is TRUE.
  - It stops when the programmed time is elapsed.
- The timer is reset when the RST input is TRUE.
  - It is not reset when IN is false.

4.9.5.5 FBD Language



4.9.5.6 FFLD Language



4.9.5.7 IL Language

```
(* MyTimer is a declared instance of TMD function block *)
Op1: CAL MyTimer (IN, RST, PT)
     FFLD: MyTimer.Q
     ST: Q
     FFLD: MyTimer.ET
     ST: ET
```

4.9.5.8 ST Language

```
(* MyTimer is a declared instance of TMD function block *)
MyTimer (IN, RST, PT);
Q := MyTimer.Q;
ET := MyTimer.ET;
```

See Also

"TMU / TMUsec" (→ p. 173)

4.9.6 TMU / TMUsec

PLCopen ✓

Function Block - Up-counting stop watch.

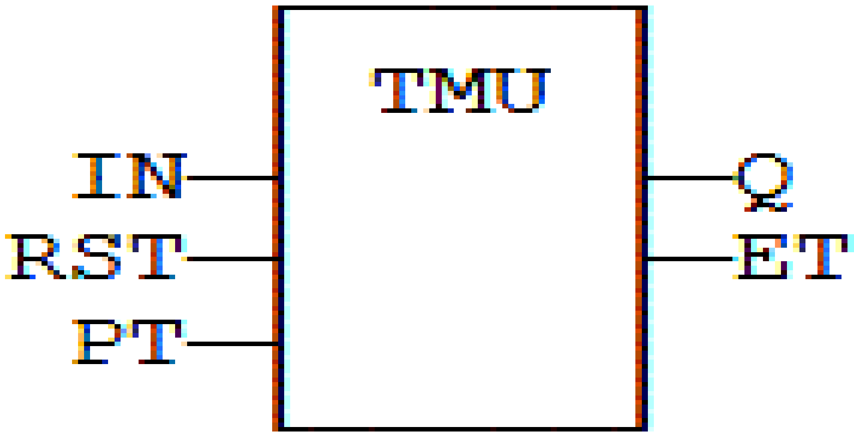
4.9.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	BOOL				The time counts when this input is TRUE.
RST	BOOL				Timer is reset to 0 (zero) when this input is TRUE.
PT	TIME				Programmed time. (TMU)
PTsec	UDINT				Programmed time. (TMUsec - seconds)

4.9.6.2 Outputs

Input	Data Type	Range	Unit	Description
Q	BOOL			Timer elapsed output signal.
ET	TIME			Elapsed time. (TMU)
ETsec	UDINT			Elapsed time. (TMU - seconds)

4.9.6.3 Time Diagram

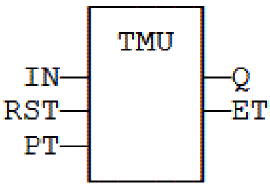


4.9.6.4 Remarks

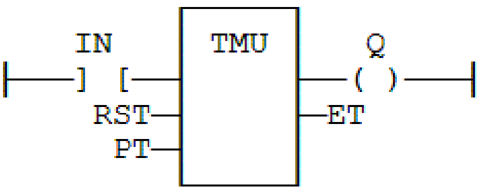
TMUsec is identical to TMU except that the parameter is a number of seconds.

- The timer counts up when the IN input is TRUE.
  - It stops when the programmed time is elapsed.
- The timer is reset when the RST input is TRUE.
  - It is not reset when IN is false.

4.9.6.5 FBD Language



4.9.6.6 FFLD Language



4.9.6.7 IL Language

```
(* MyTimer is a declared instance of TMU function block *)
Op1: CAL   MyTimer (IN, RST, PT)
     FFLD  MyTimer.Q
     ST    Q
     FFLD  MyTimer.ET
     ST    ET
```

4.9.6.8 ST Language

```
(* MyTimer is a declared instance of TMU function block *)
MyTimer (IN, RST, PT);
Q := MyTimer.Q;
ET := MyTimer.ET;
```

See Also

"TMD" (→ p. 171)

4.9.7 TOF / TOFR

PLCopen ✓

Function Block - Off timer.

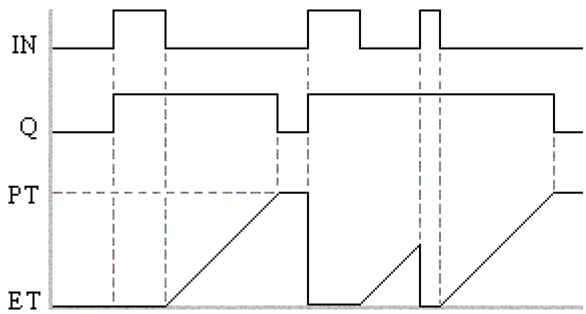
4.9.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	BOOL				Timer command.
PT	TIME				Programmed time.
RST	BOOL				Reset (TOFR only).

4.9.7.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL			Timer elapsed output signal.
ET	TIME			Elapsed time.

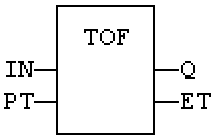
4.9.7.3 Time Diagram



4.9.7.4 Remarks

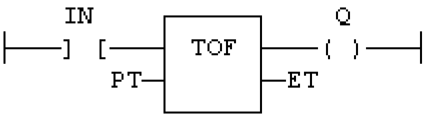
- TOFR is same as TOF but has an extra input for resetting the timer.
- The timer starts on a falling pulse of IN input.
  - It stops when the elapsed time is equal to the programmed time.
- A rising pulse of IN input resets the timer to 0 (zero).
- The output signal is set to TRUE when the IN input rises to TRUE.
  - It is reset to FALSE when the programmed time is elapsed.

4.9.7.5 FBD Language



4.9.7.6 FFLD Language

- In the FFLD language, the input rung is the IN command.
  - The output rung is the Q output. - CAN THIS BE USED INSTEAD?
  - The output rung is Q the output signal.



4.9.7.7 IL Language

```
(* MyTimer is a declared instance of TOF function block *)
Op1: CAL MyTimer (IN, PT)
     FFLD  MyTimer.Q
     ST  Q
     FFLD  MyTimer.ET
     ST  ET
```

4.9.7.8 ST Language

```
(* MyTimer is a declared instance of TOF function block *)
MyTimer (IN, PT);
Q := MyTimer.Q;
ET := MyTimer.ET;
```

See Also

- "BLINK" (➜ p. 166)
- "TON" (➜ p. 177)
- "TP / TPR" (➜ p. 178)

4.9.8 TON

PLCopen

Function Block - On timer.

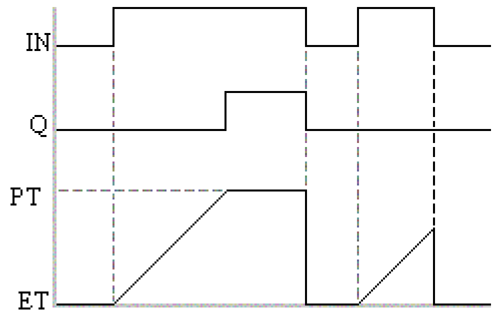
4.9.8.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	BOOL				Timer command.
PT	TIME				Programmed time.

4.9.8.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL			Timer elapsed output signal.
ET	TIME			Elapsed time.

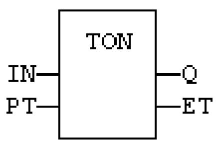
4.9.8.3 Time Diagram



4.9.8.4 Remarks

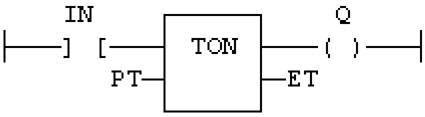
- The timer starts on a rising pulse of IN input.
  - It stops when the elapsed time is equal to the programmed time.
- A falling pulse of IN input resets the timer to 0 (zero).
- The output signal is set to TRUE when programmed time is elapsed.
  - It is reset to FALSE when the input command falls.

4.9.8.5 FBD Language



4.9.8.6 FFLD Language

- In the FFLD language, the input rung is the IN command.
  - The output rung is the Q output. - CAN THIS BE USED INSTEAD?
  - The output rung is Q the output signal.



4.9.8.7 IL Language

```
(* MyTimer is a declared instance of TON function block *)
Op1: CAL MyTimer (IN, PT)
     FFLD  MyTimer.Q
     ST  Q
     FFLD  MyTimer.ET
     ST  ET
```

4.9.8.8 ST Language

```
MyTimer is a declared instance of TON function block.
MyTimer (IN, PT);
Q := MyTimer.Q;
ET := MyTimer.ET;
```

See Also

- "BLINK" (→ p. 166)
- "TOF / TOFR" (→ p. 175)
- "TP / TPR" (→ p. 178)

4.9.9 TP / TPR

PLCopen

Function Block - Pulse timer.

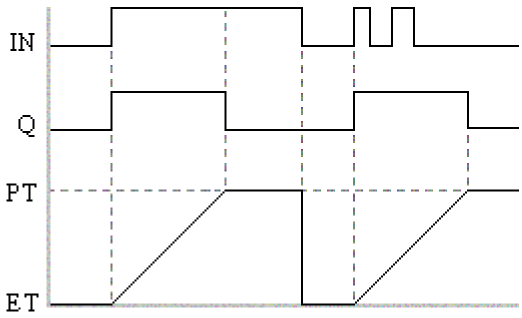
4.9.9.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	BOOL				Timer command.
PT	TIME				Programmed time.
RST	BOOL				Reset (TPR only).

4.9.9.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL			Timer elapsed output signal.
ET	TIME			Elapsed time.

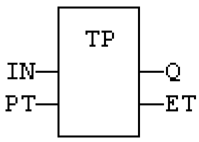
4.9.9.3 Time Diagram



4.9.9.4 Remarks

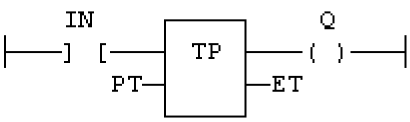
- TPR is same as TP but has an extra input for resetting the timer.
- The timer starts on a rising pulse of IN input.
  - It stops when the elapsed time is equal to the programmed time.
- A falling pulse of IN input resets the timer to 0 (zero) but only if the programmed time is elapsed.
- All pulses of IN while the timer is running are ignored.
- The output signal is set to TRUE while the timer is running.

4.9.9.5 FBD Language



4.9.9.6 FFLD Language

- In the FFLD language, the input rung is the IN command.
  - The output rung is the Q output. - CAN THIS BE USED INSTEAD?
  - The output rung is Q the output signal.



4.9.9.7 IL Language

```
(* MyTimer is a declared instance of TP function block *)
Op1: CAL MyTimer (IN, PT)
     FFLD  MyTimer.Q
     ST   Q
     FFLD  MyTimer.ET
     ST   ET
```

4.9.9.8 ST Language

```
(* MyTimer is a declared instance of TP function block *)
MyTimer (IN, PT);
Q := MyTimer.Q;
ET := MyTimer.ET;
```

See Also

- "BLINK" (➜ p. 166)
- "TOF / TOFR" (➜ p. 175)
- "TON" (➜ p. 177)

4.10 Mathematic Operations

PLCopen ✓

These are the standard functions that perform mathematic calculation:

Function	Description
"'s aslLN / LNL" (➜ p. 184)	Natural logarithm
abs / absL	Absolute value
"EXP / EXPL" (➜ p. 182) expt POW ** POWL	Power
LOG / LOGL	Logarithm
"ROOT" (➜ p. 186)	Root extraction
ScaleLin	Scaling - linear conversion
SQRT / SQRTL	Square root
trunc / truncL	Integer part

4.10.1 abs / absL

PLCopen ✓

Function - Returns the absolute value of the input.

4.10.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	REAL / LREAL				Any value.

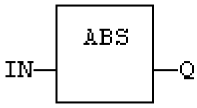
4.10.1.2 Outputs

Output	Data Type	Range	Unit	Description
Q	REAL / LREAL			Result: Absolute value of IN.

4.10.1.3 Remarks

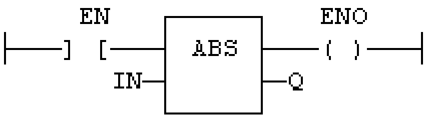
None

4.10.1.4 FBD Language



4.10.1.5 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.



4.10.1.6 IL Language

- In the IL language, the input must be loaded in the current result before calling the function.

```
Op1: LD   IN
     ABS
     ST   Q      (* Q is: ABS (IN) *)
```

4.10.1.7 ST Language

```
Q := ABS (IN);
```

See Also

- LOG / LOGL
- POW ** POWL
- SQRT / SQRTL
- trunc / truncL

4.10.2 expt

PLCopen ✓

Function - Calculates a power.

4.10.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	REAL				Real value.
EXP	DINT				Exponent.

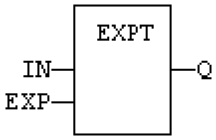
4.10.2.2 Outputs

Output	Data Type	Range	Unit	Description
Q	REAL			Result: IN at the EXP power

4.10.2.3 Remarks

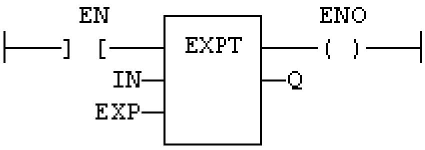
The exponent (second input of the function) must be the operand of the function.

4.10.2.4 FBD Language



4.10.2.5 FFLD Language

- In the FFLD language, the conversion is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.
  - The function is executed only if EN is TRUE.
  - ENO keeps the same value as EN.



4.10.2.6 IL Language

- In the IL language, the input must be loaded in the current result before calling the function.
-

```
Op1: LD   IN
     EXPT EXP
     ST   Q     (* Q is: (IN ** EXP) *)
```

4.10.2.7 ST Language

```
Q := EXPT (IN, EXP);
```

See Also

- abs / absL
- LOG / LOGL
- SQRT / SQRTL
- trunc / truncL

4.10.3 EXP / EXPL

Function - Calculates the natural exponential of the input.

4.10.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	REAL / LREAL				Real value.

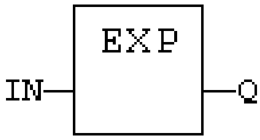
4.10.3.2 Outputs

Output	Data Type	Range	Unit	Description
Q	REAL / LREAL			Result: Natural exponential of IN.

4.10.3.3 Remarks

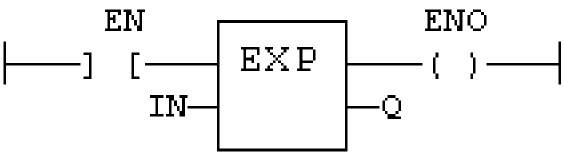
None

4.10.3.4 FBD Language



4.10.3.5 FFLD Language

- In the FFLD language, the conversion is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.
  - The function is executed only if EN is TRUE.
  - ENO has the same value as EN.



4.10.3.6 IL Language

- In the IL language, the first input must be loaded before the function call.

```
Op1: LD   IN
     EXP
     ST     Q     (* Q is: EXP (IN) *)
```

4.10.3.7 ST Language

```
Q := EXP (IN);
```

4.10.4 LOG / LOGL

Function - Calculates the logarithm (base 10) of the input.

4.10.4.1 Inputs

`IN : REAL/LREAL` Real value.

4.10.4.2 Outputs

`Q : REAL/LREAL` Result: logarithm (base 10) of IN.

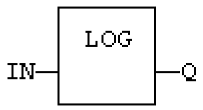
4.10.4.3 Remarks

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
    - The output rung (ENO) keeps the same value as the input rung.
- In the IL language, the first input must be loaded before the function call.

4.10.4.4 ST Language

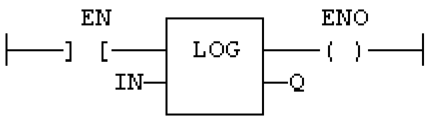
```
Q := LOG (IN);
```

4.10.4.5 FBD Language



4.10.4.6 FFLD Language

(* The function is executed only if EN is TRUE. *)
(* ENO keeps the same value as EN. *)



4.10.4.7 IL Language

Op1: LD IN
LOG
ST Q (* Q is: LOG (IN) *)

See Also

- abs / absL
- POW ** POWL
- SQRT / SQRTL
- trunc / truncL

4.10.5 's aslLN / LNL

PLCopen

Function - Calculates the natural logarithm of the input.

4.10.5.1 Inputs

`IN : REAL/LREAL` Real value.

4.10.5.2 Outputs

`Q : REAL/LREAL` Result: natural logarithm of IN.

4.10.5.3 Remarks

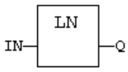
In FFLD language, the operation is executed only if the input rung (EN) is TRUE. The output rung (ENO) keeps the same value as the input rung.

In IL, the input must be loaded in the current result before calling the function.

4.10.5.4 ST Language

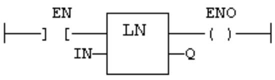
```
Q := LN (IN);
```

4.10.5.5 FBD Language



4.10.5.6 FFLD Language

(* The function is executed only if EN is TRUE. *)
(* ENO keeps the same value as EN. *)



4.10.5.7 IL Language

```
Op1: LD   IN
     LN
     ST     Q     (* Q is: LN (IN) *)
```

4.10.6 POW ** POWL

PLCopen

Function - Calculates a power.

4.10.6.1 Inputs

`IN : REAL/LREAL` Real value.
`EXP : REAL/LREAL` Exponent.

4.10.6.2 Outputs

`Q : REAL/LREAL` Result: IN at the 'EXP' power.

4.10.6.3 Remarks

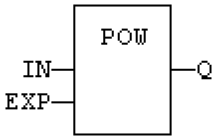
Alternatively, in ST language, the ****** operator can be used. In FFLD language, the operation is executed only if the input rung (EN) is TRUE. The output rung (ENO) keeps the same value as the input rung.

In IL, the input must be loaded in the current result before calling the function. The exponent (second input of the function) must be the operand of the function.

4.10.6.4 ST Language

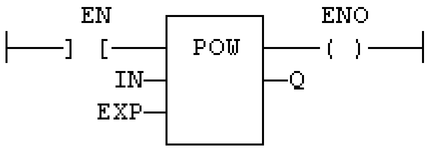
Q := POW (IN, EXP);
Q := IN ** EXP;

4.10.6.5 FBD Language



4.10.6.6 FFLD Language

(* The function is executed only if EN is TRUE. *)
(* ENO keeps the same value as EN. *)



4.10.6.7 IL Language

```
Op1: LD  IN
     POW EXP
     ST  Q    (* Q is: (IN ** EXP) *)
```

See Also

- abs / absL
- LOG / LOGL
- SQRT / SQRTL
- trunc / truncL

4.10.7 ROOT PLCopen

Function - Calculates the Nth root of the input.

4.10.7.1 Inputs

`IN : REAL` Real value.

`N : DINT` Root level.

4.10.7.2 Outputs

`Q : REAL` Result: Nth root of IN.

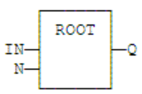
4.10.7.3 Remarks

- In FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.
- In the IL language, the input must be loaded in the current result before calling the function.

4.10.7.4 ST Language

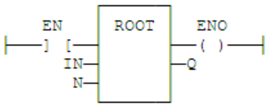
```
Q := ROOT (IN, N);
```

4.10.7.5 FBD Language



4.10.7.6 FFLD Language

(* The function is executed only if EN is TRUE *)
(* ENO keeps the same value as EN *)



4.10.7.7 IL Language

```
Op1: LD   IN
     ROOT  N
     ST    Q     (* Q is: ROOT (IN) *)
```

4.10.8 ScaleLin PLCopen

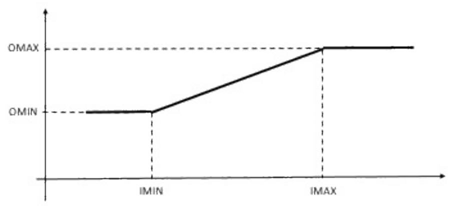
Function - Scaling - linear conversion.

4.10.8.1 Inputs

Inputs	Data Type	Description
IN	REAL	Real value.
IMIN	REAL	Minimum input value.
IMAX	REAL	Minimum input value.
OMIN	REAL	Minimum output value.
OMAX	REAL	Minimum output value.

4.10.8.2 Outputs

Output	Data Type	Description
Q	REAL	Result: OMIN + IN * (OMAX - OMIN) / (IMAX - IMIN).



4.10.8.3 Truth Table

Inputs	OUT
IMIN >= IMAX	= IN
IN < IMIN	= OMIN
IN > IMAX	= OMAX

Inputs	OUT
other	= OMIN + IN * (OMAX - OMIN) / (IMAX - IMIN)

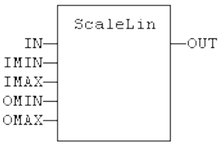
4.10.8.4 Remarks

- In FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.
- In IL, the input must be loaded in the current result before calling the function.

4.10.8.5 ST Language

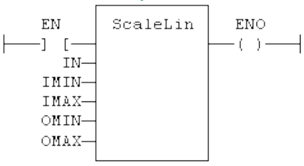
```
OUT := ScaleLin (IN, IMIN, IMAX, OMIN, OMAX);
```

4.10.8.6 FBD Language



4.10.8.7 FFLD Language

(* The function is executed only if EN is TRUE *)
(* ENO keeps the same value as EN *)



4.10.8.8 IL Language

```
Op1: LD       IN
     ScaleLin IMAX, IMIN, OMAX, OMIN
     ST       OUT
```

4.10.9 SQRT / SQRTL

PLCopen ✓

Function - Calculates the square root of the input.

4.10.9.1 Inputs

`IN : REAL/LREAL` Real value.

4.10.9.2 Outputs

`Q : REAL/LREAL` Result: square root of IN.

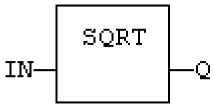
4.10.9.3 Remarks

- In FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.
- In the IL language, the first input must be loaded before the function call.

4.10.9.4 ST Language

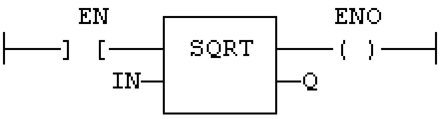
```
Q := SQRT (IN);
```

4.10.9.5 FBD Language



4.10.9.6 FFLD Language

(* The function is executed only if EN is TRUE.*)
(* ENO keeps the same value as EN. *)



4.10.9.7 IL Language

```
Op1: LD   IN
     SQRT
     ST   Q      (* Q is: SQRT (IN) *)
```

See Also

- "abs / absL" (➔ p. 180)
- "LOG / LOGL" (➔ p. 183)
- "POW ** POWL" (➔ p. 185)
- "trunc / truncL" (➔ p. 190)

4.10.10 trunc / truncL

Function - Truncates the decimal part of the input.

4.10.10.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	REAL / LREAL				Real value.

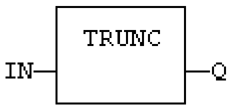
4.10.10.2 Outputs

Output	Data Type	Range	Unit	Description
Q	REAL / LREAL			Result: Integer part of N.

4.10.10.3 Remarks

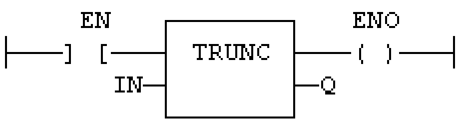
None

4.10.10.4 FBD Language



4.10.10.5 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.
  - The function is executed only if EN is TRUE.
  - ENO keeps the same value as EN.



4.10.10.6 IL Language

- In the IL language, the input must be loaded in the current result before calling the function.

```
Op1: LD   IN
     TRUNC
     ST   Q     (* Q is the integer part of IN *)
```

4.10.10.7 ST Language

```
Q := TRUNC (IN);
```

See Also

- "abs / absL" (➜ p. 180)
- "LOG / LOGL" (➜ p. 183)
- "POW ** POWL" (➜ p. 185)
- "SQRT / SQRTL" (➜ p. 188)

4.11 Trigonometric Functions

These are the standard functions for trigonometric calculation:

Function	Description
"acos / acosL" (➜ p. 191)	Arc-cosine
"asin / asinL" (➜ p. 192)	Arc-sine
"atan / atanL" (➜ p. 193)	Arc-tangent
"atan2 / atan2L" (➜ p. 194)	Arc-tangent of Y / X
"cos / cosL" (➜ p. 195)	Cosine
"sin / sinL" (➜ p. 197)	Sine
"tan / tanL" (➜ p. 198)	Tangent

See Also

"UseDegrees" (➜ p. 199)

4.11.1 acos / acosL

PLCopen ✓

Function - Calculate an arc-cosine.

4.11.1.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	REAL / LREAL				Real value.

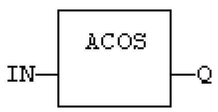
4.11.1.2 Outputs

Output	Data Type	Range	Unit	Default	Description
Q	REAL / LREAL				Result: arc-cosine of IN.

4.11.1.3 Remarks

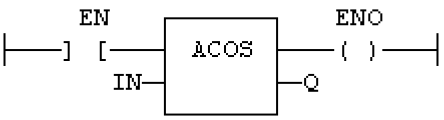
None

4.11.1.4 FBD Language



4.11.1.5 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.



4.11.1.6 IL Language

- In the IL language, the input must be loaded in the current result before calling the function.

```
Op1: LD   IN
     ACOS
     ST   Q     (* Q is: ACOS (IN) *)
```

4.11.1.7 ST Language

```
Q := ACOS (IN);
```

See Also

- asin / asinL
- atan / atanL
- atan2 / atan2L
- cos / cosL
- sin / sinL
- tan / tanL

4.11.2 asin / asinL

PLCopen

Function - Calculate an arc-sine.

4.11.2.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	REAL / LREAL				Real value.

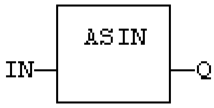
4.11.2.2 Outputs

Output	Data Type	Range	Unit	Description
Q	REAL / LREAL			Result: arc-sine of IN.

4.11.2.3 Remarks

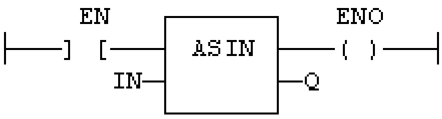
None

4.11.2.4 FBD Language



4.11.2.5 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.



4.11.2.6 IL Language

- In the IL language, the first input must be loaded before the function call.

```
Op1: LD  IN
     ASIN
     ST  Q     (* Q is: ASIN (IN) *)
```

4.11.2.7 ST Language

```
Q := ASIN (IN);
```

See Also

- "acos / acosL" (➔ p. 191)
- "atan / atanL" (➔ p. 193)
- "atan2 / atan2L" (➔ p. 194)
- "cos / cosL" (➔ p. 195)
- "sin / sinL" (➔ p. 197)
- "tan / tanL" (➔ p. 198)

4.11.3 atan / atanL

PLCopen

Function - Calculate an arc-tangent.

4.11.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	REAL / LREAL				Real value.

4.11.3.2 Outputs

Output	Data Type	Range	Unit	Description
Q	REAL / LREAL			Result: arc-tangent of IN.

4.11.3.3 Remarks

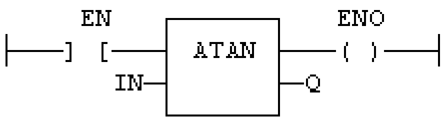
None

4.11.3.4 FBD Language

4.11.3.5 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.

(* The function is executed only if EN is TRUE. *)
(* ENO keeps the same value as EN. *)



4.11.3.6 IL Language

- In the IL language, the first input must be loaded before the function call.

```
Op1: LD   IN
     ATAN
     ST   Q     (* Q is: ATAN (IN) *)
```

4.11.3.7 ST Language

```
Q := ATAN (IN);
```

See Also

- acos / acosL
- asin / asinL
- atan2 / atan2L
- cos / cosL
- sin / sinL
- tan / tanL

4.11.4 atan2 / atan2L

Function - Calculate arc-tangent of Y/X.

4.11.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
X	REAL / LREAL				Real value.
Y	REAL / LREAL				Real value.

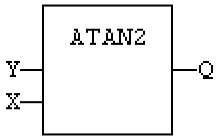
4.11.4.2 Outputs

Output	Data Type	Range	Unit	Description
Q	REAL / LREAL			Result: arc-tangent of X / Y.

4.11.4.3 Remarks

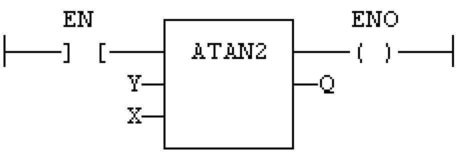
None

4.11.4.4 FBD Language



4.11.4.5 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.



4.11.4.6 IL Language

- In the IL language, the first input must be loaded before the function call.

```
Op1: LD   Y
     ATAN2 X
     ST   Q     (* Q is: ATAN2 (Y / X) *)
```

4.11.4.7 ST Language

is there an ST Language for this?

```

```

See Also

- acos / acosL
- asin / asinL
- 4.11.3 atan / atanL
- cos / cosL
- sin / sinL
- tan / tanL

4.11.5 cos / cosL

PLCopen ✓

Function - Calculate a cosine.

4.11.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	REAL / LREAL				Real value.

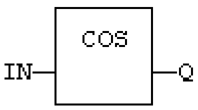
4.11.5.2 Outputs

Output	Data Type	Range	Unit	Description
Q	REAL / LREAL			Result: cosine of IN.

4.11.5.3 Remarks

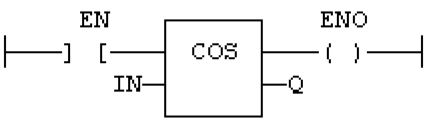
None

4.11.5.4 FBD Language



4.11.5.5 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.
  - The function is executed only if EN is TRUE.



4.11.5.6 IL Language

- In the IL language, the input must be loaded in the current result before calling the function.

```
Op1: LD   IN
     COS
     ST   Q     (* Q is: COS (IN)  *)
```

4.11.5.7 ST Language

```
Q := COS (IN);
```

See Also

- acos / acosL
- asin / asinL
- atan / atanL
- atan2 / atan2L
- sin / sinL
- tan / tanL

4.11.6 sin / sinL

PLCopen ✓

Function - Calculate a sine.

4.11.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	REAL / LREAL				Real value.

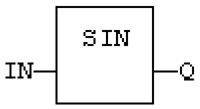
4.11.6.2 Outputs

Output	Data Type	Range	Unit	Description
Q	REAL / LREAL			Result: Sine of IN.

4.11.6.3 Remarks

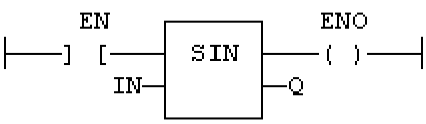
None

4.11.6.4 FBD Language



4.11.6.5 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.
- The function is executed only if EN is TRUE.
- ENO keeps the same value as EN.



4.11.6.6 IL Language

- In the IL language, the input must be loaded in the current result before calling the function.

```
Op1: LD  IN
     SIN
     ST  Q     (* Q is: SIN (IN) *)
```

4.11.6.7 ST Language

```
Q := SIN (IN);
```

See Also

- acos / acosL
- asin / asinL
- atan / atanL
- atan2 / atan2L
- cos / cosL
- tan / tanL

4.11.7 tan / tanL

Function - Calculate a tangent.

4.11.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	REAL / LREAL				Real value.

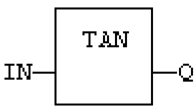
4.11.7.2 Outputs

Output	Data Type	Range	Unit	Description
Q	REAL / LREAL			Result: Tangent of IN.

4.11.7.3 Remarks

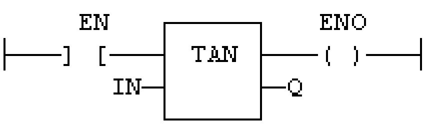
None

4.11.7.4 FBD Language



4.11.7.5 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.
- The function is executed only if EN is TRUE.
- ENO keeps the same value as EN.



4.11.7.6 IL Language

- In the IL language, the input must be loaded in the current result before calling the function.

```
Op1: LD   IN
     TAN
     ST   Q     (* Q is: TAN (IN) *)
```

4.11.7.7 ST Language

```
Q := TAN (IN);
```

See Also

- ACOS
- asin / asinL
- ATAN
- ATAN2
- COS
- SIN

4.11.8 UseDegrees

Function - Sets the unit for angles in all trigonometric functions.

4.11.8.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	BOOL				• If TRUE, turn all trigonometric functions to use degrees. • If FALSE, turn all trigonometric functions to use radians (default).

4.11.8.2 Outputs

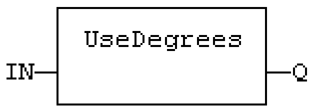
Output	Data Type	Range	Unit	Description
Q	BOOL			TRUE if functions use degrees before the call.

4.11.8.3 Remarks

This function sets the working unit for these functions:

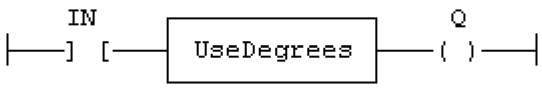
Function	Description
"acos / acosL" (➔ p. 191)	Arc-cosine
"asin / asinL" (➔ p. 192)	Arc-sine
"atan / atanL" (➔ p. 193)	Arc-tangent
"atan2 / atan2L" (➔ p. 194)	Arc-tangent of Y / X
"cos / cosL" (➔ p. 195)	Cosine
"sin / sinL" (➔ p. 197)	Sine
"tan / tanL" (➔ p. 198)	Tangent

4.11.8.4 FBD Language



4.11.8.5 FFLD Language

- The first input is the rung.
- The rung is the output.



4.11.8.6 IL Language

```
Op1: LD   IN
     UseDegrees
     ST   Q
```

4.11.8.7 ST Language

```
Q := UseDegrees (IN);
```

4.12 String Operations

4.12.1 Standard Operators

These are the standard operators and functions that manage character strings:

Functions and Operators	Operator / Function
"Addition +" (➜ p. 89)	Concatenation of strings.
"ArrayToString / ArrayToStringU" (➜ p. 201)	Copies elements of an SINT array to a STRING.
"ascii" (➜ p. 202)	Get the ASCII code of a character within a string.
"ATOH" (➜ p. 203)	Converts string to integer using hexadecimal basis.
"char" (➜ p. 204)	Build a single character string.
"concat" (➜ p. 205)	Concatenation of strings.
"CRC16" (➜ p. 206)	CRC16 calculation.
"delete" (➜ p. 207)	Delete characters in a string.
"FIND" (➜ p. 208)	Find characters in a string.
"HTOA" (➜ p. 210)	Converts integer to string using hexadecimal basis.
"INSERT" (➜ p. 211)	Insert characters in a string.
"LEFT" (➜ p. 212)	Extract a part of a string on the left.
"MID" (➜ p. 214)	Extract a part of a string.
"MLEN" (➜ p. 215)	Get string length.
"REPLACE" (➜ p. 216)	Replace characters in a string.

Functions and Operators	Operator / Function
"RIGHT" (➜ p. 218)	Extract a part of a string on the right.
"StringToArray / StringToArrayU" (➜ p. 222)	Copies characters of a STRING to an SINT array.

4.12.2 Manage String Tables

These functions are for managing string tables as resources:

Function	Description
"LoadString" (➜ p. 213)	Load a string from the active string table.
"StringTable" (➜ p. 220)	Select the active string table resource.

4.12.3 ArrayToString / ArrayToStringU

PLCopen ✓

Function - Copy an array of SINT to a STRING.

4.12.3.1 Inputs

Input	Data Type	Range	Unit	Default	Description
COUNT	DINT				Number of characters to be copied.
DST	STRING				Destination STRING.
SRC	SINT				Source array of SINT small integers. USINT for ArrayToStringU.

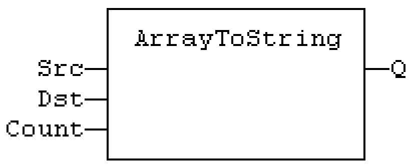
4.12.3.2 Outputs

Output	Data Type	Range	Unit	Description
Q	DINT			Number of characters copied.

4.12.3.3 Remarks

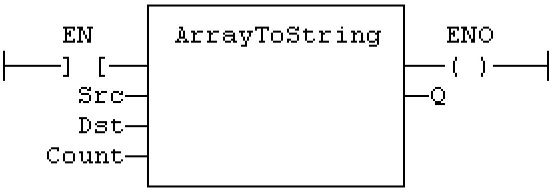
- This function copies the COUNT first elements of the SRC array to the characters of the DST string.
- The function checks the maximum size of the destination string and adjusts the COUNT number if necessary.

4.12.3.4 FBD Language



4.12.3.5 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.



4.12.3.6 IL Language

Not available.

4.12.3.7 ST Language

```
Q := ArrayToString (SRC, DST, COUNT);
```

See Also

StringToArray / StringToArrayU

4.12.4 ascii

PLCopen

Function - Get the ASCII code of a character within a string.

4.12.4.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	STRING				Input string.
POS	DINT				Position of the character within the string. The first valid position is 1.

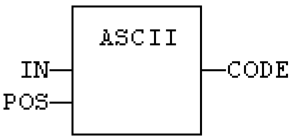
4.12.4.2 Outputs

Output	Data Type	Range	Unit	Description
CODE	DINT			ASCII code of the selected character or 0 (zero) if position is invalid.

4.12.4.3 Remarks

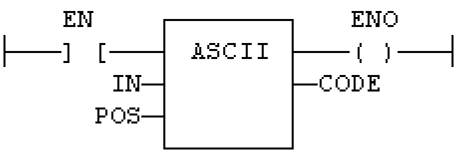
None

4.12.4.4 FBD Language



4.12.4.5 FFLD Language

- In the FFLD language, the input rung (EN) enables the operation.
  - The output rung (ENO) keeps the same value as the input rung.



4.12.4.6 IL Language

- In the IL language, the first parameter (IN) must be loaded in the current result before calling the function.
  - The other input is the operand of the function.

```
Op1: LD       IN
     AND_MASK MSK
     ST       CODE
```

4.12.4.7 ST Language

```
CODE := ASCII (IN, POS);
```

See Also

char

4.12.5 ATOH

Function - Converts string to integer using hexadecimal basis.

4.12.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	STRING				String representing an integer in hexadecimal format.

4.12.5.2 Outputs

Output	Data Type	Range	Unit	Description
Q	DINT			Integer represented by the string.

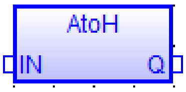
4.12.5.3 Truth Table

IN	Q
' '	0
'12'	18
'a0'	160
'A0zzz'	160

4.12.5.4 Remarks

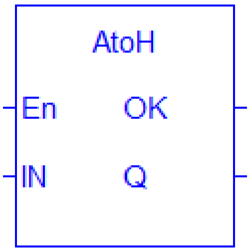
- The function is case insensitive.
- The result is 0 (zero) for an empty string.
- The conversion stops before the first invalid character.

4.12.5.5 FBD Language



4.12.5.6 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.



4.12.5.7 IL Language

- In the IL language, the first input must be loaded before the function call.

```
Op1: LD   IN
     ATOH
     ST   Q
```

4.12.5.8 ST Language

```
Q := ATOH (IN);
```

See Also

HTOA

4.12.6 char

PLCopen ✓

Function - Builds a single character string.

4.12.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
CODE	DINT				ASCII code of the specified character.

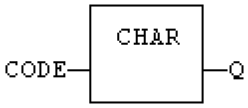
4.12.6.2 Outputs

Output	Data Type	Range	Unit	Description
Q	STRING			STRING containing only the specified character.

4.12.6.3 Remarks

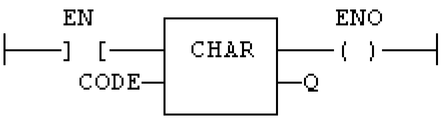
None

4.12.6.4 FBD Language



4.12.6.5 FFLD Language

- In the FFLD language, the input rung (EN) enables the operation.
  - The output rung (ENO) keeps the same value as the input rung.



4.12.6.6 IL Language

- In the IL language, the input parameter (CODE) must be loaded in the current result before calling the function.

```
Op1: LD   CODE
     CHAR
     ST   Q
```

4.12.6.7 ST Language

```
Q := CHAR (CODE);
```

See Also

ascii

4.12.7 concat

PLCopen ✓

Function - Concatenate strings.

4.12.7.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN_1	STRING				Any string variable or constant expression.
IN_N	STRING				Any string variable or constant expression.

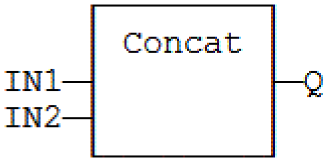
4.12.7.2 Outputs

Output	Data Type	Range	Unit	Description
Q	STRING			Concatenation of all inputs.

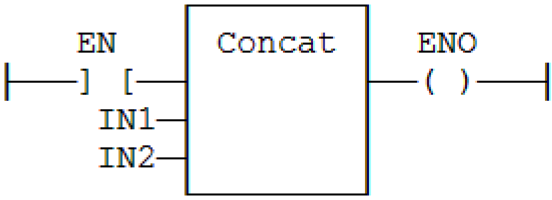
4.12.7.3 Remarks

- In the FBD or FFLD languages, the block can have up to 16 inputs.
- In the IL or ST languages, the function accepts a variable number of inputs (at least 2).
- Use the **+** operator to concatenate strings.

4.12.7.4 FBD Language



4.12.7.5 FFLD Language



4.12.7.6 IL Language

```
Op1: FFLD      'AB'
     CONCAT 'CD', 'E'
     ST   Q    (* Q is now 'ABCDE' *)
```

4.12.7.7 ST Language

```
Q := CONCAT ('AB', 'CD', 'E');
(* now Q is 'ABCDE' *)
```

4.12.8 CRC16

PLCopen ✓

Function - calculates a CRC16 on the characters of a string.

4.12.8.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	STRING				Character string.

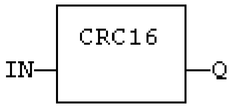
4.12.8.2 Outputs

Output	Data Type	Range	Unit	Description
Q	INT			CRC16 calculated on all the characters of the string.

4.12.8.3 Remarks

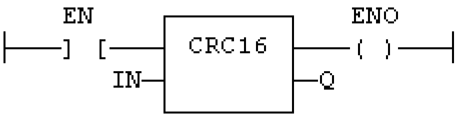
- The function calculates a Modbus CRC16, initialized at 16#FFFF value.

4.12.8.4 FBD Language



4.12.8.5 FFLD Language

- In the FFLD language, the input rung (EN) enables the operation.
  - The output rung (ENO) keeps the same value as the input rung.
  - The function is executed only if EN is TRUE.



4.12.8.6 IL Language

- In the IL language, the input must be loaded in the current result before calling the function.

```
Op1: LD    IN
     CRC16
     ST    Q
```

4.12.8.7 ST Language

```
Q := CRC16 (IN);
```

4.12.9 delete

PLCopen

Function - Delete characters in a string.

4.12.9.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	STRING				Character string.
NBC	DINT				Number of characters to be deleted.
POS	DINT				Position of the first deleted character. The first character position is 1.

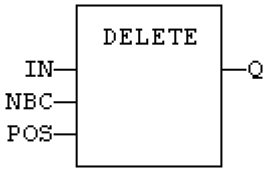
4.12.9.2 Outputs

Output	Data Type	Range	Unit	Description
Q	STRING			Modified string.

4.12.9.3 Remarks

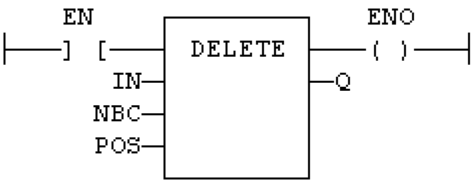
- The first valid character position is 1.

4.12.9.4 FBD Language



4.12.9.5 FFLD Language

- In the FFLD language, the conversion is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.
  - The function is executed only if EN is TRUE.



4.12.9.6 IL Language

- In the IL language, the first input (the string) must be loaded in the current result before calling the function.
  - Other arguments are operands of the function, separated by comas.

```
Op1: LD     IN
     DELETE NBC, POS
     ST     Q
```

4.12.9.7 ST Language

```
Q := DELETE (IN, NBC, POS);
```

See Also

- Addition +
- FIND
- INSERT
- LEFT
- MID
- MLEN
- REPLACE
- RIGHT

4.12.10 FIND

Function - Find position of characters in a string.

4.12.10.1 Inputs

IN	**STRING**	Character string.
STR	**STRING**	Specific characters to search for within the STRING.

4.12.10.2 Outputs

POS	**DINT**	Position of the first character of STR in IN, or 0 if not found.

4.12.10.3 Remarks

The first valid character position is 1. A return value of 0 means that the STR substring has not been found. The return value can be used with other string functions such as MID and RIGHT. The search is case sensitive.

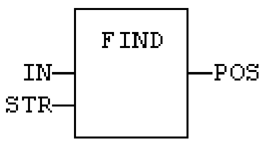
In FFLD language, the operation is executed only if the input rung (EN) is TRUE. The output rung (ENO) keeps the same value as the input rung.

In IL, the first input (the string) must be loaded in the current result before calling the function. The second argument is the operand of the function.

4.12.10.4 ST Language

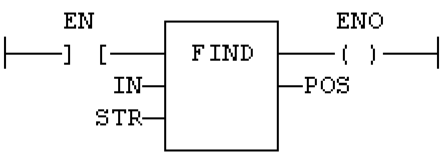
POS := FIND (IN, STR);

4.12.10.5 FBD Language



4.12.10.6 FFLD Language

(* The function is executed only if EN is TRUE. *)
(* ENO keeps the same value as EN. *)



4.12.10.7 IL Language

```
Op1: LD    IN
     FIND  STR
     ST    POS
```

See Also

- Addition +
- delete
- INSERT
- LEFT
- MID
- MLEN

- REPLACE
- RIGHT

4.12.11 HTOA

PLCopen

Function - Converts integer to string using hexadecimal basis.

4.12.11.1 Inputs

Input	Data Type	Range	Unit	Default	Description
IN	DINT				Integer value.

4.12.11.2 Outputs

Output	Data Type	Range	Unit	Description
Q	STRING			String representing an integer in hexadecimal format.

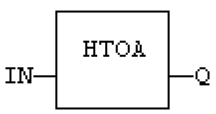
4.12.11.3 Truth Table

IN	Q
0	'0'
18	'12'
160	'A0'

4.12.11.4 Remarks

None

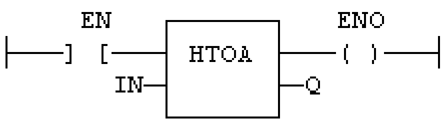
4.12.11.5 FBD Language



4.12.11.6 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.

- The function is executed only if EN is TRUE.
- ENO keeps the same value as EN.



4.12.11.7 IL Language

- In the IL language, the input must be loaded in the current result before calling the function.

```
Op1: LD   IN
     HTOA
     ST   Q
```

4.12.11.8 ST Language

```
Q := HTOA (IN);
```

See Also

ATOH

4.12.12 INSERT

Function - Insert characters in a string.

4.12.12.1 Inputs

`IN : STRING` Character string.
`STR : STRING` String containing characters to be inserted.
`POS : DINT` Position of the first inserted character (first character position is 1).

4.12.12.2 Outputs

`Q : STRING` Modified string.

4.12.12.3 Remarks

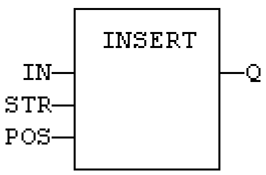
The first valid character position is 1. In FFLD language, the operation is executed only if the input rung (EN) is TRUE. The output rung (ENO) keeps the same value as the input rung.

In IL, the first input (the string) must be loaded in the current result before calling the function. Other arguments are operands of the function, separated by comas.

4.12.12.4 ST Language

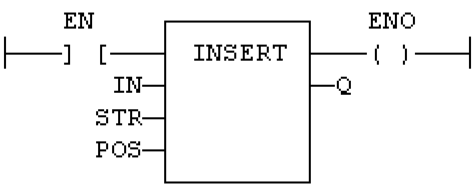
Q := INSERT (IN, STR, POS);

4.12.12.5 FBD Language



4.12.12.6 FFLD Language

(* The function is executed only if EN is TRUE. *)
(* ENO keeps the same value as EN. *)



4.12.12.7 IL Language

```
Op1: LD    IN
     INSERT STR, POS
     ST    Q
```

See Also

- Addition +
- delete
- FIND
- LEFT
- MID
- MLEN
- REPLACE
- RIGHT

4.12.13 LEFT

Function - Extract characters of a string on the left.

4.12.13.1 Inputs

`IN : STRING` Character string.
`NBC : DINT` Number of characters to extract.

4.12.13.2 Outputs

`Q : STRING` String containing the first NBC characters of IN.

4.12.13.3 Remarks

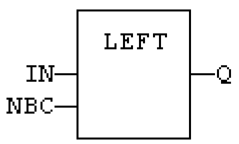
In FFLD language, the operation is executed only if the input rung (EN) is TRUE. The output rung (ENO) keeps the same value as the input rung.

In IL, the first input (the string) must be loaded in the current result before calling the function. The second argument is the operand of the function.

4.12.13.4 ST Language

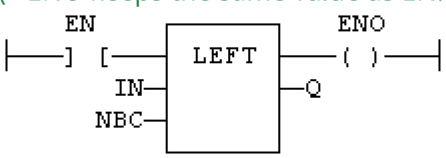
Q := LEFT (IN, NBC);

4.12.13.5 FBD Language



4.12.13.6 FFLD Language

(* The function is executed only if EN is TRUE. *)
(* ENO keeps the same value as EN. *)



4.12.13.7 IL Language

Op1: LD IN
LEFT NBC
ST Q

See Also

- Addition +
- delete
- FIND
- INSERT
- MID
- MLEN
- REPLACE
- RIGHT

4.12.14 LoadString

Function - Load a string from the active string table.

4.12.14.1 Inputs

`ID: DINT` ID of the string as declared in the string table.

4.12.14.2 Outputs

`Q : STRING` Loaded string or empty string in case of error.

4.12.14.3 Remarks

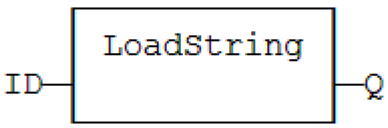
This function loads a string from the active string table and stores it in a STRING variable. The StringTable() function is used for selecting the active string table.

The ID input (the string item identifier) is an identifier such as declared within the string table resource. You don't need to "define" this identifier again - the system does it for you.

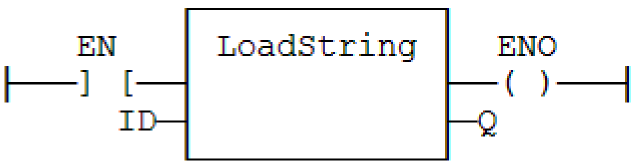
4.12.14.4 ST Language

Q := LoadString (ID);

4.12.14.5 FBD Language



4.12.14.6 FFLD Language



4.12.14.7 IL Language

```
Op1: LD      ID
    LoadString
    ST      Q
```

See Also

StringTable String tables

4.12.15 MID

Function - Extract characters of a string at any position.

4.12.15.1 Inputs

`IN : STRING` Character string.
`NBC : DINT` Number of characters to extract.
`POS : DINT` Position of the first character to extract (first character of IN is at position 1).

4.12.15.2 Outputs

`Q : STRING` String containing the first NBC characters of IN.

4.12.15.3 Remarks

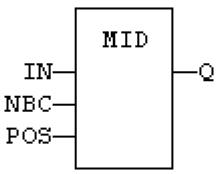
The first valid position is 1. In FFLD language, the operation is executed only if the input rung (EN) is TRUE. The output rung (ENO) keeps the same value as the input rung.

In IL, the first input (the string) must be loaded in the current result before calling the function. Other arguments are operands of the function, separated by comas.

4.12.15.4 ST Language

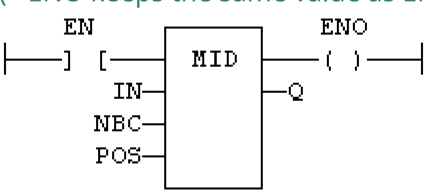
```
Q := MID (IN, NBC, POS);
```

4.12.15.5 FBD Language



4.12.15.6 FFLD Language

(* The function is executed only if EN is TRUE. *)
(* ENO keeps the same value as EN. *)



4.12.15.7 IL Language

```
Op1: LD  IN
    MID  NBC, POS
    ST  Q
```

See Also

- Addition +
- delete
- FIND
- INSERT
- LEFT
- MLEN
- REPLACE
- RIGHT

4.12.16 MLEN

Function - Get the number of characters in a string.

4.12.16.1 Inputs

`IN : STRING` Character string.

4.12.16.2 Outputs

`NBC : DINT` Number of characters currently in the string. 0 if string is empty.

4.12.16.3 Remarks

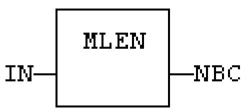
In FFLD language, the operation is executed only if the input rung (EN) is TRUE. The output rung (ENO) keeps the same value as the input rung.

In IL, the input must be loaded in the current result before calling the function.

4.12.16.4 ST Language

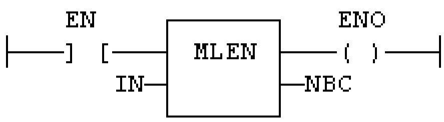
NBC := MLEN (IN);

4.12.16.5 FBD Language



4.12.16.6 FFLD Language

(* The function is executed only if EN is TRUE. *)
(* ENO keeps the same value as EN. *)



4.12.16.7 IL Language

```
Op1: LD   IN
     MLEN
     ST   NBC
```

See Also

+ DELETE INSERT FIND REPLACE LEFT RIGHT MID

- Addition +
- delete
- FIND
- INSERT
- LEFT
- MID
- REPLACE
- RIGHT

4.12.17 REPLACE

Function - Replace characters in a string.

4.12.17.1 Inputs

Inputs	Data Type	Description
IN	STRING	Character string.
STR	STRING	String containing the characters to be inserted in place of NDEL removed characters.
NDEL	DINT	Number of characters to be deleted before insertion of STR.
POS	DINT	Position where characters are replaced (first character position is 1).

4.12.17.2 Outputs

Output	Data Type	Description
Q	STRING	Modified string.

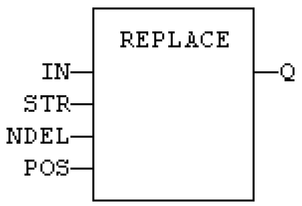
4.12.17.3 Remarks

- The first valid character position is 1.
- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.
- In the IL language, the first input (the string) must be loaded in the current result before calling the function.
  - Other arguments are operands of the function separated by comas.

4.12.17.4 ST Language

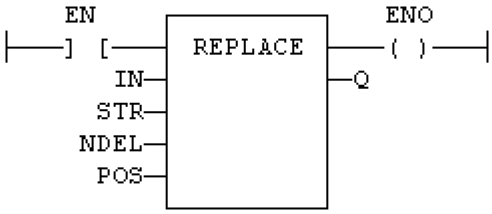
```
Q := REPLACE (IN, STR, NDEL, POS);
```

4.12.17.5 FBD Language



4.12.17.6 FFLD Language

(* The function is executed only if EN is TRUE *)
(* ENO keeps the same value as EN *)



4.12.17.7 IL Language

```
Op1: LD      IN
     REPLACE STR, NDEL, POS
     ST      Q
```

See Also

- "Addition +" (➔ p. 89)
- "delete" (➔ p. 207)
- "FIND" (➔ p. 208)
- "INSERT" (➔ p. 211)
- "LEFT" (➔ p. 212)
- "MID" (➔ p. 214)
- "MLEN" (➔ p. 215)
- "RIGHT" (➔ p. 218)

4.12.18 RIGHT

Function - Extract characters of a string on the right.

4.12.18.1 Inputs

`IN : STRING` Character string.

`NBC : DINT` Number of characters to extract .

4.12.18.2 Outputs

`Q : STRING` String containing the last NBC characters of IN.

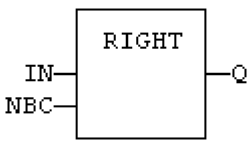
4.12.18.3 Remarks

- In FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.
- In the IL language, the first input (the string) must be loaded in the current result before calling the function.
  - The second input is the operand of the function.

4.12.18.4 ST Language

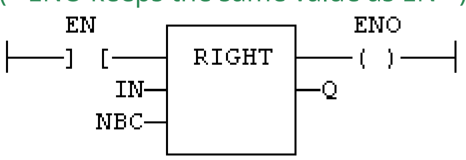
```
Q := RIGHT (IN, NBC);
```

4.12.18.5 FBD Language



4.12.18.6 FFLD Language

(* The function is executed only if EN is TRUE *)
(* ENO keeps the same value as EN *)



4.12.18.7 IL Language

```
Op1: LD     IN
     RIGHT  NBC
     ST     Q
```

See Also

- "delete" (→ p. 207)
- "FIND" (→ p. 208)
- "INSERT" (→ p. 211)

- "LEFT" (➔ p. 212)
- "MID" (➔ p. 214)
- "MLEN" (➔ p. 215)
- "REPLACE" (➔ p. 216)

4.12.19 StringTable

PLCopen

Function - Selects the active string table.

4.12.19.1 Inputs

Input	Data Type	Range	Unit	Default	Description
TABLE	STRING				Name of the Sting Table resource. Must be a constant.
COL	STRING				Name of the column in the table. Must be a constant.

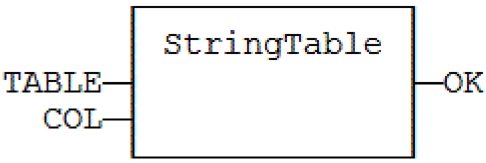
4.12.19.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL			TRUE if OK.

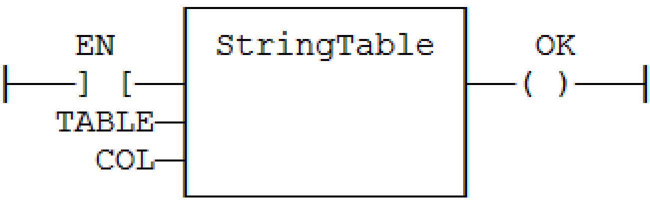
4.12.19.3 Remarks

- This function selects a column of a valid String Table resource to become the active string table.
  - The "LoadString" (→ p. 213) function always refers to the active string table.
- Arguments must:
  - be constant string expressions.
  - fit to a declared string table and a valid column name within this table.
- If there is only one string table with only one column defined in the project, you do not need to call this function.
  - It is the default string table.

4.12.19.4 FBD Language



4.12.19.5 FFLD Language



4.12.19.6 IL Language

```
Op1: LD
     'MyTable'
     StringTable 'First Column'
```

```
    ST
OK
```

4.12.19.7 ST Language

```
OK := StringTable ('MyTable', 'FirstColumn");
```

See Also

- "LoadString" (→ p. 213)
- "String Table Resources" (→ p. 221)

4.12.19.8 String Table Resources

String tables are resources (embedded configuration data) edited with Workbench.

- A string table is a list of items identified by a name and referring to one or more character strings.
- String tables are typically used for defining static texts to be used in the application.
- These functions can be used for getting access to string tables in the programs:
  - "StringTable" (→ p. 220): selects the active string table.
  - "LoadString" (→ p. 213): Load a string from the active table.

- Each string table may contain several columns of texts for each item, and thus ease the localization of application, simply by defining a column for each language.
  - This way, the language can be selected dynamically at runtime by specifying the active language (as a column) in the StringTable() function.

The name entered in the string table as an ID is automatically declared for the compiler.

- The name:
  - Can directly be passed to the LoadString() function without re-declaring it.
  - Must conform to IEC standard naming rules.

You could do the same by declaring an array of `STRING` variables and enter some initial values for all items in the array.

- String tables provide significant advantages compared to arrays:
  - The editor provides a comfortable view of multiple columns at editing.
  - String tables are loaded in the application code and does not require any further RAM memory unlike declared arrays.
  - The string table editor automatically declares readable IDs for any string item to be used in programs instead of working with hard-coded index values.

TIP

If the text is too long for the `STRING` variable when used at runtime, it is truncated.
Use special $ sequences in strings to specify non printable characters, according to the IEC standard:

Code	Meaning
$$	A "$" character.
$'	A Single quote.
$T	A tab stop (ASCII code 9).
$R	A carriage return character (ASCII code 13).

Code	Meaning
$L	A line feed character (ASCII code 10).
$N	Carriage return plus line feed characters (ASCII codes 13 and 10).
$P	A page break character (ASCII code 12).
$xx	Any character (xx is the ASCII code expressed on two hexadecimal digits.

4.12.20 StringToArray / StringToArrayU

PLCopen

Function - Copies the characters of a STRING to an array of SINT.

4.12.20.1 Inputs

Input	Data Type	Range	Unit	Default	Description
DST	SINT				Destination array of SINT small integers. USINT for StringToArrayU.
SRC	STRING				Source STRING.

4.12.20.2 Outputs

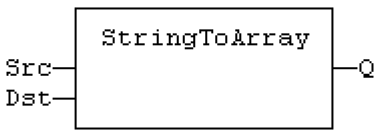
Output	Data Type	Range	Unit	Description
Q	DINT			Number of characters copied.

4.12.20.3 Remarks

This function:

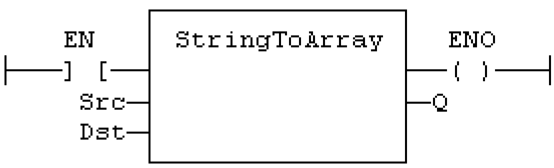
- Copies the characters of the SRC string to the first characters of the DST array.
- Checks the maximum size destination arrays and reduces the number of copied characters if necessary.

4.12.20.4 FBD Language



4.12.20.5 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.



4.12.20.6 IL Language

- In the IL language, the input must be loaded in the current result before calling the function.

```
Op1: LD
SRC
StringToArray DST
ST            Q
```

4.12.20.7 ST Language

```
Q := StringToArray (SRC, DST);
```

See Also

"ArrayToString / ArrayToStringU" (➜ p. 201)

5 PLC Advanced Libraries

These are the standard blocks that perform advanced operations.

5.1 Analog Signal Processing

Block	Description
average / averageL	Calculates the average of signal samples.
CurveLin	Linear interpolation on a curve.
derivate	Computes the derivative of a signal with respect to time.
hyster	Hysteresis detection.
integral	Calculates the integral of a signal with respect to time.
lim_alrm	Detects high and low limits of a signal with hysteresis.
PID	PID loop.
RAMP	Ramp signal.
"rand" (➜ p. 262)	Returns a pseudo-random integer value between 0 (zero) and (base - 1).
SigPlay	Play an analog signal from a resource.
SigScale	Get a point from a signal resource.
SurfLin	Linear interpolation on a surface.

5.2 Alarm Management

Block	Description
"Alarm_A" (➜ p. 225)	Alarm with automatic reset.
Alarm_M	Alarm with manual reset.
lim_alrm	Detects high and low limits of a signal with hysteresis.

5.3 Data Collections and Serialization

Block	Description
FIFO	Manages a first in / first out list.
LIFO	Last in / first out stack.
"SerializeIn" (➜ p. 293)	Extract the value of a variable from a binary frame.

Block	Description
"SerializeOut" (➔ p. 296)	Copy the value of a variable to a binary frame.
stackint	Manages a stack of DINT integers.

5.4 Data Log

Block	Description
"LogFileCSV" (➔ p. 255)	Create a log file in CSV format for a list of variables.

5.5 Special Operations

Block	Description
ApplyRecipeColumn	Apply the values of a column from a recipe file.
CycleStop	Sets the application in cycle stepping mode.
EnableEvents	Enable or disable the production of events for binding (runtime to runtime variable exchange).
FatalStop	Breaks the cycle and stop with fatal error.
GetSysInfo	Get system information.
printf	Trace messages.
SigID	Get the ID of a signal resource.
VLID	Get the ID of an embedded list of variables.

5.6 Communication

- AS-interface Functions
- TCP/IP Function Blocks
- UDP Functions for PxMM & Simulator

5.7 Others

- File Management
- File Tools Function Blocks
- Real Time Clock Management Functions

5.8 Alarm_A

PLCopen

Function Block - Alarm with automatic reset.

5.8.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ACK	BOOL				Acknowledge command.
IN	BOOL				Process signal.

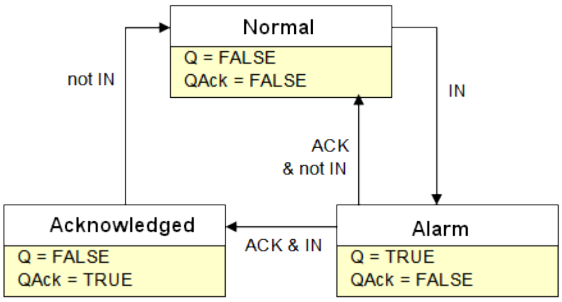
5.8.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL			TRUE if alarm is active.
QACK	BOOL			TRUE if alarm is acknowledged.

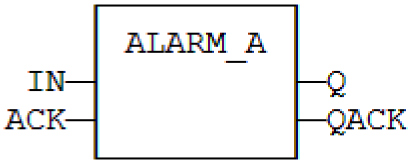
5.8.3 Remarks

- Combine this block with the lim_alrm block for managing analog alarms.

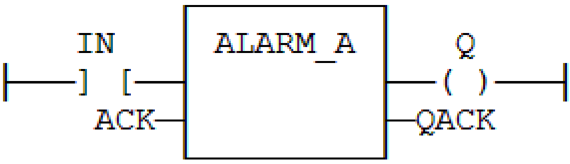
5.8.3.1 Sequence



5.8.4 FBD Language



5.8.5 FFLD Language



5.8.6 IL Language

```
(* MyALARM is declared as an instance of ALARM_A function block *)
Op1: CAL
MyALARM (IN, ACK)
FFLD  MyALARM.Q
ST  Q
```

```
FFLD  MyALARM.QACK
ST
QACK
```

5.8.7 ST Language

```
(* MyALARM is declared as an instance of ALARM_A function block *)
MyALARM (IN, ACK, RST);
Q := MyALARM.Q;
QACK := MyALARM.QACK;
```

See Also

Alarm_M

5.9 Alarm_M

PLCopen

Function Block - Alarm with manual reset.

5.9.1 Inputs

Input	Data Type	Range	Unit	Default	Description
ACK	BOOL				Acknowledge command.
IN	BOOL				Process signal.
RST	BOOL				Reset command.

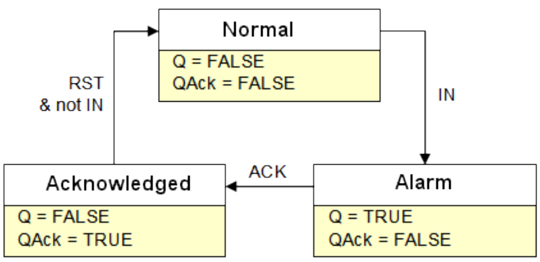
5.9.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL			TRUE if alarm is active.
QACK	BOOL			TRUE if alarm is acknowledged.

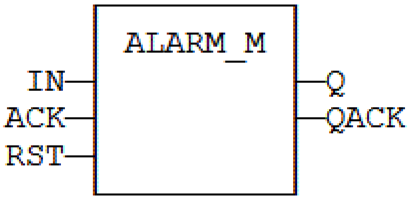
5.9.3 Remarks

- Combine this block with the lim_alrm block for managing analog alarms.

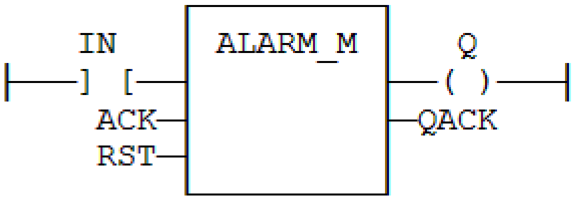
5.9.3.1 Sequence



5.9.4 FBD Language



5.9.5 FFLD Language



5.9.6 IL Language

```
(* MyALARM is declared as an instance of ALARM_M function block *)
Op1: CAL
MyALARM (IN, ACK, RST)
FFLD  MyALARM.Q
ST  Q
FFLD  MyALARM.QACK
ST
QACK
```

5.9.7 ST Language

```
(* MyALARM is declared as an instance of ALARM_M function block *)
MyALARM (IN, ACK, RST);
Q := MyALARM.Q;
QACK := MyALARM.QACK;
```

See Also

Alarm_A

5.10 ApplyRecipeColumn

PLCopen

Function - Apply the values of a column from a recipe file.

5.10.1 Inputs

Input	Data Type	Range	Unit	Default	Description
FILE	STRING				Path name of the recipe file (.CSV or .RCP). Must be a constant value.
COL	DINT				Index of the column in the recipe (0 (zero) based).

5.10.2 Outputs

Output	Data Type	Range	Unit	Description
OK	BOOL			• TRUE if OK. • FALSE if parameters are invalid.

5.10.3 Remarks

- The **FILE** input is a constant string expression specifying the path name of a valid .CSV or .RCP file.
  - If no path is specified, the file is assumed to be located in the project folder.
  - CSV files are created using Excel or Notepad.
  - RCP files are created using an external recipe editor.
- In CSV files, the first line must contain column headers, and is ignored during compiling.
  - There is one variable per line.
  - The first column contains the symbol of the variable.
    - Other columns are values.
- If a cell is empty, it is assumed to be the same value as the previous (left side) cell.
  - If it is the first cell of a row, it is assumed to be null (0 or FALSE or empty string).

Example of CSV File

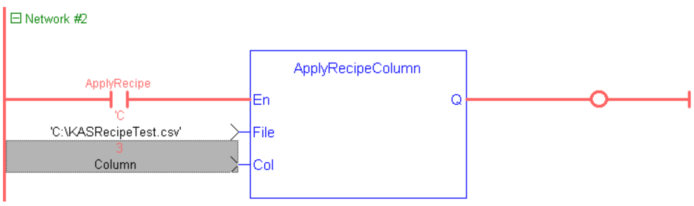
Example of CSV file with five variables and five set of values

```
comment lines here
TravelSpeed;100;200;300;400;500
MasterAbsPos;0;45;90;135;180
MasterDeltaPos;0;90;180;270;360
MachineSpeed;50;100;150;200;250
MachineState;0;0;1;1;2
```

NOTE

For your CSV file to be valid, ensure the data are separated with **semicolons** (and not commas).

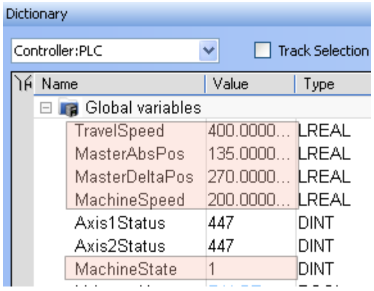
Usage in a FFLD program where column 3 is selected



Column 3 corresponds to column E in the Excel sheet because this parameter is 0 based

	A	B	C	D	E	F
1	comment lines here					
2	TravelSpeed	100	200	300	400	500
3	MasterAbsPos	0	45	90	135	180
4	MasterDeltaPos	0	90	180	270	360
5	MachineSpeed	50	100	150	200	250
6	MachineState	0	0	1	1	2

Result displayed in the Dictionary when the application is running



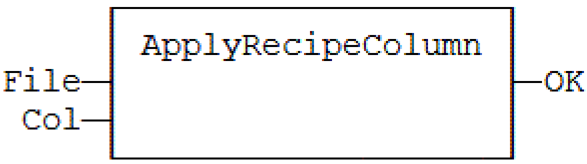
Example of RCP File

```
@COLNAME=Col3 Col4
@SIZECOL1=100
@SIZECOL2=100
@SIZECOL3=100
@SIZECOL4=100
bCommand
tPerio
bFast
Blink1
test_var
bOut
@EXPANDED=Blink1
```

IMPORTANT

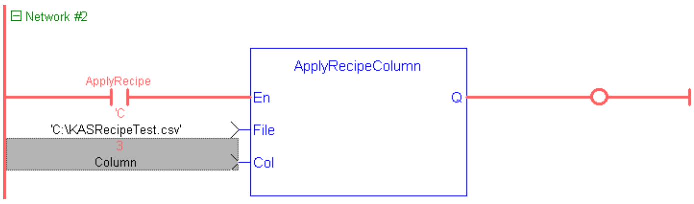
Recipe files are read at compiling time and are embedded into the downloaded application code. This implies that a modification performed in the recipe file after downloading is not taken into account by the application.

5.10.4 FBD Language



5.10.5 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
    - The output rung is the result of the function.
- The function is executed only if **ApplyRecipe** is TRUE.



5.10.6 IL Language

```
Op1: LD
'MyFile.rcp'
ApplyRecipeColumn COL
ST
OK
```

5.10.7 ST Language

```
OK := ApplyRecipeColumn ('MyFile.rcp', COL);
```

5.11 AS-interface Functions

These functions enable special operation on AS-i networks:

Function	Description
ASiReadPI	Read actual parameters of an AS-i slave.
ASiReadPP	Read permanent parameters of an AS-i slave.
ASiSendParam	Send parameters to an AS-i slave.
ASiStorePI	Store actual parameters as permanent parameters.
ASiWritePP	Write permanent parameters of an AS-i slave.

IMPORTANT

AS-i networking may be not available on some targets.
See the OEM instructions for more information.

5.11.1 Interface

```
Params := ASiReadPP (Master, Slave);
bOK := ASiWritePP (Master, Slave, Params);
bOK := ASiSendParam (Master, Slave, Params);
Params := ASiReadPI (Master, Slave);
bOK := ASiStorePI (Master);
```

5.11.2 Arguments

```
Master : DINT Index of the AS-i master (1..N) such as shown in
configuration.
Slave : DINT  Address of the AS-i slave (1..32 / 33..63).
```

```
Params : DINT Value of AS-i parameters.
bOK : BOOL    TRUE if successful.
```

5.12 average / averageL

PLCopen

Function Block - Calculates the average of signal samples.

5.12.1 Inputs

Input	Data Type	Range	Unit	Default	Description
RUN	BOOL				Enabling command.
XIN	REAL				Input signal.
N	DINT				Number of samples stored for average calculation. Cannot exceed 128.

5.12.2 Outputs

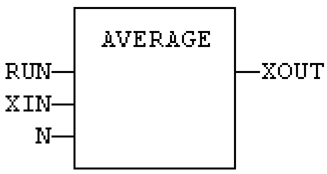
Output	Data Type	Range	Unit	Description
XOUT	REAL			Average of the stored samples (*).

(*) averageL has LREAL arguments.

5.12.3 Remarks

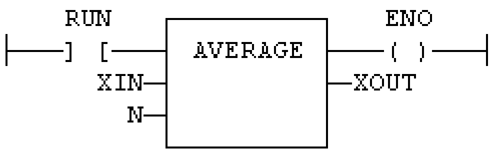
- Average is calculated according to the number of stored samples.
  - This can be less than N when the block is enabled.
    - The "N" input (or the number of samples) is taken into account **only** when the RUN input is FALSE.
  - By default, the number of samples is 128.
- The "RUN" must be reset after a change in the number of samples.
  - Cycle the RUN input when you first call this function; this clears the default.

5.12.4 FBD Language



5.12.5 FFLD Language

- In the FFLD language, the input rung is the RUN command.
  - The output rung keeps the state of the input rung.
  - ENO has the same value as RUN.



5.12.6 IL Language

```
(* MyAve is a declared instance of AVERAGE function block *)
Op1: CAL MyAve (RUN, XIN, N)
     FFLD  MyAve.XOUT
     ST   XOUT
```

5.12.7 ST Language

```
(* MyAve is a declared instance of AVERAGE function block. *)
MyAve (RUN, XIN, N);
XOUT := MyAve.XOUT;
```

See Also

- derivate
- hyster
- integral
- lim_alrm
- stackint

5.13 CurveLin

PLCopen

Function Block - Linear interpolation on a curve.

5.13.1 Inputs

Input	Data Type	Range	Unit	Default	Description
X	REAL				X coordinate of the point to be interpolated.
XAxis	REAL[ ]				X coordinates of the known points of the X axis.
YVal	REAL[ ]				Y coordinate of the points defined on the X axis.

5.13.2 Outputs

Output	Data Type	Range	Unit	Description
ERR	DINT			• Error code if failed. • 0 (zero) if OK.
OK	BOOL			TRUE if successful.
Y	REAL			Interpolated Y value corresponding to the X input.

5.13.3 Remarks

- This function performs linear interpolation in between a list of points defined in the XAxis single dimension array.
  - The output Y value is an interpolation of the Y values of the two rounding points defined in the X axis.
  - Y values of defined points are passed in the YVal single dimension array.
- Values in XAxis must be sorted from the smallest to the biggest.
  - There must be at least two points defined in the X axis.
    - YVal and XAxis input arrays must have the same dimension.
- In case the X input is **less than** the smallest defined X point:
  - The Y output takes the first value defined in YVal.
  - An error is reported.
- In case the X input is **greater than** the biggest defined X point:
  - The Y output takes the last value defined in YVal.
  - An error is reported.

The ERR output gives the cause of the error if the function fails:

Error Code	Meaning
0	OK
1	Invalid dimension of input arrays
2	Invalid points for the X axis
4	X is out of the defined X axis

5.14 derivate

PLCopen ✓

Function Block - Computes the derivative of a signal with respect to time.

5.14.1 Inputs

Input	Data Type	Range	Unit	Default	Description
RUN	BOOL	TRUE, FALSE			Run command: • TRUE=derivate. • FALSE=hold.
XIN	REAL				Input signal.
CYCLE	TIME				Sampling period. Must not be less than the target cycle timing.

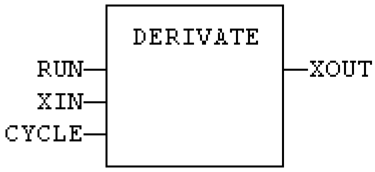
5.14.2 Outputs

Output	Data Type	Range	Unit	Description
XOUT	REAL			Output signal.

5.14.3 Remarks

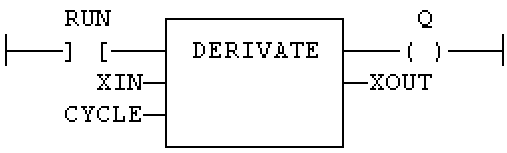
- The time unit is seconds.
- The output signal has the units of the input signal divided by seconds.
- The **derivate** block samples the input signal at a maximum rate of 1 millisecond.

5.14.4 FBD Language



5.14.5 FFLD Language

- In the FFLD language, the input rung is the RUN command.
  - The output rung keeps the state of the input rung.
  - ENO has the same state as RUN.



5.14.6 IL Language

```
(* MyDerv is a declared instance of DERIVATE function block *)
Op1: CAL MyDerv (RUN, XIN, CYCLE)
FFLD  MyDerv.XOUT
ST   XOUT
```

5.14.7 ST Language

```
(* MyDerv is a declared instance of DERIVATE function block. *)
MyDerv (RUN, XIN, CYCLE);
XOUT := MyDerv.XOUT;
```

See Also

- average / averageL
- hyster
- integral
- lim_alrm
- stackint

5.15 EnableEvents

PLCopen ✓

Function - Enable or disable the production of events for binding (runtime to runtime variable exchange).

5.15.1 Inputs

Input	Data Type	Range	Unit	Default	Description
EN	BOOL				• TRUE to enable events. • FALSE to disable events.

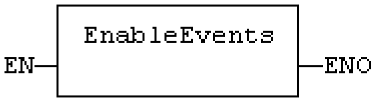
5.15.2 Outputs

Output	Data Type	Range	Unit	Description
ENO	BOOL			Echo of EN input.

5.15.3 Remarks

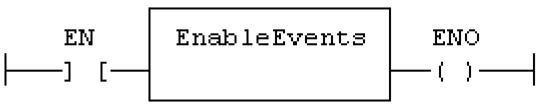
- Production is enabled when the application starts.
- The first production is operated after the first cycle.
- To disable events since the beginning, you must call `EnableEvents (FALSE)` in the very first cycle.

5.15.4 FBD Language



5.15.5 FFLD Language

- In FFLD language, the input rung (EN) enables the event production.
  - The output rung keeps the state of the input rung.
  - Events are enables if EN is TRUE.
  - ENO has the same value as EN.



5.15.6 IL Language

- In the IL language, the first input must be loaded before the function call.

```
Op1: LD   EN
     EnableEvents
     ST   ENO
```

5.15.7 ST Language

```
ENO := EnableEvents (EN);
```

See Also

Alarm_A

5.16 FIFO

PLCopen

Function Block - Manages a first in / first out list.

5.16.1 Inputs

Inputs	Data Type	Range	Unit	Default	Description
@Tail	ANY				Value of the oldest pushed value - *updated after call!*
Buf[]	ANY				Array for storing values.
IN	ANY				Value to be pushed.
POP	BOOL				Pop a new value (on rising edge).
PUSH	BOOL				Push a new value (on rising edge).
RST	BOOL				Reset the list.

5.16.2 Outputs

Outputs	Data Type	Range	Unit	Description
EMPTY	BOOL			TRUE if the list is empty.
OFLO	BOOL			TRUE if overflow on a PUSH command.
Count	DINT			Number of values in the list.
pRead	DINT			Index in the buffer of the oldest pushed value.
pWrite	DINT			Index in the buffer of the next push position.

5.16.3 Remarks

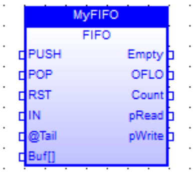
- IN, @Tail and Buf[] must have the same data type.
  - It cannot be STRING.

- The @Tail argument specifies a variable filled with the oldest push value after the block is called.
- Values are stored in the Buf[] array.
  - Data is arranged as a roll over buffer and is never shifted or reset.
  - Only read and write pointers and pushed values are updated.
  - The maximum size of the list is the dimension of the array.

- The first time an instance of the FIFO function block is called, that instance stores which array is passed to BUF[].

- If a later call to the same instance passes a different array for the BUF[] argument, the call is considered invalid and no action is performed.
- In this instance, the EMPTY output returns TRUE.

5.16.4 FBD Language



5.16.5 FFLD Language

- In the FFLD language, the input rung is the PUSH input.
  - The output rung is the EMPTY output.

5.16.6 IL Language

```
(* MyFIFO is a declared instance of FIFO function block *)
Op1: CAL MyFIFO (PUSH, POP, RST, IN, @Tail , BUFF[])
FFLD  MyFIFO.EMPTY
ST   EMPTY
FFLD  MyFIFO.OFLO
ST   OFLO
FFLD  MyFIFO.COUNT
ST   COUNT
FFLD  MyFIFO.PREAD
ST   PREAD
FFLD  MyFIFO.PWRITE
ST   PWRITE
```

5.16.7 ST Language

```
(* MyFIFO is a declared instance of FIFO function block: *)
MyFIFO (PUSH, POP, RST, IN, @Tail , BUFFER);
EMPTY := MyFIFO.EMPTY;
OFLO := MyFIFO.OFLO;
COUNT := MyFIFO.COUNT;
PREAD := MyFIFO.PREAD;
PWRITE := MyFIFO.PWRITE;
```

See Also

LIFO

5.17 File Management

File Management functions provide the ability to:

- Read machine recipes or other machine operational data into the KAS program from the SD card, USB flash drive, or a shared directory.
- Read cam tables into the program from the SD card, USB flash drive, or a shared directory.
- Store machine operational data in internal PxMM or PCMM2G flash memory (retrievable through the web server), the SD card, USB flash drive, or a shared directory.

NOTE

A shared directory connection is setup through the web server.

TIP

- Functions to parse out information from a file using a string format can be found in "String Operations" (➔ p. 200).
- If the file is in a .CSV format, these functions can be used: "LogFileCSV" (➔ p. 255), "ApplyRecipeColumn" (➔ p. 228).
- You can create, store, and retrieve recipes and other data using either:
  - the AKI Terminals. For more information see the KVB manual.
  - an external bus connection to the PxMM or PCMM2G with a supported fieldbus (e.g., UDP or HTTP).

These function blocks enable sequential read / write operations in disk files:

Function Block	Use
FileClose	Close an open file.
FileCopy	Copy a file.
FileDelete	Remove a file.
FileEOF	Test if the end of the file is reached in a file that is open for reading.
FileExists	Test if a file exists.
FileOpenA	Create or open a file in append mode.
FileOpenR	Open a file for reading.
FileOpenW	Create or reset a file and open it for writing.
FileReadBinData	Read binary data from a file.
FileReadLine	Read a string value from a text file.
FileRename	Rename a file.
FileSeek	Set the current position of a file.
FileSize	Get the size of a file.
FileWriteBinData	Write binary data to a file.
FileWriteLine	Write a string value to a text file.

These functions handle mounting of SD cards:

Name	Use
"SD_ISREADY" (➜ p. 242)	Check that the SD card is ready for read/write
"SD_MOUNT" (➜ p. 241)	Mount an SD card
"SD_UNMOUNT" (➜ p. 241)	Unmount an SD card

Each file is identified in the application by a unique handle manipulated as a DINT value.

- The file handles are allocated by the target system.
- Handles are returned by the Open function blocks and used by all other function blocks for identifying the file.

Related Function Blocks

`LogFileCSV` log values of variables to a CSV file

IMPORTANT

- Files are opened and closed directly by the Operating System of the target.
  - Opening some files can be dangerous for system safety and integrity.
  - **The number of open files (from FileOpenA, FileOpenR, and FileOpenW) is limited by the resources available on the target system.**
- Ensure that each file successfully opened using FileOpenA, FileOpenR, and FileOpenW.
  - The FileOpenW has a corresponding FileClose to close the file.
  - Closing the file will release the file ID, making it available for operations on other files.

NOTE

- Opening a file with FileOpenA, FileOpenR, and FileOpenW can be unsuccessful (invalid path or file name, too many open files.)
  - Your application must check the file ID for a NULL value.
  - If the file ID is NULL (zero), then file read or write operations will fail.
- File management may be unavailable on some targets.
- Memory on the SD card is available in addition to the existing flash memory.
- Valid paths for storing files depend on the target implementation.
- Error messages are logged in the Controller log section of KAS Runtime where there is a failure in any related function block.
- Using the KAS Simulator, all path names are ignored, and files are stored in a reserved directory. Only the file name passed to the Open functions is taken into account.
- AKD PDMM / PCMM files are big endian.
  - PCMM2G files are little endian.

TIP

Review the "File Path Conventions" (➜ p. 243) to understand hardware-based functional differences.

5.17.1 SD Card Access

Files may be written to and read from an SD card. This is typically used for storing a firmware image for Recovery Mode.

Use an SD card on the controller:

1. Verify the SD card is inserted.
2. Mount the card using "SD_MOUNT" (➜ p. 241).
3. Verify the card is accessible using "SD_ISREADY" (➜ p. 242) before performing a read or write action.
4. Unmount the card, using "SD_UNMOUNT" (➜ p. 241) after performing read/write actions.

NOTE

SD Card is not supported by PCMM2G.

TIP

Recommended: Stop all motion before using SD_MOUNT and SD_UNMOUNT.

5.17.2 SD Card Mounting Functions

Function - These functions handle mounting of an SD card:

Name	Use
"SD_MOUNT" (→ p. 241)	Mount an SD card.
"SD_UNMOUNT" (→ p. 241)	Unmount an SD card.
"SD_ISREADY" (→ p. 242)	Verify the SD card is ready for read/write.

5.17.2.1 SD_MOUNT

PLCopen ✓

Mount the SD Card.

TIP

Recommended: Stop all motion before using SD_MOUNT.

Device	Action	Return Value	Example
AKD PDMM	Mount the SD Card.	If the mount is successful, the return value is TRUE. If the mount is not successful, the return value is FALSE.	`OK := SD_MOUNT();` **OK : BOOL** — TRUE if mounting the SD Card is successful.
PCMM2G	SD Card is not supported by PCMM2G. This does not perform any action.	It always returns FALSE.	
Simulator	This does not perform any action.	It always returns TRUE.	

5.17.2.2 SD_UNMOUNT

PLCopen ✓

Unmount the SD Card.

TIP

Recommended: Stop all motion before using SD_UNMOUNT.

Device	Action	Return Value	Example
AKD PDMM	Unmount the SD Card.	If the unmount is successful, the return value is TRUE. If the unmount is not successful, the return value is FALSE.	`OK := SD_UNMOUNT();` **OK : BOOL** TRUE if unmounting the SD Card is successful.
PCMM2G	SD Card is not supported by PCMM2G. This does not perform any action.	It always returns FALSE.	
Simulator	This does not perform any action.	It always returns TRUE.	

5.17.2.3 SD_ISREADY

PLCopen ✓

Device	Action	Return Value	Example
AKD PDM M	Verify the SD Card is mounted in the AKD PDMM.	If the SD Card is mounted, the return value is TRUE. If the SD Card is not mounted, the return value is FALSE.	`OK := SD_ISREADY();` **OK : BOOL** TRUE if the SD Card is mounted.
PCMM2G	SD Card is not supported by PCMM2G. This does not perform any action.	It always returns FALSE.	

Device	Action	Return Value	Example
Simulator	Verify the **SDCard** folder exists here: C:\Users\[user's name]\AppData\Local\Kollmorgen\KAS\Sinope Simulator\Application\userdata**SDCard**	If the SDCard folder exists, the return value is TRUE. If the SDCard folder does not exist, the return value is FALSE.	`OK := SD_ISREADY();` **OK : BOOL** — TRUE if the SDCard folder exists.

5.17.3 File Path Conventions

Depending on the system used, paths to file locations may be defined as either:

- **absolute** (`C://dir1/file1`)
- **relative paths** (`/dir1/file1`)

Not all systems handle all options.

The paths vary depending upon the system.

System	Absolute Paths	Relative Paths	Handling of Directories
AKD PDMM **PCMM** **PCMM2G**	✗	✓	There is no support for creating directories on the controller. Any path provided to the function blocks (e.g., `file1`) is appended to the default user data folder. User Data Folders • PCMM & AKD PDMM: `/mount/flash/userdata/` • PCMM2G: `/home/kas/kas/userdata/`
Simulator	✓	✓	When a relative path is provided to the function blocks, the path is appended to the default user data folder: `<User Directory>/Kollmorgen/Kollmorgen Automation Suite/SinopeSimulator/Application/userdata/`

See Also

- "File Name Warning and Limitations" (➔ p. 243)
- "SD Card Path Conventions" (➔ p. 245)
- "Shared Directory Path Conventions" (➔ p. 244)
- "USB Flash Drive Path Conventions" (➔ p. 245)

5.17.3.1 File Name Warning and Limitations

- File names in the controller's flash storage are case-sensitive.
- The SD card or USB flash drive (FAT16 or FAT32) are NOT case-sensitive.

Storage	File System	Case-Sensitive
Embedded flash: AKD PDMM / PCMM/ PCMM2G	FFS3 (POSIX-like)	Yes
SD card / USB flash drive: AKD PDMM / PCMM	FAT16 or FAT32	No

Example

- Two files (`MyFile.txt` and `myfile.txt`) can exist in the same directory of the controller's flash.
  - They **cannot** exist in the same directory on the controller's SD card.
- If you copy two files (via backup operation or function) with the same name but different upper/lower case letters, from the controller's flash to the SD card or USB flash drive, one of the files is lost.

IMPORTANT

Use unique file names to prevent conflicts and to keep the application compatible across all platforms.
Do not rely on case-sensitive file names.

See Also

- "SD Card Path Conventions" (→ p. 245)
- "Shared Directory Path Conventions" (→ p. 244)
- "USB Flash Drive Path Conventions" (→ p. 245)

5.17.3.2 Shared Directory Path Conventions

The AKD PDMM, PCMM, and PCMM2G support access to a shared directory on a remote computer.

To access files in a shared directory from the AKD PDMM, PCMM, and PCMM2G use `/mount/shared` at the beginning of the path, before the shared directory's relative path and file name:

```
/mount/shared/directory/filename
```

Valid Paths	Notes
/mount/shared mount/shared \mount\shared mount\shared	• The path is not case sensitive. • The /MOUNT/SHARED, MOUNT/SHARED/, etc. are also valid.

Example 1

Opening the file `example.txt` from a shared directory on a remote computer.

```
fileID := Inst_FileOpenA(TRUE,'/mount/shared/example.txt');
```

Example 2

Opening the file `myfiles/example.txt` from a shared directory on a remote computer.

```
fileID := Inst_FileOpenA(TRUE,'/mount/shared/myfiles/example.txt');
```

See Also

- "File Name Warning and Limitations" (➔ p. 243)
- "SD Card Path Conventions" (➔ p. 245)
- "USB Flash Drive Path Conventions" (➔ p. 245)

5.17.3.3 SD Card Path Conventions

Access to the SD card memory requires that a valid SD card label be used at the beginning of the path, followed by the relative path to the SD card.

```
(Valid SD Card Label)/(Relative Path)
```

- A valid SD card relative path starts with `//`, `/`, `\\`, or `\`.
- This is immediately followed by `SDCard` which is followed by `\` or `/`.
- This path label is case insensitive.

The `SDCard` folder is created inside the `userdata` folder to maintain compatibility with the Simulator.

File access points to `userdata/SDCard` when a AKD PDMM SDCard path is used on the Simulator.

5.17.3.3.1 Valid Paths

Valid Paths	Notes
//SDCard/file1	
\Sdcard/dir1/file1	dir1 must have been already created.
/sdcard/dir1/file1	dir1 must have been already created.
//sdCard\file1	

5.17.3.3.2 Invalid Paths

Invalid Paths	Invalid Reason
///SDCard/file1	Started with more than two forward or two backward slashes.
/\Sdcard/dir1/file1	Started with one forward and one backward slash.
/sdcarddir1/file1	No forward or backward slash.
/sdcard1/dir1/file1	Invalid label.

See Also

- "File Name Warning and Limitations" (➔ p. 243)
- "Shared Directory Path Conventions" (➔ p. 244)
- "USB Flash Drive Path Conventions" (➔ p. 245)

5.17.3.4 USB Flash Drive Path Conventions

Access to the USB flash drive memory requires that a valid USB flash drive label be used at the beginning of the path, followed by the relative path to the USB flash drive.

```
(Valid USB Flash Drive Label)/(Relative Path)
```

- A valid USB flash drive relative path starts with `//`, `/`, `\\`, or `\`.
- This is immediately followed by `usbflash` which is followed by `\` or `/`.
- This path label is case insensitive.

The `usbflash` folder is created inside the `userdata` folder to maintain compatibility with the Simulator.

File access points to `userdata/usbflash` when a PCMM2G usbflash path is used on the Simulator.

5.17.3.4.0.1 Valid Paths

Valid Paths	Notes
//usbflash/file1	
\usbflash/dir1/file1	dir1 must have been already created.
/usbflash/dir1/file1	dir1 must have been already created.
//usbflash\file1	

5.17.3.4.0.2 Invalid Paths

Invalid Paths	Invalid Reason
///usbflash/file1	Started with more than two forward or two backward slashes.
/\usbflash/dir1/file1	Started with one forward and one backward slash.
/usbflashdir1/file1	No forward or backward slash.
/ubflash1/dir1/file1	Invalid label.

See Also

- "File Name Warning and Limitations" (➔ p. 243)
- "SD Card Path Conventions" (➔ p. 245)
- "Shared Directory Path Conventions" (➔ p. 244)

5.18 FilterOrder1

Function Block - First order filter.

5.18.1 Inputs

Input	Data Type	Range	Unit	Default	Description
XIN	REAL				Input analog value.
GAIN	REAL				Transformation gain.

5.18.2 Outputs

Output	Data Type	Range	Unit	Description
XOUT	REAL			Output signal.

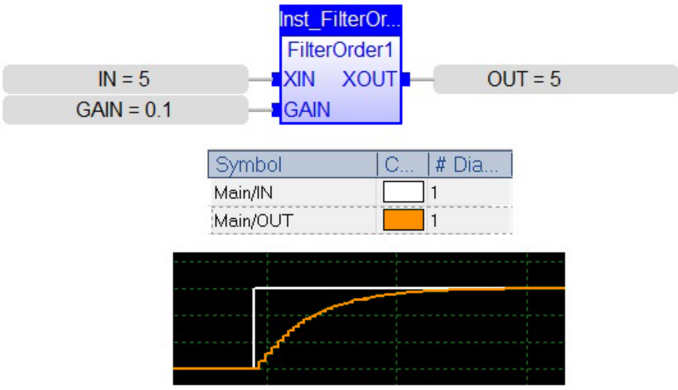
5.18.3 Remarks

The operation performed is:

```
Output = (Input x Gain) + (OutputPrev * (1-Gain))
```

The allowed range for the gain is [0.05 .. 1.0]

5.18.4 Example



5.18.5 FBD Language

Not available.

5.18.6 FFLD Language

Not available.

5.18.7 IL Language

Not available.

5.18.8 ST Language

Filt1 is a declared instance of FilterOrder1 function block.

```
Filt1 (rIn, rGain);
Signal := Fillt1.Xout;
```

5.19 GetSysInfo

PLCopen ✓

Function - Get system information.

5.19.1 Inputs

Input	Data Type	Range	Unit	Default	Description
INFO	DINT	N/A	N/A	N/A	Identifier of the requested information.

5.19.2 Outputs

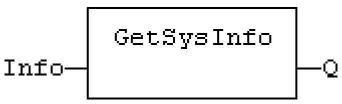
Output	Data Type	Range	Unit	Description
Q	DINT			Value of the requested information or 0 (zero) if error.

5.19.3 Remarks

The INFO parameter can be one of these predefined values:

Value	Definition
_SYSINFO_APPSTAMP	Compiling date stamp of the application.
_SYSINFO_BIGENDIAN	Non zero if the runtime processor is big endian.
_SYSINFO_CHANGE_CYCLE	Indicates a cycle just after an Online Change
_SYSINFO_CODECRC	CRC of the application code.
_SYSINFO_CYCLECOUNT	Counter of cycles.
_SYSINFO_CYCLEMAX_MICROS	Maximum detected cycle time in micro-seconds.
_SYSINFO_CYCLEMAX_MS	Maximum detected cycle time in milliseconds.
_SYSINFO_CYCLEOVERFLOWS	Number of detected cycle time overflows.
_SYSINFO_CYCLESTAMP_MS	Timestamp of the current cycle in milliseconds (OEM dependent).
_SYSINFO_CYCLETIME_MICROS	Duration of the previous cycle in micro-seconds.
_SYSINFO_CYCLETIME_MS	Duration of the previous cycle in milliseconds.
_SYSINFO_DATACRC	CRC of the application symbols.
_SYSINFO_DBSIZE	Space used in RAM (bytes).
_SYSINFO_DEMOAPP	Non zero if the application was compiled in DEMO mode.
_SYSINFO_ELAPSED	Seconds elapsed since startup.
_SYSINFO_FREEHEAP	Available space in memory heap (bytes).
_SYSINFO_NBBREAKPOINTS	Number of installed breakpoints.
_SYSINFO_NBLOCKED	Number of locked variables.
_SYSINFO_TRIGGER_MICROS	Programmed cycle time in micro-seconds.
_SYSINFO_TRIGGER_MS	Programmed cycle time in milliseconds.
_SYSINFO_WARMSTART	Non zero if RETAIN variables were loaded at the last start.

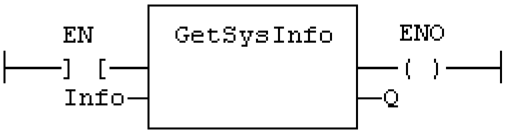
5.19.4 FBD Language



5.19.5 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.

- The function is executed only if EN is TRUE.
- ENO keeps the same value as EN.



5.19.6 IL Language

- In the IL language, the input must be loaded in the current result before calling the function.

```
Op1: LD   INFO
     GETSYSINFO
     ST   Q
```

5.19.7 ST Language

```
Q := GETSYSINFO (INFO);
```

5.20 hyster

Function Block - Hysteresis detection.

5.20.1 Inputs

Input	Data Type	Range	Unit	Default	Description
XIN1	REAL				First input.
XIN2	REAL				Second input.
EPS	REAL				Hysteresis.

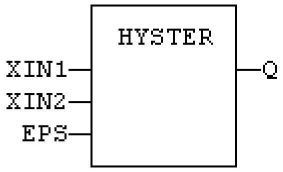
5.20.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL			Detected hysteresis: TRUE if XIN1 becomes greater than XIN2+EPS and is not yet below XIN2-EPS.

5.20.3 Remarks

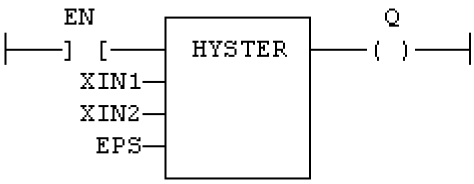
- The hysteresis is detected on the difference of XIN1 and XIN2 signals.

5.20.4 FBD Language



5.20.5 FFLD Language

- In the FFLD language, the input rung (EN) is used for enabling the block.
  - The output rung is the Q output.
  - The block is not called if EN is FALSE.



5.20.6 IL Language

```
(* MyHyst is a declared instance of HYSTER function block *)
Op1: CAL MyHyst (XIN1, XIN2, EPS)
FFLD  MyHyst.Q
ST   Q
```

5.20.7 ST Language

```
(* MyHyst is a declared instance of HYSTER function block. *)
MyHyst (XIN1, XIN2, EPS);
Q := MyHyst.Q;
```

See Also

- average / averageL
- derivate
- integral
- lim_alrm
- stackint

5.21 integral

PLCopen ✓

Function Block - Calculates the integral of a signal with respect to time.

5.21.1 Inputs

Input	Data Type	Range	Unit	Default	Description
CYCLE	TIME				Sampling period. Must not be less than the target cycle timing.

Input	Data Type	Range	Unit	Default	Description
R1	BOOL				Overriding reset.
RUN	BOOL				Run command: • TRUE = integrate. • FALSE = hold.
X0	REAL				Initial value.
XIN	REAL				Input signal.

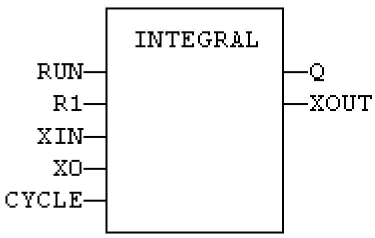
5.21.2 Outputs

Output	Data Type	Range	Unit	Description
Q	DINT			Running mode report: NOT (R1).
XOUT	REAL			Output signal.

5.21.3 Remarks

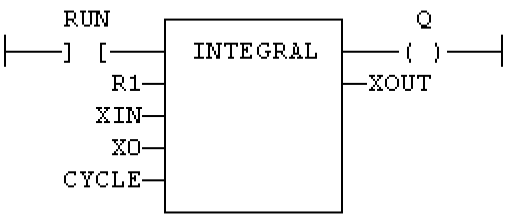
- The time unit is seconds.
- The output signal has the units of the input signal multiplied by seconds.
- The **integral** block samples the input signal at a maximum rate of 1 millisecond.

5.21.4 FBD Language



5.21.5 FFLD Language

- In the FFLD language, the input rung is the RUN command.
  - The output rung is the Q report status.



5.21.6 IL Language

```
(* MyIntg is a declared instance of INTEGRAL function block. *)
Op1: CAL MyIntg (RUN, R1, XIN, X0, CYCLE)
     FFLD  MyIntg.Q
     ST   Q
     FFLD  MyIntg.XOUT
     ST   XOUT
```

5.21.7 ST Language

```
(* MyIntg is a declared instance of INTEGRAL function block. *)
MyIntg (RUN, R1, XIN, X0, CYCLE);
Q := MyIntg.Q;
XOUT := MyIntg.XOUT;
```

See Also

- "average / averageL" (→ p. 232)
- "derivate" (→ p. 234)
- "hyster" (→ p. 249)
- "lim_alrm" (→ p. 254)
- stackint

5.22 LIFO

PLCopen ✓

Function block - Manages a "last in / first out" stack.

5.22.1 Inputs

PUSH	**BOOL**	Push a new value (on rising edge).
POP	**BOOL**	Pop a new value (on rising edge).
RST	**BOOL**	Reset the list.
NEXTIN	**ANY**	Value to be pushed.
NEXTOUT	**ANY**	Value at the top of the stack - *updated after call!*.
BUFFER	**ANY**	Array for storing values.

5.22.2 Outputs

EMPTY	**BOOL**	TRUE if the stack is empty.
OFLO	**BOOL**	TRUE if overflow on a PUSH command.
COUNT	**DINT**	Number of values in the stack.
PREAD	**DINT**	Index in the buffer of the top of the stack.
PWRITE	**DINT**	Index in the buffer of the next push position.

5.22.3 Remarks

NEXTIN, NEXTOUT and BUFFER must have the same data type *and cannot be STRING*. The NEXTOUT argument specifies a variable which is filled with the value at the top of the stack after the block is called.

Values are stored in the BUFFER array. Data is never shifted or reset. Only read and write pointers and pushed values are updated. The maximum size of the stack is the dimension of the array.

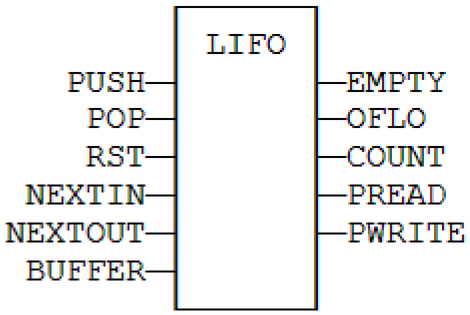
The first time an instance of the LIFO function block is called, that instance will store which array is passed to BUFFER. If a later call to the same instance passes a different array for the BUFFER argument, the call is considered invalid and no action is performed. The EMPTY output returns TRUE in this case.

In FFLD language, input rung is the PUSH input. The output rung is the EMPTY output.

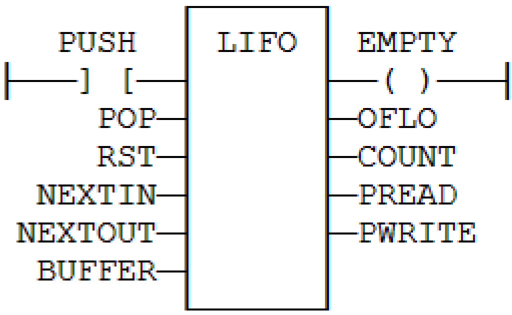
5.22.4 ST Language

```
(* MyLIFO is a declared instance of LIFO function block. *)
MyLIFO (PUSH, POP, RST, NEXTIN, NEXTOUT, BUFFER);
EMPTY := MyLIFO.EMPTY;
OFLO := MyLIFO.OFLO;
COUNT := MyLIFO.COUNT;
PREAD := MyLIFO.PREAD;
PWRITE := MyLIFO.PWRITE;
```

5.22.5 FBD Language



5.22.6 FFLD Language



5.22.7 IL Language

```
(* MyLIFO is a declared instance of LIFO function block *)
Op1: CAL MyLIFO (PUSH, POP, RST, NEXTIN, NEXTOUT, BUFFER)
FFLD  MyLIFO.EMPTY
ST   EMPTY
```

```
FFLD  MyLIFO.OFLO
ST  OFLO
FFLD  MyLIFO.COUNT
ST  COUNT
FFLD  MyLIFO.PREAD
ST  PREAD
FFLD  MyLIFO.PWRITE
ST  PWRITE
```

See Also

FIFO

5.23 lim_alrm

PLCopen ✓

Function Block - Detects high and low limits of a signal with hysteresis.

5.23.1 Inputs

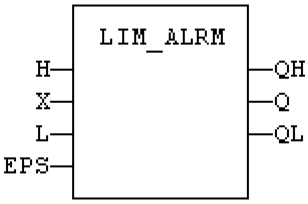
Input	Data Type	Range	Unit	Default	Description
EPS	REAL				Value of the hysteresis.
H	REAL				Value of the high limit.
L	REAL				Value of the low limit.
X	REAL				Input signal.

5.23.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL			TRUE if the signal exceeds one of the limits. Equals to QH OR QL.
QH	BOOL			TRUE if the signal exceeds the high limit.
QL	BOOL			TRUE if the signal exceeds the low limit.

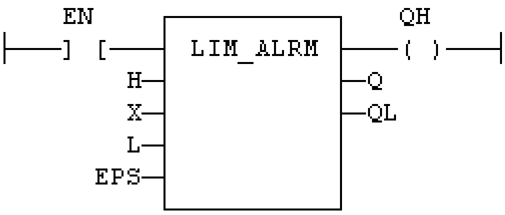
5.23.3 Remarks

5.23.4 FBD Language



5.23.5 FFLD Language

- In the FFLD language, the input rung (EN) is used for enabling the block.
  - The output rung is the QH output.
  - The block is not called if EN is FALSE.



5.23.6 IL Language

```
(* MyAlarm is a declared instance of LIM_ALRM function block *)
Op1: CAL MyAlarm (H, X, L, EPS)
     FFLD  MyAlarm.QH
     ST  QH
     FFLD  MyAlarm.Q
     ST  Q
     FFLD  MyAlarm.QL
     ST  QL
```

5.23.7 ST Language

```
(* MyAlarm is a declared instance of LIM_ALRM function block *)
MyAlarm (H, X, L, EPS);
QH := MyAlarm.QH;
Q := MyAlarm.Q;
QL := MyAlarm.QL;
```

See Also

- Alarm_A
- Alarm_M

5.24 LogFileCSV

PLCopen

Function Block - Create a log file in CSV format for a list of variables.

5.24.1 Inputs

Input	Data Type	Range	Unit	Default	Description
LOG	BOOL				Variables are saved on any rising edge of this input.
RST	BOOL				Reset the contents of the CSV file.
LIST	DINT				ID of the list of variables to log (use VLID function).
PATH	STRING				Path name of the CSV file (PxMM flash memory, SD card, or Shared Directory).

5.24.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL			TRUE if the requested operation has been performed without error.
ERR	DINT			Error report for the last requested operation. 0 (zero) is OK.

5.24.3 Remarks

IMPORTANT

Calling this function can lead to missing several PLC cycles.
Files are opened and closed directly by the target's Operating System.
Opening some files may be dangerous for system safety and integrity.
The number of open files may be limited by the target system.

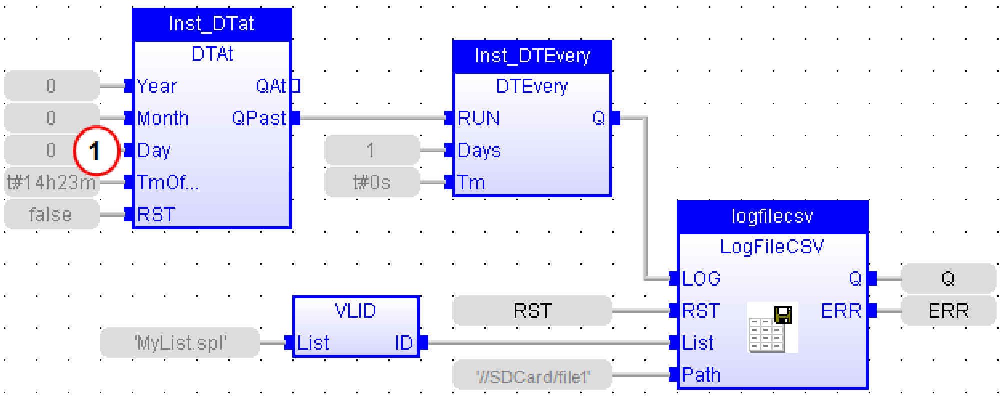
NOTE

- Opening a file may be unsuccessful (invalid path or file name, too many open files...). Your application has to process such error cases in a safe way.
- File management may be not available on some targets.
  - See the OEM instructions for more information about available features.
- Valid paths for storing files depend on the target implementation.
  - See the OEM instructions for more information about available paths.

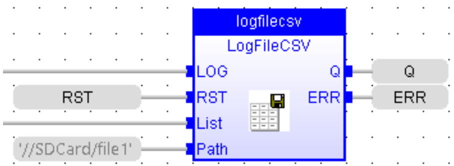
- This function enables to log values of a list of variables in a CSV file.
  - On each rising edge of the LOG input, one more line of values is added to the file.
  - There is one column for each variable, as they are defined in the list.
- The list of variables is prepared using the KAS IDE or a text editor.
  - Use the VLID function to get the identifier of the list.
- On a rising edge of the RST command, the file is emptied.
- When a LOG or RST command is requested, the Q output is set to TRUE if successful.
- In case of error, a report is given in the ERR output.
  - Possible error values are:
    - 1 = Cannot reset file on a RST command.
    - 2 = Cannot open file for data storing on a LOG command.
    - 3 = Embedded lists are not supported by the runtime.
    - 4 = Invalid list ID.
    - 5 = Error while writing to file.

- Combined with real time clock management functions, this block provides a very easy way to generate a periodical log file.

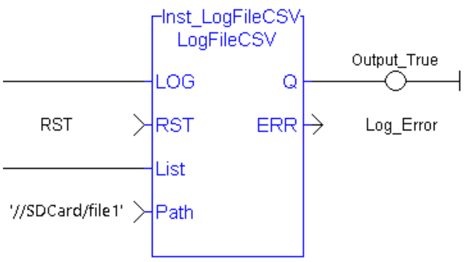
This example shows a list and a program that log values everyday at 14h23m (2:23 pm) (see call out 1).



5.24.4 FBD Language



5.24.5 FFLD Language



5.24.6 IL Language

```
(* MyLOG is a declared instance of LogFileCSV function block *)
Op1: CAL MyLOG (b_LOG, RST, LIST, PATH);
FFLD  MyLOG.Q
ST  Q
FFLD  MyLog.ERR
ST  ERR
```

5.24.7 ST Language

```
(* MyLOG is a declared instance of LogFileCSV function block *)
MyLOG (b_LOG, RST, LIST, PATH);
```

```
:= MyLOG.Q;
R := MyLog.ERR;
```

See Also

VLID

5.25 PID

PLCopen

Function Block - PID loop

5.25.1 Inputs

Input	Data Type	Range	Unit	Default	Description
AUTO	BOOL				• TRUE = normal mode • FALSE = manual mode
DEADB_ERR	REAL				Hysteresis on PV. PV is considered as unchanged if it is both: • Greater than (PVprev - DEADBAND_W). • Less than (PRprev + DEADBAND_W).
FFD	REAL				Disturbance value on output.
I_ITL_ON	BOOL				If TRUE, the integrated value is reset to I_ITLVAL.
I_ITLVAL	REAL				Reset value for integration when I_ITL_ON is TRUE.
I_SEL	BOOL				If FALSE, the integrated value is ignored.
INT_HOLD	BOOL				If TRUE, the integrated value is frozen.
KP	REAL				Gain.
PV	REAL				Process value.
SP	REAL				Set point.
TD	REAL				Derivation factor.
TI	REAL				Integration factor.
TS	TIME				Sampling period.
XMAX	REAL				Maximum output value.
XMIN	REAL				Minimum allowed output value.
Xout_Manu	REAL				Output value in manual mode.

5.25.2 Outputs

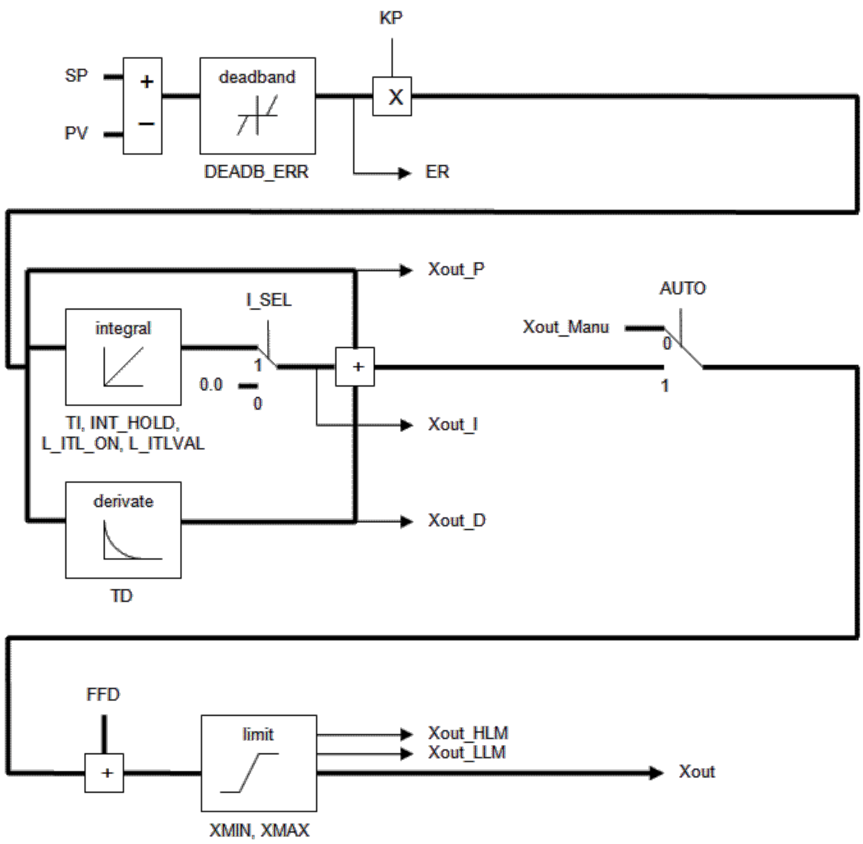
Output	Data Type	Range	Unit	Description
ER	REAL			Last calculated error.
Xout	REAL			Output command value.

Output	Data Type	Range	Unit	Description
Xout_D	REAL			Last calculated derivated value.
Xout_HLM	BOOL			TRUE if the output value is saturated to XMAX.
Xout_I	REAL			Last calculated integrated value.
Xout_LLM	BOOL			TRUE if the output value is saturated to XMIN.
Xout_P	REAL			Last calculated proportional value.

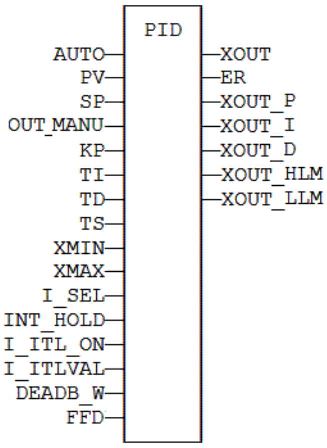
5.25.3 Remarks

- It is important for the stability of the control that the TS sampling period is much bigger than the cycle time.
- Output of the PID block always starts with zero.
  - The value varies per the inputs provided upon further cycle executions.
- In the FFLD Language, the output rung has the same value as the AUTO input, corresponding to the input rung.

5.25.3.1 Diagram

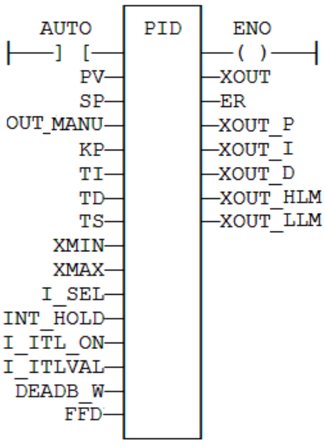


5.25.4 FBD Language



5.25.5 FFLD Language

(* ENO has the same state as the input rung. *)



5.25.6 IL Language

```
(* MyPID is a declared instance of PID function block. *)
Op1:    CAL MyPID (AUTO, PV, SP, XOUT_MANU, KP, TI, TD, TS, XMIN, XMAX, I_
SEL, I_ITL_ON, I_ITLVAL, DEADB_ERR, FFD)
        FFLD  MyPID.XOUT
        ST  XOUT
        FFLD  MyPID.ER
        ST  ER
        FFLD  MyPID.XOUT_P
        ST  XOUT_P
        FFLD  MyPID.XOUT_I
        ST  XOUT_I
        FFLD  MyPID.XOUT_D
        ST  XOUT_D
        FFLD  MyPID.XOUT_HLM
        ST  XOUT_HLM
        FFLD  MyPID.XOUT_LLM
        ST  XOUT_LLM
```

5.25.7 ST Language

```
(* MyPID is a declared instance of PID function block. *)
MyPID (AUTO, PV, SP, XOUT_MANU, KP, TI, TD, TS, XMIN, XMAX, I_SEL, I_ITL_
ON, I_ITLVAL, DEADB_ERR, FFD);
XOUT := MyPID.XOUT;
ER := MyPID.ER;
XOUT_P := MyPID.XOUT_P;
XOUT_I := MyPID.XOUT_I;
XOUT_D := MyPID.XOUT_D;
XOUT_HLM := MyPID.XOUT_HLM;
XOUT_LLM := MyPID.XOUT_LLM;
```

5.26 PWM

PLCopen ✓

Function Block - Generate a PWM signal.

5.26.1 Inputs

Input	Data Type	Range	Unit	Default	Description
XIN	REAL				Input analog value.
XinMin	REAL				Minimum input value.
XinMax	REAL				Maximum input value.
MinPulse	TIME				Minimum pulse time on output.
Period	TIME				Period of the output signal.

5.26.2 Outputs

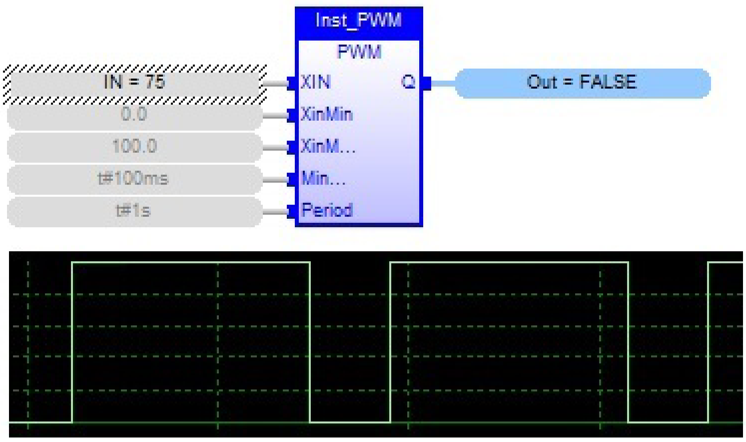
Output	Data Type	Range	Unit	Default	Description
Default (.Q) **is this correct?** **the original text here was just Q**	BOOL				Blinking PWM signal.

5.26.3 Remarks

- The input value is truncated to `[XinMin .. XinMax]` interval.
  - **XinMax** must be greater than **XinMin**.
- The signal is TRUE during:

```
(Xin - XinMin) * Period / (XinMax - XinMin)
```

5.26.4 FBD Language



5.26.5 FFLD Language

Not available. - IS THIS TRUE?

5.26.6 IL Language

Not available. - IS THIS TRUE?

5.26.7 ST Language

PWM1 is a declared instance of PWM function block.

```
PWM1 (rIn, rInMin, rInMax, tMinPulse, tPeriod);
Signal := PWM1.Q;
```

5.27 rand

Function Block - Returns a pseudo-random integer value between 0 (zero) and (base - 1).

5.27.1 Inputs

Input	Data Type	Range	Unit	Default	Description
base	DINT	1 to 2147483647	N/A	No default	The number of possible outcomes. Example: When base is 5, there are 5 possible outcomes: 0,1,2,3,4.

5.27.2 Outputs

Output	Data Type	Range	Unit	Description
Return Value	DINT	0 (zero) to (base - 1)	N/A	The generated pseudo-random number.

5.27.3 Remarks

- **rand** uses a low-quality, but fast number generation algorithm.
  - It is sufficient for small bases and where security is not a concern.
- There is no way to seed the random number generator.
  - It is possible to receive the same pattern of generated numbers after the controller reboots.

5.27.4 FBD Language

Not available.

5.27.5 FFLD Language

Not available.

5.27.6 IL Language

Not available.

5.27.7 ST Language

```
dieValue := 1 + rand(6);
```

5.28 RAMP

PLCopen ✓

Function - Limit the ascendance or descendance of a signal.

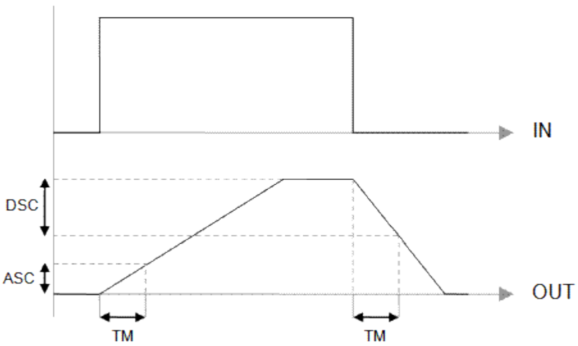
5.28.1 Inputs

Input	Data Type	Description
IN	REAL	Input signal.
ASC	REAL	Maximum ascendance during time base.
DSC	REAL	Maximum descendance during time base.
TM	TIME	Time base.
RST	BOOL	Reset.

5.28.2 Outputs

Output	Data Type	Description
OUT	REAL	Ramp signal.

5.28.3 Time Diagram



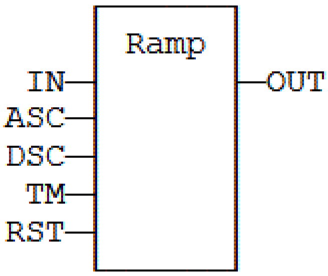
5.28.4 Remarks

- Parameters are not updated constantly.
  - They are taken into account only when the:
    - first time the block is called.
    - reset input (RST) is TRUE.
  - In these two situations, the output is set to the value of IN input.
- ASC and DSC give the maximum ascendant and descendant growth during the TB time base.
  - Both must be expressed as **positive** numbers.
- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.

5.28.5 ST Language

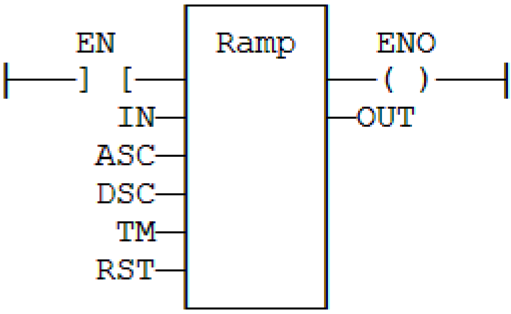
```
(* MyRamp is a declared instance of RAMP function block *)
MyRamp (IN, ASC, DSC, TM, RST);
OUT := MyBlinker.OUT;
```

5.28.6 FBD Language



5.28.7 FFLD Language

(* The function is executed only if EN is TRUE *)
(* ENO keeps the same value as EN *)



5.28.8 IL Language

```
(* MyRamp is a declared instance of RAMP function block *)
Op1: CAL
MyRamp (IN, ASC, DSC, TM, RST)
FFLD  MyBlinker.OUT
ST   OUT
```

5.29 Real Time Clock Management Functions

5.29.1 Time Zone and Clock Synchronization

These function blocks configure the time zone and clock synchronization for the controller.

Function	Description
"DTGetNTPServer" (→ p. 273)	Read the NTP server address.
"DTGetNTPSync" (→ p. 275)	Read the NTP synchronization enable state.
"DTGetTimeZone" (→ p. 277)	Read the Time Zone.
"DTListTimeZones" (→ p. 281)	List the time zones available on the controller.
"DTSetDateTime" (→ p. 285)	Sets the local date and time.
"DTSetNTPServer" (→ p. 288)	Set the NTP server address.
"DTSetNTPSync" (→ p. 289)	Set the NTP synchronization enable state.
"DTSetTimeZone" (→ p. 291)	Set the time zone.

5.29.2 Read the Real Time Clock

These functions read the real time clock of the target system:

Function	Description
"DTCurDate" (→ p. 269)	Get the present date stamp.
"DTCurDateTime" (→ p. 270)	Get the present date and time stamp.
"DTCurTime" (→ p. 272)	Get the present time stamp.

Function	Description
"DTDay" (➔ p. 272)	Get the day of the month from the date stamp.
"DTHour" (➔ p. 281)	Get the hours from the time stamp.
"DTMin" (➔ p. 283)	Get the minutes from the time stamp.
"DTMonth" (➔ p. 284)	Get the month from the date stamp.
"DTMs" (➔ p. 284)	Get the milliseconds from the time stamp.
"DTSec" (➔ p. 285)	Get the seconds from the time stamp.
"DTYear" (➔ p. 293)	Get the year from the date stamp.

5.29.3 Format the Present Date / Time

These functions format the present date/time to a string:

Function	Description
"day_time" (➔ p. 266)	Format the present date / time to a string.
"DTFormat" (➔ p. 279)	Format the present date/time to a string with a custom format.

5.29.4 Triggering Operations

These functions are used for triggering operations:

Function	Description
"DTAt" (➔ p. 267)	Generate a pulse at designated time stamp (date and time).
"DTEvery" (➔ p. 278)	Generate a pulse signal with long period.

IMPORTANT

- A real-time clock may not be available on all controller hardware models.
  See the controller hardware specifications for real-time clock availability.
- The AKD PDMM and PCMM reset the date and time when powered-on.
  The reset is to Jan 1, 1970 00:00:00.
  The elapsed time from device power-on can be determined from the Real Time Clock functions.
- PCMM2G does **not** reset the date and time when powered on.

5.29.5 day_time

Function - Format the present date / time to a string.

5.29.5.1 Inputs

Input	Data Type	Range	Unit	Default	Description
SEL	DINT	0 to 2	N/A	No default	Format string.

5.29.5.2 Outputs

Output	Data Type	Range	Unit	Description
Q	STRING	No range	N/A	String containing formatted date or time.

IMPORTANT

PCMM generation 1 controllers do **not** have real-time clock hardware.
PCMM2G does have re-time clock hardware.
Real-time clock may not be available on all controller hardware models.
See the controller hardware specifications for real-time clock availability.

5.29.5.3 Remarks

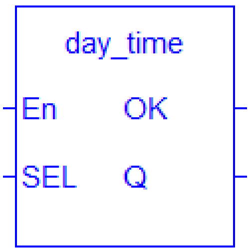
Valid values of the SEL input are:

Value	Description
0 (default)	Current date - format: YYYY/MM/DD.
1	Current time - format: HH:MM:SS.
2	Day of the week.

5.29.5.4 FBD Language

5.29.5.5 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.



5.29.5.6 IL Language

```
Op1: LD SEL
DAY_TIME
ST Q
```

5.29.5.7 ST Language

```
Q := DAY_TIME (SEL);
```

See Also

DTFormat

5.29.6 DTAt

Function Block - Generate a pulse at designated time stamp (date and time).

5.29.6.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Year	DINT	1900 to 2200	Years	No default	Year of the time stamp (e.g., 2006).
Month	DINT	1 to 12	Months	No default	Month of the time stamp (1 = January).
Day	DINT	1 to 31	Days	No default	Day of the time stamp .
TmOfDay	TIME	0 to 86,399,999	Milliseconds	No default	Time of day of the time stamp.
RST	BOOL	TRUE, FALSE	n/a	No default	Reset command.

5.29.6.2 Outputs

Output	Data Type	Range	Unit	Description
QAt	BOOL	TRUE, FALSE	N/A	Pulse signal.
QPast	BOOL	TRUE, FALSE	N/A	True if elapsed.

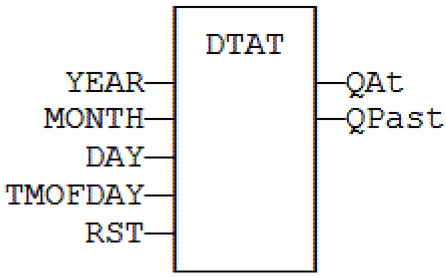
5.29.6.3 Remarks

IMPORTANT

The real-time clock may not be available on all controller hardware models.
See the controller hardware specifications for real-time clock availability.

- Parameters are not updated constantly.
  They are taken into account when only:
  - The first time the block is called.
  - When the reset input (RST) is TRUE.
- In these two situations, the outputs are reset to FALSE.
  - The first time the block is called with RST=FALSE and the specified date/stamp is passed:
    - The output QPAST is set to TRUE.
    - The output QAT is set to TRUE for one cycle only (pulse signal).
- Highest units are ignored if set to 0.
  - Example: If arguments are `year=0, month=0, day = 3, tmofday=t#10h`, the block triggers on the next 3rd day of the month at 10h.

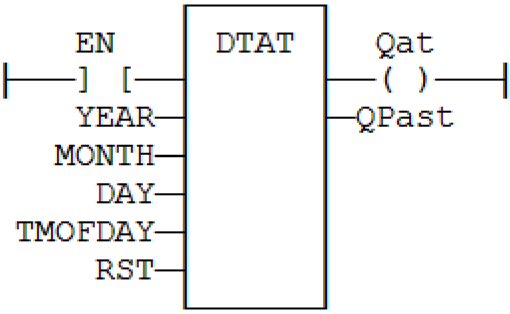
5.29.6.4 FBD Language



5.29.6.5 FFLD Language

In the FFLD language, the block is activated only if the input rung is TRUE.

(* Called only if EN is TRUE. *)



5.29.6.6 IL Language

```
(* MyDTAT is a declared instance of DTAT function block. *)
Op1: CAL
MyDTAT (YEAR, MONTH, DAY, TMOFDAY, RST)
FFLD  MyDTAT.QAT
ST  QAT
FFLD  MyDTATA.QPAST
ST  QPAST
```

5.29.6.7 ST Language

```
(* MyDTAT is a declared instance of DTAT function block. *)
MyDTAT (YEAR, MONTH, DAY, TMOFDAY, RST);
QAT := MyDTAT.QAT;
QPAST := MyDTATA.QPAST;
```

See Also

- DTEvery
- Real Time Clock Management Functions

5.29.7 DTCurDate

Function - Get the present date stamp.

5.29.7.1 Inputs

None

5.29.7.2 Outputs

Output	Data Type	Range	Unit	Description
Q	DINT	No range	N/A	Numerical stamp representing the current date.

5.29.7.3 ST Language

```
Q := DTCurDate ();
```

See Also

- "DTDay" (→ p. 272)
- "DTMonth" (→ p. 284)
- "DTYear" (→ p. 293)

5.29.8 DTCurDateTime

Function Block - Get the present date and time stamp.

5.29.8.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Local	BOOL	TRUE, FALSE	N/A	No default	• TRUE if local time is requested. • FALSE if GMT is requested.

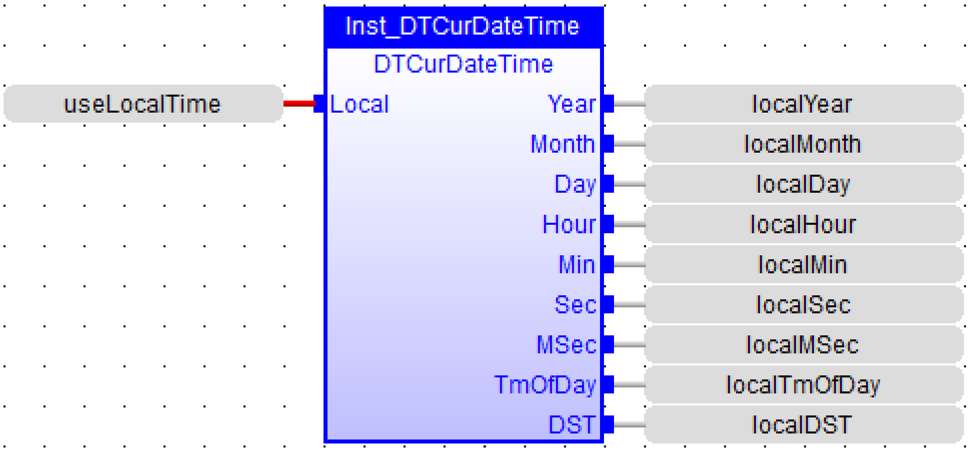
5.29.8.2 Outputs

Output	Data Type	Range	Unit	Description
Year	DINT	1900 to 2200	Years	Present year.
Month	DINT	1 to 12	Months	Present month.
Day	DINT	1 to 31	Days	Present day.
Hour	DINT	0 to 23	Hours	Present time: hours.
Min	DINT	0 to 59	Minutes	Present time: minutes.
Sec	DINT	0 to 60	Seconds	Present time: seconds.
MSec	DINT	0 to 999	Milliseconds	Present time: milliseconds.
TmOfDay	TIME	0 to 86,399,999	Milliseconds	Present time of day (milliseconds since midnight).
DST	BOOL	TRUE, FALSE	N/A	Indicates if the time is in: • Daylight saving time (DST = TRUE) • Standard time (DST = FALSE)

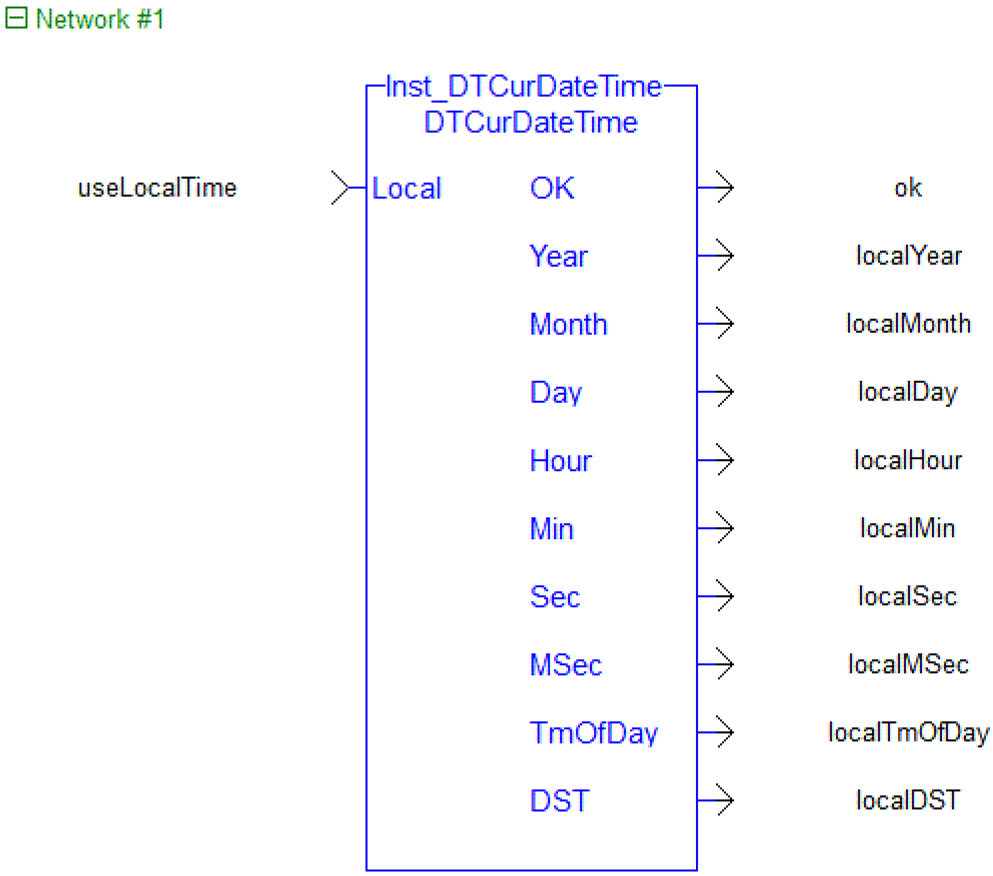
5.29.8.3 Remarks

None

5.29.8.4 FBD Language



5.29.8.5 FFLD Language



5.29.8.6 IL Language

Not available.

5.29.8.7 ST Language

```
Inst_DTCurDateTime(useLocalTime);
localYear    := Inst_DTCurDateTime.Year;
localMonth   := Inst_DTCurDateTime.Month;
localDay     := Inst_DTCurDateTime.Day;
localHour    := Inst_DTCurDateTime.Hour;
localMin     := Inst_DTCurDateTime.Min;
```

```
localSec      := Inst_DTCurDateTime.Sec;
localMSec     := Inst_DTCurDateTime.MSec;
localTmOfDay := Inst_DTCurDateTime.TmOfDay;
localDST      := Inst_DTCurDateTime.DST;
```

5.29.9 DTCurTime

Function - Get the present time stamp.

5.29.9.1 Inputs

None

5.29.9.2 Output

Output	Data Type	Range	Unit	Description
Q	DINT	0 to 86,399,999	Milliseconds	Present milliseconds of the time.

5.29.9.3 ST Language

```
Q := DTCurTime ();
```

See Also

- "DTHour" (→ p. 281)
- "DTMin" (→ p. 283)
- "DTMs" (→ p. 284)
- "DTSec" (→ p. 285)

5.29.10 DTDay

Function - Get the day of the month from the date stamp.

5.29.10.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Date	DINT	No range	N/A	No default	Numerical stamp representing a date.

5.29.10.2 Outputs

Output	Data Type	Range	Unit	Description
Q	DINT	1 to 31	N/A	Day of the month of the date.

5.29.10.3 ST Language

```
Q := DTDay (iDate);
```

See Also

- "DTCurDate" (➜ p. 269)
- "DTMonth" (➜ p. 284)
- "DTYear" (➜ p. 293)

5.29.11 DTGetNTPServer

Function Block - Read the NTP server address.

This function block is specific for PCMM2G only.

5.29.11.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	TRUE, FALSE	N/A	No default	If TRUE, request to read the NTP server address.

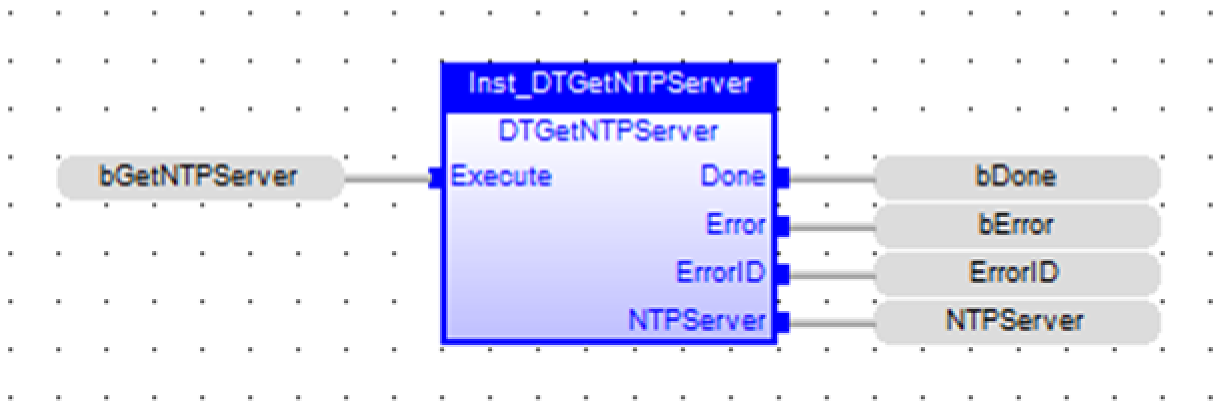
5.29.11.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	TRUE, FALSE	N/A	If TRUE, the command completed successfully.
Error	BOOL	TRUE, FALSE	N/A	If TRUE, an error has occurred.
ErrorID	DINT	No range	N/A	Indicates the error if the **Error** output is set to TRUE. **Error Codes** • 23 = Internal error. See controller log for details. • 15000 = Controller type does not support this function block. • 16200 = Could not read NTP server configuration file.
NTPServer	STRING	No range	N/A	The address of the NTP server used for clock synchronization.

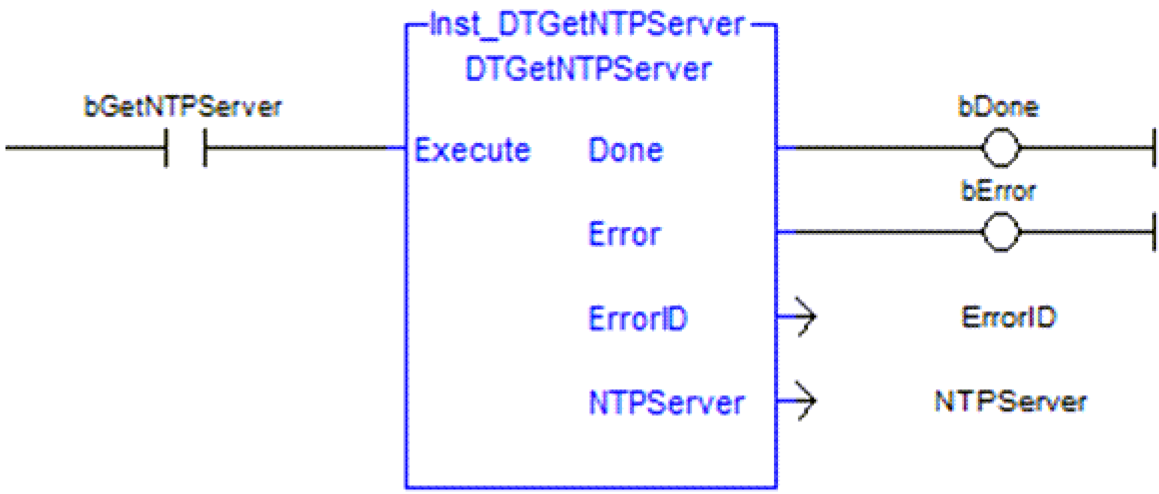
5.29.11.3 Remarks

None

5.29.11.4 FBD Language



5.29.11.5 FFLD Language



5.29.11.6 IL Language

Not available.

5.29.11.7 ST Language

```
// read the NTP server address
Inst_DTGetNTPServer( bGetNTPServer );
if Inst_DTGetNTPServer.Done then
   bGetNTPServer := false;
   if NOT Inst_DTGetNTPServer.Error then
      NTPServer := Inst_DTGetNTPServer.NTPServer;
   else
      ErrorID := Inst_DTGetNTPServer.ErrorID;
   end_if;
end_if;
```

See Also

- "DTCurDateTime" (➜ p. 270)
- "DTGetNTPServer" (➜ p. 273)
- "DTSetNTPServer" (➜ p. 288)
- "DTSetNTPSync" (➜ p. 289)
- "List of Date / Time / NTP ErrorID Codes" (➜ p. 293)

5.29.12 DTGetNTPSync

Function Block - Read the NTP synchronization enable state.
This function block is specific for PCMM2G only.

5.29.12.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	TRUE, FALSE	N/A	No default	If TRUE, request to read the synchronization enable state.

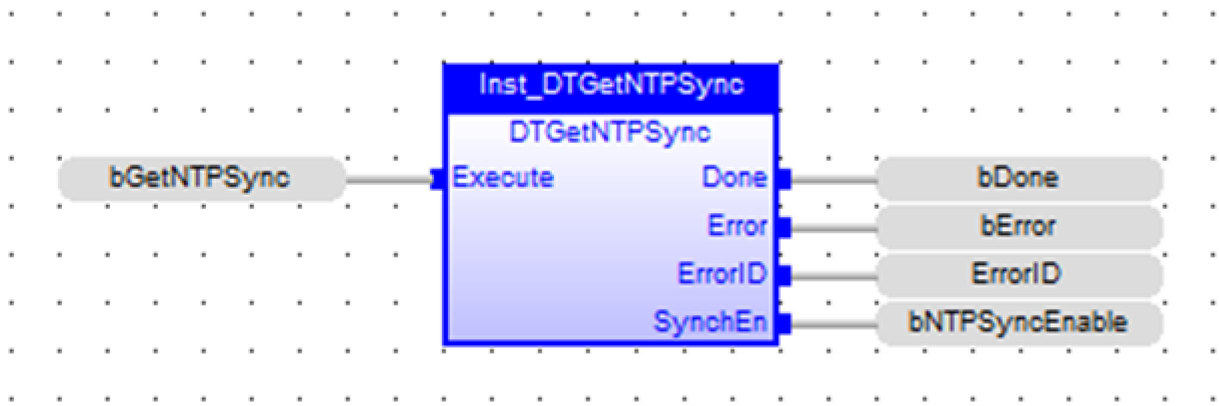
5.29.12.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	TRUE, FALSE	N/A	If TRUE, the command completed successfully.
Error	BOOL	TRUE, FALSE	N/A	If TRUE, an error has occurred.
ErrorID	DINT	No range	N/A	Indicates the error if the **Error** output is set to TRUE. **Error Codes** • 23 = Internal error. See controller log for details. • 15000 = Controller type does not support this function block.
SynchEn	BOOL	TRUE, FALSE	N/A	The present NTP synchronization state. • TRUE = synchronization enabled. • FALSE = synchronization disabled.

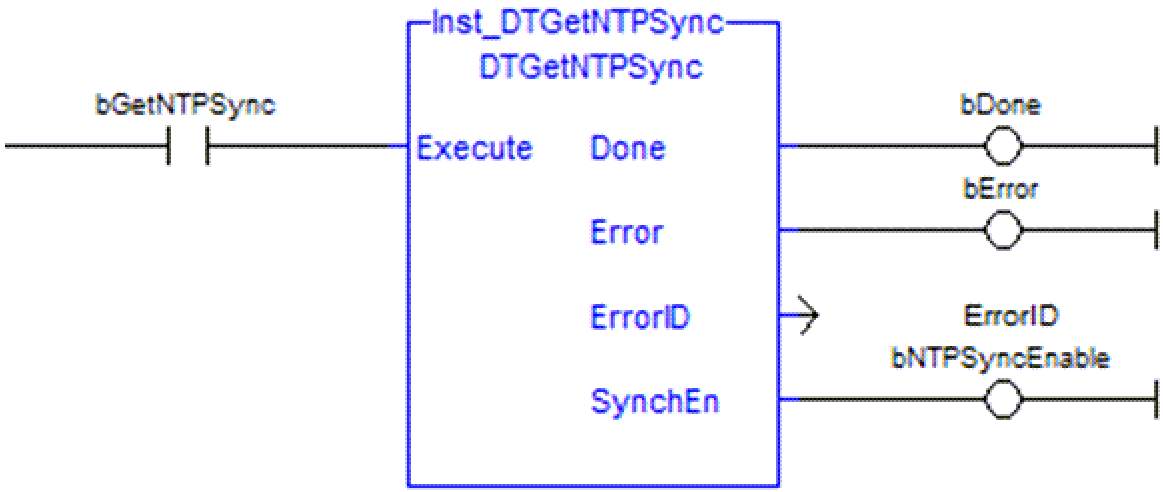
5.29.12.3 Remarks

None

5.29.12.4 FBD Language



5.29.12.5 FFLD Language



5.29.12.6 IL Language

Not available.

5.29.12.7 ST Language

```
// read the NTP synchronization state
Inst_DTGetNTPSync( bGetNTPSync );
if Inst_DTGetNTPSync.Done then
   bGetNTPSync := false;

   if NOT Inst_DTGetNTPSync.Error then
      bNTPSyncEnable := Inst_DTGetNTPSync.SynchEn;
   else
      ErrorID := Inst_DTGetNTPSync.ErrorID;
   end_if;
end_if;
```

See Also

- "DTCurDateTime" (➜ p. 270)
- "DTSetNTPSync" (➜ p. 289)
- "List of Date / Time / NTP ErrorID Codes" (➜ p. 293)

5.29.13 DTGetTimeZone

Function Block - Read the Time Zone.

This function block is specific for PCMM2G only.

5.29.13.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	TRUE, FALSE	N/A	No default	If TRUE, request to read the time zone.

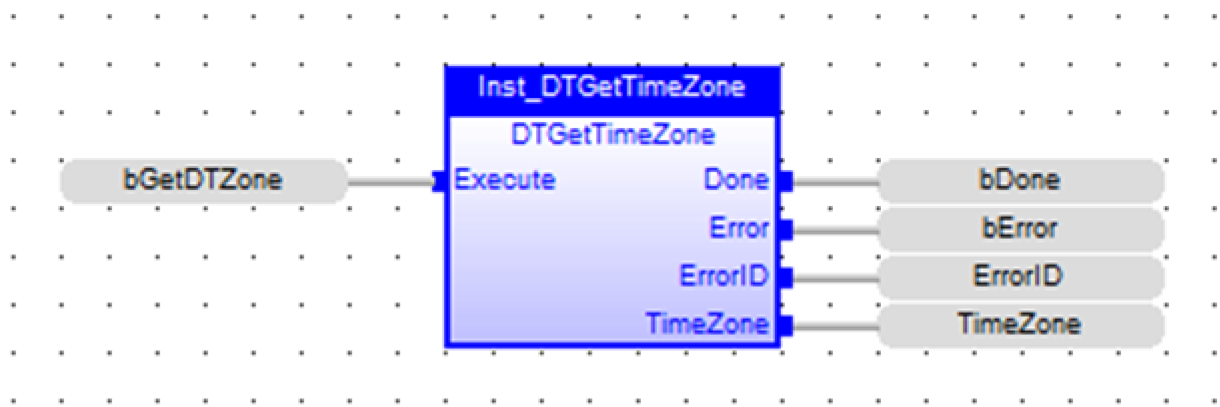
5.29.13.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	TRUE, FALSE	N/A	If TRUE, the command completed successfully.
Error	BOOL	TRUE, FALSE	N/A	If TRUE, an error has occurred.
ErrorID	DINT	No range	N/A	Indicates the error if the **Error** output is set to TRUE. **Error Codes** • 23 = Internal error. See controller log for details. • 15000 = Controller type does not support this function block.
TimeZone	STRING	No range	N/A	The time zone the controller should use.

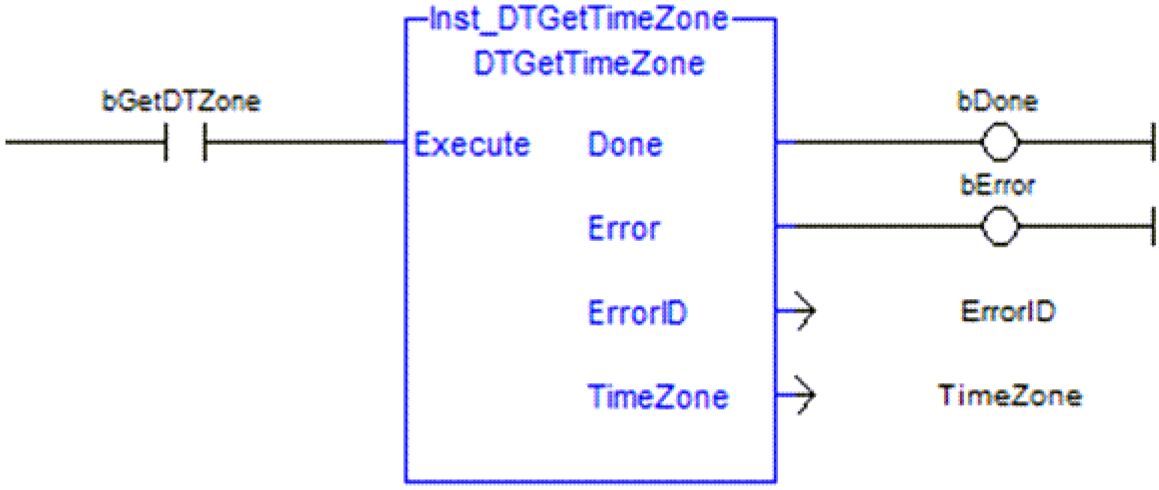
5.29.13.3 Remarks

None

5.29.13.4 FBD Language



5.29.13.5 FFLD Language



5.29.13.6 IL Language

Not available.

5.29.13.7 ST Language

```
// read the configured time zone
Inst_DTGetTimeZone( bGetDTZone );
if Inst_DTGetTimeZone.Done then
   bGetDTZone := false;

   if NOT Inst_DTGetTimeZone.Error then
      TimeZone := Inst_DTGetTimeZone.TimeZone;
   else
      ErrorID := Inst_DTGetTimeZone.ErrorID;
   end_if;
end_if;
```

See Also

- "DTCurDateTime" (➜ p. 270)
- "DTSetTimeZone" (➜ p. 291)
- "List of Date / Time / NTP ErrorID Codes" (➜ p. 293)

5.29.14 DTEvery

Function Block - Generate a pulse signal with long period.

5.29.14.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Run	BOOL	TRUE, FALSE	N/A	No default	When TRUE, the signal generation is enabled.

Input	Data Type	Range	Unit	Default	Description
Days	DINT	1 to 65536	Days	No default	Period : number of days.
TM	Time	0 to 86,399,999	Milliseconds	No default	Rest of the period (if not a multiple of 24h).

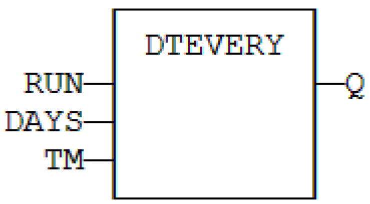
5.29.14.2 Outputs

Output	Data Type	Range	Unit	Description
Q	BOOL	TRUE, FALSE	N/A	Pulse signal.

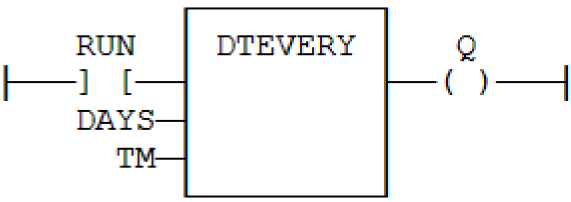
5.29.14.3 Remarks

- This function block provides a pulse signal with a period of more than 24h.
  - The period is expressed as:
    `DAYS * 24h + TM`
- Example: Specifying DAYS=1 and TM=6h means a period of 30 hours.

5.29.14.4 FBD Language



5.29.14.5 FFLD Language



5.29.14.6 IL Language

Not available.

5.29.14.7 ST Language

```
(* MyDTEVERY is a declared instance of DTEVERY function block. *)
MyDTEVERY (RUN, DAYS, TM);
Q := MyDTEVERY.Q;
```

See Also

- DTAt
- Real Time Clock Management Functions

5.29.15 DTFormat

Function - Format the present date/time to a string with a custom format.

5.29.15.1 Inputs

Input	Data Type	Range	Unit	Default	Description
FMT	STRING	No range	n/a	'%Y/%m/%d - %H:%M:%S'	Format string

5.29.15.2 Outputs

Output	Data Type	Range	Unit	Description
Q	STRING	No range	N/A	String containing formatted date or time.

5.29.15.3 Remarks

IMPORTANT

The real-time clock may not be available on all controller hardware models.
See the controller hardware specifications for real-time clock availability.

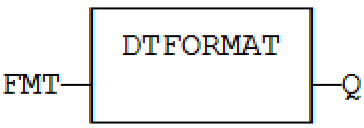
- The format string may contain any character.
- Special markers beginning with the **%** character indicates a date/time information:

Marker	Description
%Y	Year including century (e.g., 2006)
%y	Year without century (e.g., 06)
%m	Month (1..12)
%d	Day of the month (1..31)
%H	Hours (0..23)
%M	Minutes (0..59)
%S	Seconds (0..59)
%T	Milliseconds (0..999)

Example

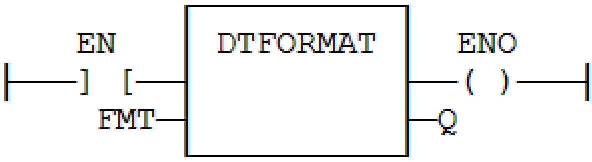
```
(* we are at July 04th 2006, 18:45:20 *)
   Q := DTFORMAT ('Today is %Y/%m/%d -%H:%M:%S');
(* Q is 'Today is 2006/07/04 - 18:45:20 *)
```

5.29.15.4 FBD Language



5.29.15.5 FFLD Language

(* The function is executed only if EN is TRUE. *)
(* ENO keeps the same value as EN. *)



5.29.15.6 IL Language

```
Op1: LD FMT
DTFORMAT
ST Q
```

5.29.15.7 ST Language

```
Q := DTFORMAT (FMT);
```

See Also

day_time

5.29.16 DTHour

Function - Get the hours from the time stamp.

5.29.16.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Time	DINT	0 to 86,399,999	Milliseconds	No default	The number of milliseconds that have passed since midnight. This value is typically retrieved from **DTCurTime**.

5.29.16.2 Outputs

Output	Data Type	Range	Unit	Description
Q	DINT	0 to 23	Hours	Hours of the time.

5.29.16.3 ST Language

```
Q := DTHour (iTime);
```

See Also

- "DTCurTime" (➔ p. 272)
- "DTMin" (➔ p. 283)
- "DTMs" (➔ p. 284)
- "DTSec" (➔ p. 285)

5.29.17 DTListTimeZones

Function Block - List the time zones available on the controller.

This function block is specific for PCMM2G only.

5.29.17.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	TRUE, FALSE	N/A	No default	If TRUE, request to read the available time zones.
TimeZones	STRING[ ]	No range	N/A	No default	An array where the list of time zones available on the system are copied. This is effectively an output parameter, but because it is an array, it must be an input.

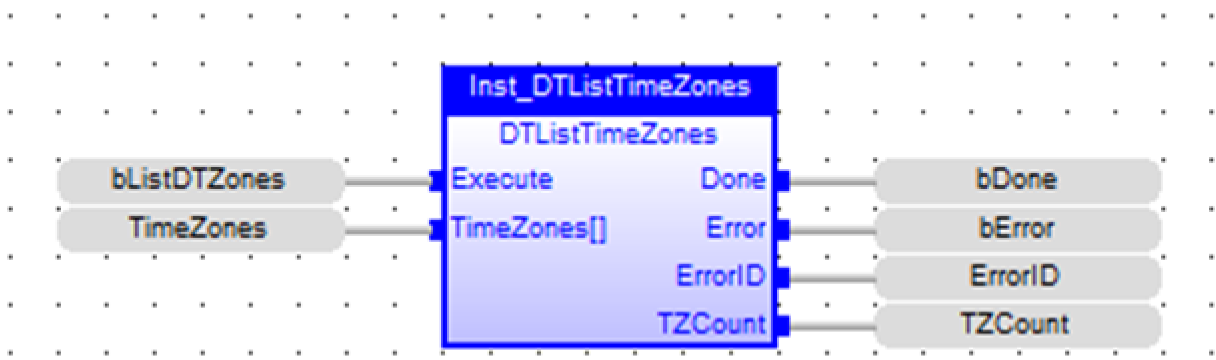
5.29.17.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	TRUE, FALSE	N/A	If TRUE, the command completed successfully.
Error	BOOL	TRUE, FALSE	N/A	If TRUE, an error has occurred.
ErrorID	DINT	No range	N/A	Indicates the error if **Error** output is TRUE. **Error Codes** • 23 = Internal error. See controller log for details. • 15000 = Controller type does not support this function block.
TZCount	DINT	No range	N/A	The number of time zones on the system.

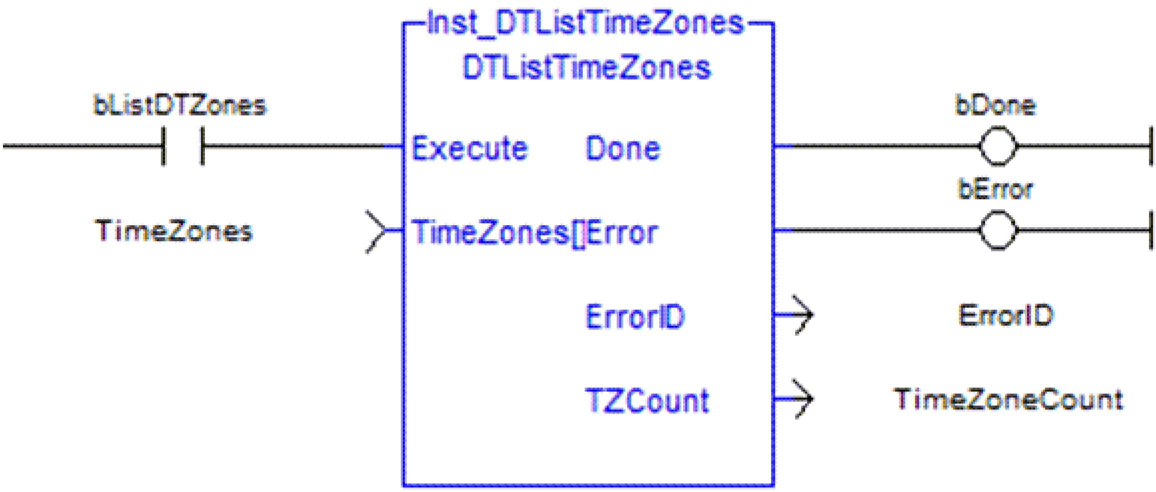
5.29.17.3 Remarks

None

5.29.17.4 FBD Language



5.29.17.5 FFLD Language



5.29.17.6 IL Language

Not available.

5.29.17.7 ST Language

```
// read the list of supported time zones
Inst_DTListTimeZones( bListDTZones, TimeZones );
if NOT Inst_DTListTimeZones.Error then
   TZCount := Inst_DTListTimeZones.TZCount;
else
   ErrorID := Inst_DTListTimeZones.ErrorID;
end_if;

if  Inst_DTListTimeZones.Done then
   bListDTZones := false;
end_if;
```

See Also

- "DTGetTimeZone" (→ p. 277)
- "DTSetTimeZone" (→ p. 291)

5.29.18 DTMin

Function - Get the minutes from the time stamp.

5.29.18.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Time	DINT	0 to 86,399,999	Milliseconds	No default	The number of milliseconds that have passed since midnight. This value is typically retrieved from **DTCurTime**.

5.29.18.2 Outputs

Output	Data Type	Range	Unit	Description
Q	DINT	0 to 59	Minutes	Minutes of the time.

5.29.18.3 ST Language

```
Q := DTMin (iTime);
```

See Also

- "DTCurTime" (➔ p. 272)
- "DTHour" (➔ p. 281)
- "DTMs" (➔ p. 284)
- "DTSec" (➔ p. 285)

5.29.19 DTMonth

Function - Get the month from the date stamp.

5.29.19.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Date	DINT	No range	N/A	No default	Numerical stamp representing a date.

5.29.19.2 Outputs

Output	Data Type	Range	Unit	Description
Q	DINT	1 to 12	N/A	Month of the date.

5.29.19.3 ST Language

```
Q := DTMonth (iDate);
```

See Also

- "DTCurDate" (➔ p. 269)
- "DTDay" (➔ p. 272)
- "DTYear" (➔ p. 293)

5.29.20 DTMs

Function - Get the milliseconds from the time stamp.

5.29.20.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Time	DINT	0 to 86,399,999	Milliseconds	No default	The number of milliseconds that have passed since midnight. This value is typically retrieved from **DTCurTime**.

5.29.20.2 Outputs

Output	Data Type	Range	Unit	Description
Q	DINT	0 to 999	Milliseconds	Present milliseconds of the time.

5.29.20.3 ST Language

```
Q := DTMs (iTime);
```

See Also

- "DTCurTime" (→ p. 272)
- "DTHour" (→ p. 281)
- "DTMin" (→ p. 283)
- "DTSec" (→ p. 285)

5.29.21 DTSec

Function - Get the seconds from the time stamp.

5.29.21.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Time	DINT	0 to 86,399,999	Milliseconds	No default	The number of milliseconds that have passed since midnight. This value is typically retrieved from **DTCurTime**.

5.29.21.2 Outputs

Output	Data Type	Range	Unit	Description
Q	DINT	0 to 59	Seconds	Seconds of the time.

5.29.21.3 ST Language

```
Q := DTSec (iTime);
```

See Also

- "DTCurTime" (→ p. 272)
- "DTHour" (→ p. 281)
- "DTMin" (→ p. 283)
- "DTMs" (→ p. 284)

5.29.22 DTSetDateTime

Function Block - Sets the local date and time.

This function block is specific for PCMM2G only.

5.29.22.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	TRUE, FALSE	N/A	No default	If TRUE, request to set the local date and time.
Year	DINT	1900 to 2200	Year	No default	The local date's new value of the year.
Month	DINT	1 to 12	Month	No default	The local date's new value of the month.
Day	DINT	1 to 31	Day	No default	The local date's new value of the day.
Hour	DINT	0 to 23	Hour	No default	The local date's new value of the hour.
Min	DINT	0 to 59	Minute	No default	The local date's new value of the minute.
Sec	DINT	0 to 60	Second	No default	The local date's new value of the second. **NOTE** 60 is valid because leap seconds may have a value of 60.

TIP

If the UTC time needs to be set, change the time zone to UTC using "DTSetTimeZone" (→ p. 291), set the time, then restore the time zone.

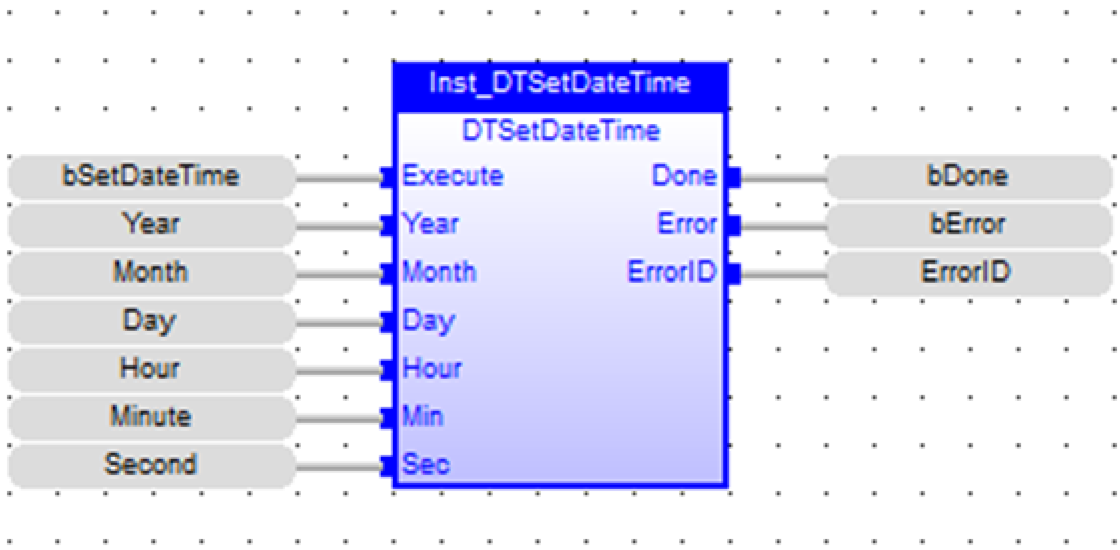
5.29.22.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	TRUE, FALSE	N/A	If TRUE, the command completed successfully.
Error	BOOL	TRUE, FALSE	N/A	If TRUE, an error has occurred.
ErrorID	DINT	No range	N/A	Indicates the error if the **Error** output is set to TRUE. **Error Codes** • 23 = Internal error. See controller log for details. • 15000 = Controller type does not support this function block. • 16202 = Cannot set date / time when NTP synchronization is active. • 16203 = Invalid date / time value specified.

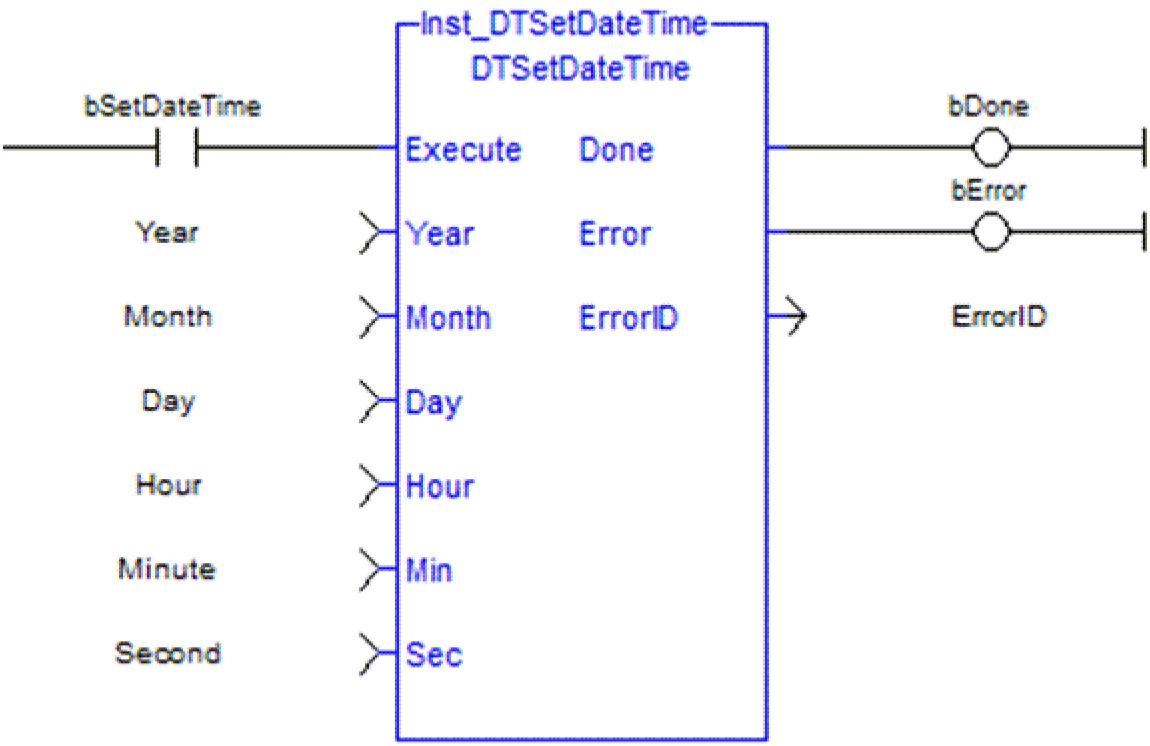
5.29.22.3 Remarks

None

5.29.22.4 FBD Language



5.29.22.5 FFLD Language



5.29.22.6 IL Language

Not available.

5.29.22.7 ST Language

```
// write the date and time
Inst_DTSetDateTime( bSetDateTime, Year, Month, Day, Hour, Minute, Second
```

```
);
if Inst_DTSetDateTime.Done then
   bSetDateTime := false;

   bError := Inst_DTSetDateTime.Error;
   ErrorID := Inst_DTSetDateTime.ErrorID;
end_if;
```

See Also

- "DTCurDateTime" (➜ p. 270)
- "DTSetTimeZone" (➜ p. 291)
- "List of Date / Time / NTP ErrorID Codes" (➜ p. 293)

5.29.23 DTSetNTPServer

Function Block - Set the NTP server address.

This function block is specific for PCMM2G only.

5.29.23.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	TRUE, FALSE	N/A	No default	If TRUE, request to set the NTP server address.
NTPServer	STRING	No range	N/A	No default	The address of the NTP server used for clock synchronization.

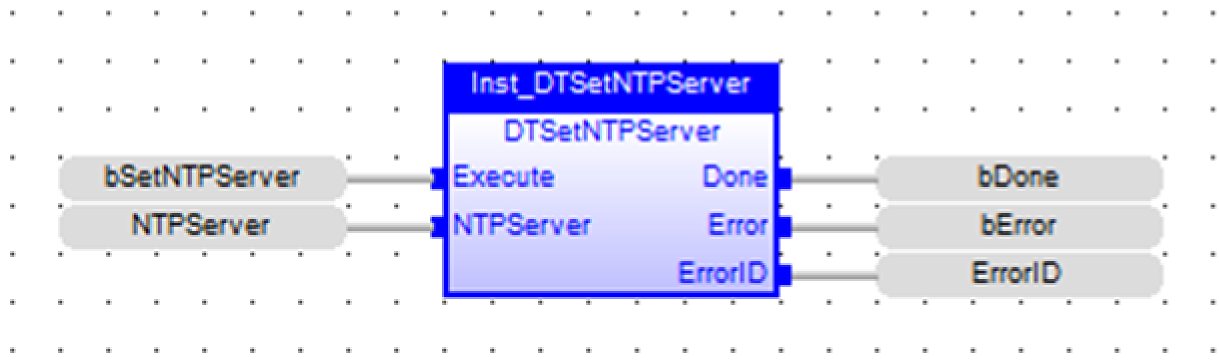
5.29.23.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	TRUE, FALSE	N/A	If TRUE, the command completed successfully.
Error	BOOL	TRUE, FALSE	N/A	If TRUE, an error has occurred.
ErrorID	DINT	No range	N/A	Indicates the error if the **Error** output is set to TRUE. **Error Codes** • 23 = Internal error. See controller log for details. • 15000 = Controller type does not support this function block.

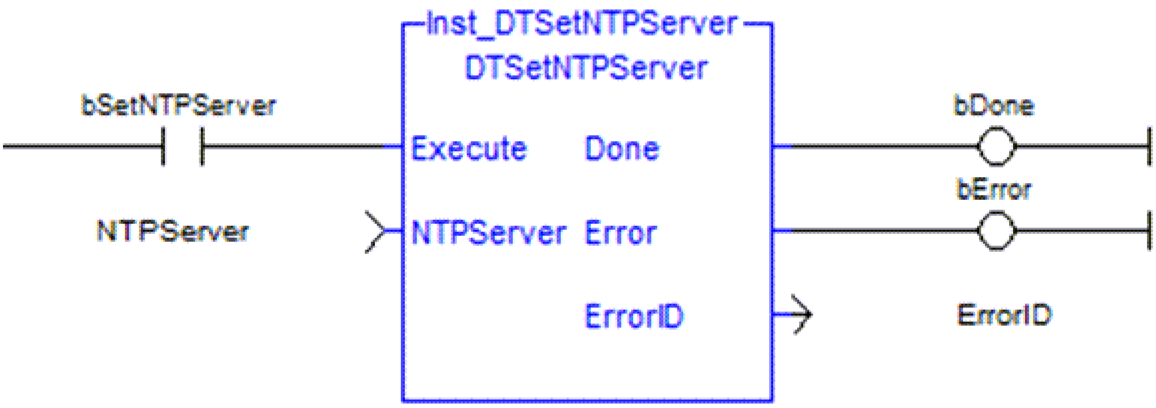
5.29.23.3 Remarks

None

5.29.23.4 FBD Language



5.29.23.5 FFLD Language



5.29.23.6 IL Language

Not available.

5.29.23.7 ST Language

```
// configure the NTP server address
Inst_DTSetNTPServer( bSetNTPServer, NTPServer );
if Inst_DTSetNTPServer.Done then
   bSetNTPServer := false;
   bError := Inst_DTSetNTPServer.Error;
   ErrorID := Inst_DTSetNTPServer.ErrorID;
end_if;
```

See Also

- "DTCurDateTime" (→ p. 270)
- "DTGetNTPServer" (→ p. 273)
- "DTGetNTPSync" (→ p. 275)
- "DTSetNTPSync" (→ p. 289)
- "List of Date / Time / NTP ErrorID Codes" (→ p. 293)

5.29.24 DTSetNTPSync

Function Block - Set the NTP synchronization enable state.

This function block is specific for PCMM2G only.

5.29.24.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	TRUE, FALSE	N/A	No default	If TRUE, request to set the synchronization enable state.
SynchEn	BOOL	TRUE, FALSE	N/A	No default	• TRUE = enable NTP synchronization. • FALSE = disable NTP synchronization.

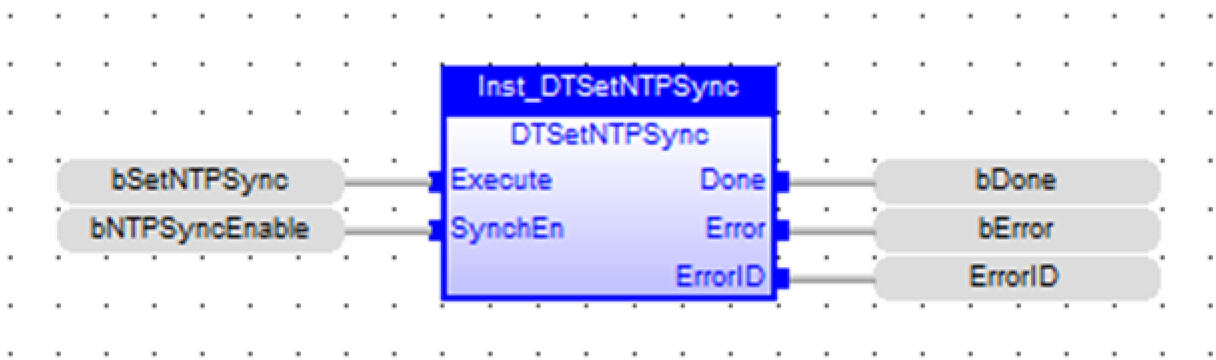
5.29.24.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	TRUE, FALSE	N/A	If TRUE, the command completed successfully.
Error	BOOL	TRUE, FALSE	N/A	If TRUE, an error has occurred.
ErrorID	DINT	No range	N/A	Indicates the error if the **Error** output is set to TRUE. **Error Codes** • 23 = Internal error. See controller log for details. • 15000 = Controller type does not support this function block.

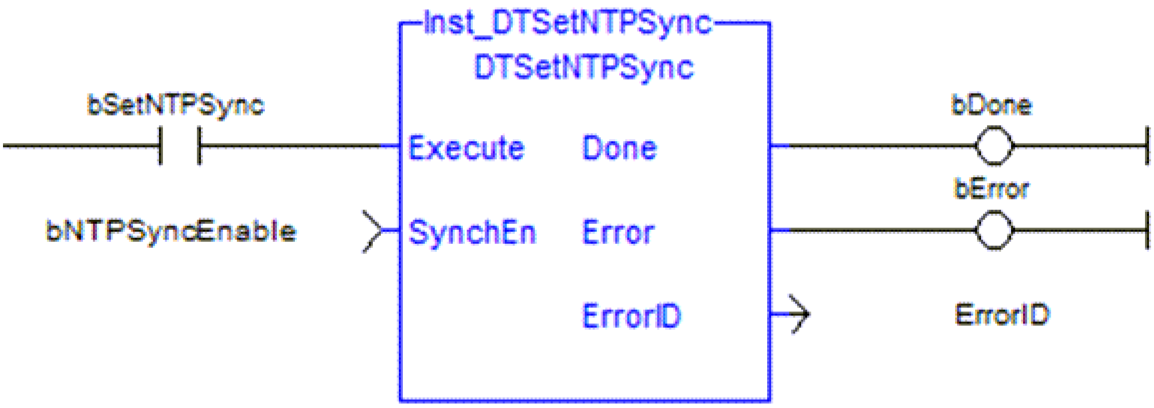
5.29.24.3 Remarks

None

5.29.24.4 FBD Language



5.29.24.5 FFLD Language



5.29.24.6 IL Language

Not available.

5.29.24.7 ST Language

```
// enable NTP server synchronization
Inst_DTSetNTPSync( bSetNTPSync, bNTPSyncEnable );
if Inst_DTSetNTPSync.Done then
   bSetNTPSync := false;

   bError := Inst_DTSetNTPSync.Error;
   ErrorID := Inst_DTSetNTPSync.ErrorID;
end_if;
```

See Also

- "DTCurDateTime" (→ p. 270)
- "DTGetNTPServer" (→ p. 273)
- "DTGetNTPSync" (→ p. 275)
- "DTSetNTPServer" (→ p. 288)
- "List of Date / Time / NTP ErrorID Codes" (→ p. 293)

5.29.25 DTSetTimeZone

Function Block - Set the time zone.
This function block is specific for PCMM2G only.

5.29.25.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Execute	BOOL	TRUE, FALSE	N/A	No default	If TRUE, request to set the time zone.
TimeZone	STRING	No range	N/A	No default	The time zone the controller should use.

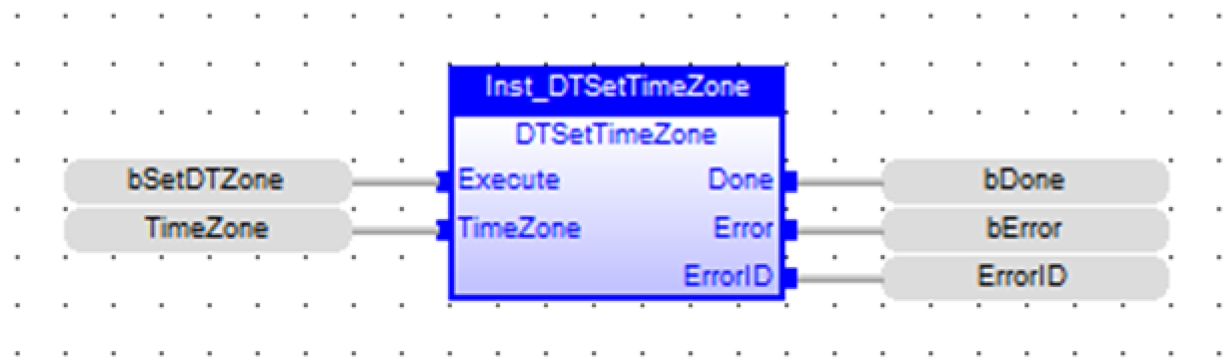
5.29.25.2 Outputs

Output	Data Type	Range	Unit	Description
Done	BOOL	TRUE, FALSE	N/A	If TRUE, the command completed successfully.
Error	BOOL	TRUE, FALSE	N/A	If TRUE, an error has occurred.
ErrorID	DINT	No range	N/A	Indicates the error if the **Error** output is set to TRUE. **Error Codes** • 23 = Internal error. See controller log for details. • 15000 = Controller type does not support this function block. • 16201 = Invalid time zone.

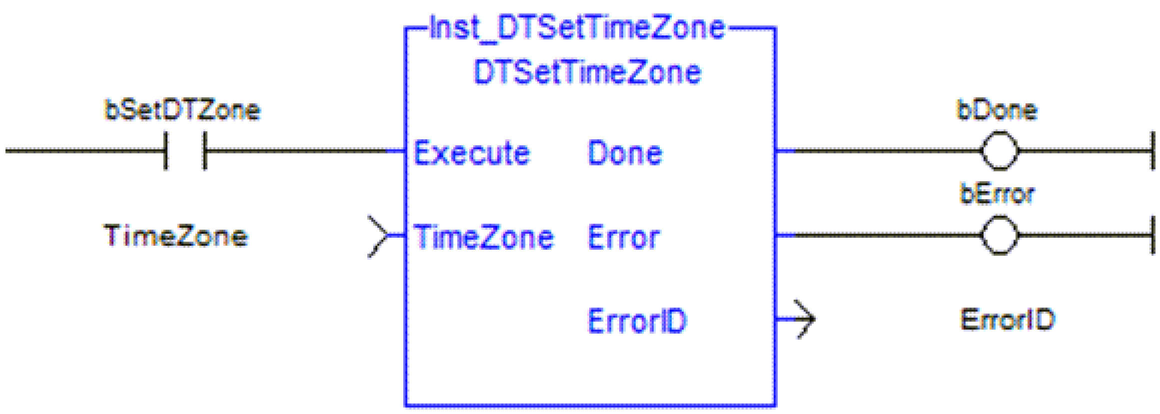
5.29.25.3 Remarks

None

5.29.25.4 FBD Language



5.29.25.5 FFLD Language



5.29.25.6 IL Language

Not available.

5.29.25.7 ST Language

```
// configure the time zone
Inst_DTSetTimeZone( bSetDTZone, TimeZone );
if Inst_DTSetTimeZone.Done then
   bSetDTZone := false;

   bError := Inst_DTSetTimeZone.Error;
   ErrorID := Inst_DTSetTimeZone.ErrorID;
end_if;
```

See Also

- "DTCurDateTime" (➜ p. 270)
- "DTGetTimeZone" (➜ p. 277)
- "DTListTimeZones" (➜ p. 281)
- "List of Date / Time / NTP ErrorID Codes" (➜ p. 293)

5.29.26 DTYear

Function - Get the year from the date stamp.

5.29.26.1 Inputs

Input	Data Type	Range	Unit	Default	Description
Date	DINT	No range	N/A	No default	Numerical stamp representing a date.

5.29.26.2 Outputs

Output	Data Type	Range	Unit	Description
Q	DINT	No range	N/A	Year of the date.

5.29.26.3 ST Language

```
Q := DTYear (iDate);
```

See Also

- "DTCurDate" (➜ p. 269)
- "DTDay" (➜ p. 272)
- "DTMonth" (➜ p. 284)

5.29.27 List of Date / Time / NTP ErrorID Codes

- 23 = Internal error. See controller log for details.
- 15000 = Controller type does not support this function block.
- 16200 = Could not read NTP server configuration file.
- 16201 = Invalid time zone.
- 16202 = Cannot set date / time when NTP synchronization is active.
- 16203 = Invalid date / time value specified.

5.30 SerializeIn

Function - Extract the value of a variable from a binary frame.

5.30.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BIGENDIAN	BOOL	TRUE, FALSE			TRUE if the frame is encoded with Big Endian format.
DATA	ANY(*)	N/A	N/A	No default	Destination variable to be copied.
EN	BOOL	0, 1	N/A	No default	Execute the function.
FRAME	USINT	0,+65535	N/A	N/A	Source buffer - must be an array.
POS	DINT	0,+65535	N/A	N/A	Position in the source buffer.

(*) DATA cannot be a STRING.

5.30.2 Outputs

Output	Data Type	Range	Unit	Description
NEXTPOS	DINT		N/A	• Position in the source buffer after the copied data. • 0 (zero) in case or error (e.g., invalid position or buffer size).
OK	BOOL		N/A	Returns true when the function successfully executes. See Function - General Rules.

5.30.3 Remarks

This function is used for extracting data from a communication frame in binary format.

- The **DATA** input must be directly connected to a variable.
  - It cannot be a constant or complex expression.
  - This variable is forced with the extracted value.

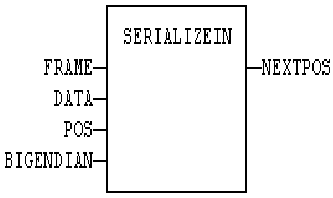
- The **FRAME** input must fit the input position and data size.
  - If the value cannot be safely extracted, the function returns 0 (zero).

- This function cannot be used to serialize STRING variables.
- The function returns the position in the source frame, after the extracted data.
  - The return value can be used as a position for the next serialization.

This function extracts these number of bytes from the source frame:

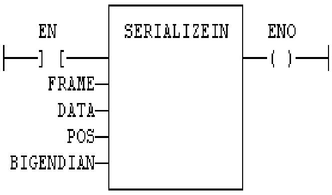
Bytes	Description
1 byte	BOOL, BYTE, SINT, and USINT variables.
2 bytes	INT, UINT, and WORD variables.
4 bytes	DINT, DWORD, REAL, and UDINT variables.
8 bytes	LINT and LREAL variables.

5.30.4 FBD Language



5.30.5 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.



5.30.6 IL Language

Not available.

5.30.7 ST Language

```
Q := SERIALIZEIN (FRAME, DATA, POS, BIGENDIAN);
```

See Also

"SerializeOut" (→ p. 296)

5.30.8 Arguments

5.30.8.1 Input

En	**Description**	Execute the function
	Data type	BOOL
	Range	[0,1]
	Unit	N/A
	Default	_
Frame[]	**Description**	Source buffer - must be an array.
	Data type	USINT
	Range	[0,+65535]
	Unit	N/A
	Default	N/A

Data	Description	Destination variable to be copied
	Data type	any except STRING
	Range	–
	Unit	N/A
	Default	–
Pos	Description	Position in the source buffer
	Data type	DINT
	Range	[0,+65535]
	Unit	N/A
	Default	N/A
BigEndian	Description	TRUE if the frame is encoded with Big Endian format.
	Data type	BOOL
	Range	?
	Unit	?
	Default	?

5.30.8.2 Output

OK	Description	Returns true when the function successfully executes. See Function - General Rules.
	Data type	BOOL
	Unit	N/A
NextPos	Description	Position in the source buffer after the extracted data. 0 in case or error (invalid position / buffer size).
	Data type	DINT
	Unit	N/A

5.31 SerializeOut

PLCopen

Function - Copy the value of a variable to a binary frame.

5.31.1 Inputs

Input	Data Type	Range	Unit	Default	Description
BIGENDIAN	BOOL	TRUE, FALSE			TRUE if the frame is encoded with Big Endian format.
DATA	ANY(*)	N/A	N/A	No default	Source variable to be copied.
EN	BOOL	0, 1	N/A	No default	Execute the function.
FRAME	USINT	0,+65535	N/A	N/A	Destination buffer - must be an array.
POS	DINT	0,+65535	N/A	N/A	Position in the destination buffer.

(*) DATA cannot be a STRING.

5.31.2 Outputs

Output	Data Type	Range	Unit	Description
NEXTPOS	DINT		N/A	• Position in the destination buffer after the copied data. • 0 (zero) in case or error (e.g., invalid position or buffer size).
OK	BOOL		N/A	Returns true when the function successfully executes. See Function - General Rules.

5.31.3 Remarks

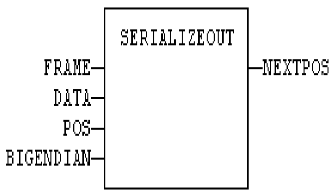
This function is used for building a communication frame in binary format.

- The **FRAME** input must be an array large enough to receive the data.
  - If the data cannot be safely copied to the destination buffer, the function returns 0 (zero).
- This function cannot be used to serialize STRING variables.
- The function returns the position in the destination frame, after the copied data.
  - The return value can be used as a position for the next serialization.

This function copies these number of bytes to the destination frame:

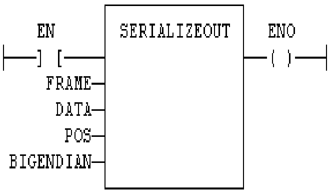
Bytes	Description
1 byte	BOOL, BYTE, SINT, and USINT variables.
2 bytes	INT, UINT, and WORD variables.
4 bytes	DINT, DWORD, REAL, and UDINT variables.
8 bytes	LINT and LREAL variables.

5.31.4 FBD Language



5.31.5 FFLD Language

- In the FFLD language, the operation is executed only if the input rung (EN) is TRUE.
  - The output rung (ENO) keeps the same value as the input rung.



5.31.6 IL Language

Not available.

5.31.7 ST Language

```
Q := SERIALIZEOUT (FRAME, DATA, POS, BIGENDIAN);
```

See Also

"SerializeIn" (➜ p. 293)

5.31.8 Arguments

5.31.8.1 Input

En	**Description**	Execute the function
	Data type	BOOL
	Range	[0,1]
	Unit	N/A
	Default	—
Frame[]	**Description**	Destination buffer - must be an array.
	Data type	USINT
	Range	[0,+65535]
	Unit	N/A
	Default	—
Data	**Description**	Source variable to be copied
	Data type	any except STRING
	Range	—
	Unit	N/A
	Default	—

Pos	**Description**	Position in the destination buffer
	Data type	DINT
	Range	[0,+65535]
	Unit	N/A
	Default	—
BigEndian	**Description**	TRUE if the frame is encoded with Big Endian format.
	Data type	BOOL
	Range	[0,1]
	Unit	N/A
	Default	—

5.31.8.2 Output

OK	**Description**	Returns true when the function successfully executes. See Function - General rules.
	Data type	BOOL
	Unit	N/A
NextPos	**Description**	Position in the destination buffer after the copied data. 0 in case or error (invalid position / buffer size).
	Data type	DINT
	Unit	N/A

5.32 SigID

PLCopen ✓

Function - Get the identifier of a Signal resource.

5.32.1 Inputs

`SIGNAL : STRING` Name of the signal resource - must be a constant value.

`COL : STRING` Name of the column within the signal resource - must be a constant value.

5.32.2 Outputs

`ID : DINT` ID of the signal - to be passed to other blocks.

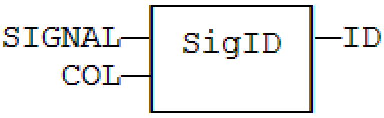
5.32.3 Remarks

- This function enables you to get the identifier of a signal defined as a resource.
- Some blocks have arguments that refer to a signal"signal" resource.
  - For all these blocks, the signal argument is materialized by a numerical identifier.

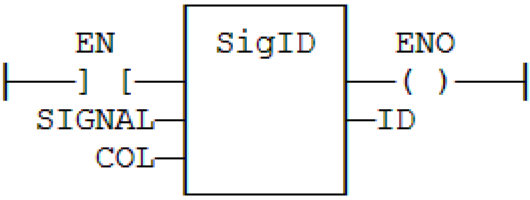
5.32.4 ST Language

```
ID := SigID ('MySignal', 'FirstColumn');
```

5.32.5 FBD Language



5.32.6 FFLD Language



5.32.7 IL Language

```
Op1: LD    'MySignal'
     SigID 'FirstColumn'
     ST ID
```

See Also

- "SigPlay" (→ p. 301)
- "SigScale" (→ p. 303)
- there is supposed to be a link to a Copa-Data Analog Signals Resources topic that we don't have - verify this CD topic is needed and link to it if so.

5.33 SigPlay

PLCopen ✓

Function block - Generate a signal defined in a resource.

5.33.1 Inputs

Input	Data Type	Description
IN	BOOL	Triggering command.
ID	DINT	ID of the signal resource, provided by the "SigID" (→ p. 299) function.
RST	BOOL	Reset command.
TM	TIME	Minimum duration between two changes of the output.

5.33.2 Outputs

Output	Data Type	Description
Q	BOOL	TRUE when the signal is finished.
OUT	REAL	Generated signal.
ET	TIME	Elapsed time.

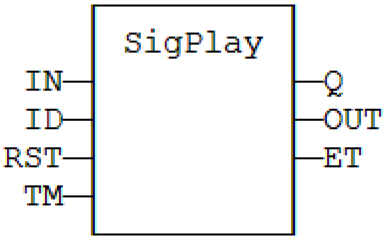
5.33.3 Remarks

- The ID argument is the identifier of the signal"signal" resource.
  - Use the "SigID" (→ p. 299) function to get this value.
- The IN argument is used as a Play / Pause command to play the signal.
  - The signal is not reset to the beginning when IN becomes FALSE.
  - Instead, use the RST input that resets the signal and forces the OUT output to 0 (zero).
- The TM input specifies the minimum amount of time in between two changes of the output signal.
  - This parameter is ignored if less than the cycle scan time.
- This function block includes its own timer.
  - Alternatively, use the "SigScale" (→ p. 303) function if you want to trigger the signal using a specific timer.

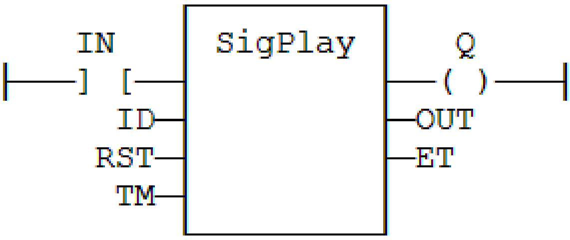
5.33.4 ST Language

```
MySig (II, ID, RST, TM);
Q := MySig.Q;
OUT := MySig.OUT;
ET := MySig.ET;
```

5.33.5 FBD Language



5.33.6 FFLD Language



5.33.7 IL Language

```
Op1: FFLD  IN
SigScale ID
ST       Q
```

See Also

- "SigID" (➔ p. 299)
- "SigScale" (➔ p. 303)
- there is supposed to be a link to a Copa-Data Analog Signals Resources topic that we don't have - verify this CD topic is needed and link to it if so.

5.34 SigScale

Function - Get a point from a Signal resource.

5.34.1 Inputs

`ID : DINT` ID of the signal resource, provided by the "SigID" (→ p. 299) function.

`IN : TIME` Time (X) coordinate of the wished point within the signal resource.

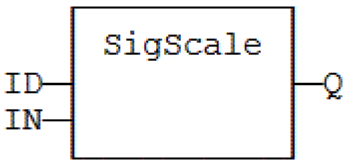
5.34.2 Outputs

`Q : REAL` Value (Y) coordinate of the point in the signal.

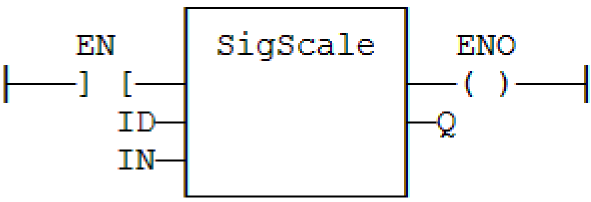
5.34.3 Remarks

- The ID argument is the identifier of the signal"signal" resource.
  - Use the "SigID" (→ p. 299) function to get this value.
- This function:
  - Converts a time value to a analog value such as defined in the signal resource.
  - Can be used instead of the "SigPlay" (→ p. 301) function block to trigger the signal using a specific timer.

5.34.4 FBD Language



5.34.5 FFLD Language



5.34.6 IL Language

```
Op1: LD        IN
SigScale ID
ST         Q
```

5.34.7 ST Language

```
Q := SigScale (ID, IN);
```

See Also

- "SigID" (➔ p. 299)
- "SigPlay" (➔ p. 301)
- there is supposed to be a link to a Copa-Data Analog Signals Resources topic that we don't have - verify this CD topic is needed and link to it if so.

5.35 stackint

PLCopen ✓

Function Block - Manages a stack of DINT integers.

5.35.1 Inputs

Input	Data Type	Range	Unit	Default	Description
PUSH	BOOL				Command: When changing from FALSE to TRUE, the value of IN is pushed on the stack.
POP	BOOL				Pop command: When changing from FALSE to TRUE, deletes the top of the stack.
R1	BOOL				Reset command: If TRUE, the stack is emptied and its size is set to N.
IN	DINT				Value to be pushed on a rising pulse of PUSH.
N	DINT				Maximum stack size - cannot exceed 128.

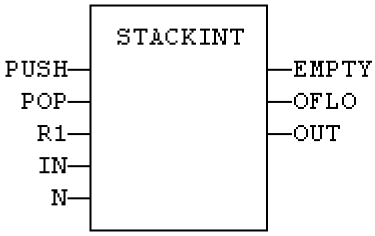
5.35.2 Outputs

Output	Data Type	Range	Unit	Description
EMPTY	BOOL			TRUE if the stack is empty.
OFLO	BOOL			TRUE if the stack is full.
OUT	DINT			Value at the top of the stack.

5.35.3 Remarks

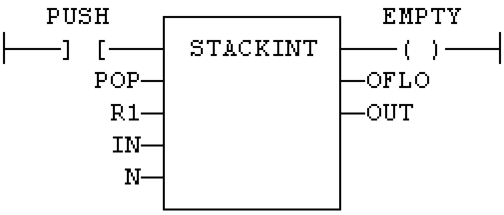
- Push and pop operations are performed on rising pulse of PUSH and POP inputs.
- The specified size (N) is taken into account only when the R1 (reset) input is TRUE.

5.35.4 FBD Language



5.35.5 FFLD Language

- In the FFLD language, the input rung is the PUSH command.
  - The output rung is the EMPTY output.



5.35.6 IL Language

```
(* MyStack is a declared instance of STACKINT function block *)
Op1: CAL MyStack (PUSH, POP, R1, IN, N)
     FFLD MyStack.EMPTY
     ST EMPTY
     FFLD MyStack.OFLO
     ST OFLO
     FFLD MyStack.OUT
     ST OUT
```

5.35.7 ST Language

```
(* MyStack is a declared instance of STACKINT function block *)
MyStack (PUSH, POP, R1, IN, N);
EMPTY := MyStack.EMPTY;
OFLO := MyStack.OFLO;
OUT := MyStack.OUT;
```

See Also

- "average / averageL" (➜ p. 232)
- "derivate" (➜ p. 234)
- "hyster" (➜ p. 249)
- "integral" (➜ p. 250)
- "lim_alrm" (➜ p. 254)

5.36 SurfLin

PLCopen ✓

Function Block - Linear interpolation on a surface.

5.36.1 Inputs

Input	Data Type	Range	Unit	Default	Description
X	REAL				X coordinate of the point to be interpolated.
XAxis	REAL[]				X coordinates of the known points of the X axis.
Y	REAL				Y coordinate of the point to be interpolated.
YAxis	REAL[]				Y coordinates of the known points of the Y axis.
ZVal	REAL[,] what does the [] and [,] mean				Z coordinate of the points defined by the axis.

5.36.2 Outputs

Output	Data Type	Range	Unit	Description
ERR	DINT			• Error code if failed. • 0 (zero) if OK.
OK	BOOL			TRUE if successful.
Z	REAL			Interpolated Z value corresponding to the X,Y input point.

5.36.3 Remarks

is this a function or function block?

This function performs linear surface interpolation in between a list of points defined in XAxis and YAxis single dimension arrays.

- The output Z value is an interpolation of the Z values of the four rounding points defined in the axis.
  - Z values of defined points are passed in the ZVal matrix (two dimension array).
  - ZVal dimensions must be understood as: ZVal [iX , iY]
- Values in X and Y axis must be sorted from the smallest to the biggest.
  - There must be at least two points defined in each axis.
  - ZVal must fit the dimension of XAxis and YAxis arrays.
    - For instance:
      - XAxis : ARRAY [0..2] of REAL;
      - YAxis : ARRAY [0.3] of REAL;
      - ZVal : ARRAY [0..2,0..3] of REAL;
- If the input point is outside the rectangle defined by XAxis and YAxis limits, the Z output is bound to the corresponding value and an error is reported.

The ERR output gives the cause of the error if the function fails:

Error Code	Meaning
0	OK
1	Invalid dimension of input arrays.
2	Invalid points for the X axis.
3	Invalid points for the Y axis.
4	X,Y point is out of the defined axis.

5.37 VLID

Function - Get the identifier (ID) of an embedded list of variables.

5.37.1 Inputs

Input	Data Type	Range	Unit	Default	Description
FILE	STRING				Pathname of the list file (.SPL or .TXT) - must be a constant value.

5.37.2 Outputs

Output	Data Type	Range	Unit	Description
ID	DINT			ID of the list - to be passed to other blocks.

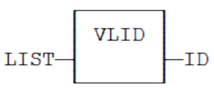
5.37.3 Remarks

- This function is used to create an Identifier (ID) or ListID for a list of application variables that are typically stored on the development PC.
- The list of application variables:
  - is a simple .TXT file.
  - can contain only one variable name per line
  - can be only global variables
- This function's ID output can be used as an input to "LogFileCSV" (→ p. 255).
  - It defines the application variables whose present value is recorded each time LogFileCSV is executed.

IMPORTANT

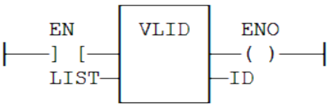
List files are read at compiling time and are embedded into the downloaded application code. This implies that a modification performed in the list file after downloading is not taken into account by the application.

5.37.4 FBD Language



5.37.5 FFLD Language

- The function is executed only if EN is TRUE.



5.37.6 IL Language

```
Op1: LD   'MyFile.txt'
VLID COL
ST   ListID
```

5.37.7 ST Language

```
ID := VLID ('MyFile.spl');
```

6 Support and Services

About KOLLMORGEN

Kollmorgen is a leading provider of motion systems and components for machine builders. Through world-class knowledge in motion, industry-leading quality and deep expertise in linking and integrating standard and custom products, Kollmorgen delivers breakthrough solutions that are unmatched in performance, reliability and ease-of-use, giving machine builders an irrefutable marketplace advantage.

Join the Kollmorgen Developer Network for product support. Ask the community questions, search the knowledge base for answers, get downloads, and suggest improvements.

North America
KOLLMORGEN
201 West Rock Road
Radford, VA 24141, USA

Web:	www.kollmorgen.com
Mail:	support@kollmorgen.com
Tel.:	+1 - 540 - 633 - 3545
Fax:	+1 - 540 - 639 - 4162

Europe
KOLLMORGEN Europe GmbH
Pempelfurtstr. 1
40880 Ratingen, Germany

Web:	www.kollmorgen.com
Mail:	technik@kollmorgen.com
Tel.:	+49 - 2102 - 9394 - 0
Fax:	+49 - 2102 - 9394 - 3155

South America
KOLLMORGEN
Avenida João Paulo Ablas, 2970
Jardim da Glória, Cotia - SP
CEP 06711-250, Brazil

Web:	www.kollmorgen.com
Mail:	contato@kollmorgen.com
Tel.:	+55 11 4615-6300

China and SEA
KOLLMORGEN
Room 302, Building 5, Lihpao Plaza,
88 Shenbin Road, Minhang District,
Shanghai, China.

Web:	www.kollmorgen.cn
Mail:	sales.china@kollmorgen.com
Tel.:	+86 - 400 668 2802
Fax:	+86 - 21 6248 5367