Crimson® 3.2 Function Block

Reference Guide | June 2020

LP1158 | Revision A

COPYRIGHT

SOFTWARE LICENSE

Software supplied with each Red Lion® product remains the exclusive property of Red Lion. Red Lion grants with each unit a perpetual license to use this software with the express limitations that the software may not be copied or used in any other product for any purpose. It may not be reverse engineered, or used for any other purpose other than in and with the computer hardware sold by Red Lion.

Red Lion Controls, Inc.
20 Willow Springs Circle
York, PA 17406

CONTACT INFORMATION:

AMERICAS

Inside US: +1 (877) 432-9908
Outside US: +1 (717) 767-6511
Hours: 8 am-6 pm Eastern Standard Time
(UTC/GMT -5 hours)

ASIA-PACIFIC

Shanghai, P.R. China: +86 21-6113-3688 x767
Hours: 9 am-6 pm China Standard Time
(UTC/GMT +8 hours)

EUROPE

Netherlands: +31 33-4723-225
France: +33 (0) 1 84 88 75 25
Germany: +49 (0) 1 89 5795-9421
UK: +44 (0) 20 3868 0909
Hours: 9 am-5 pm Central European Time
(UTC/GMT +1 hour)

Website: www.redlion.net
Support: support.redlion.net

Table of Contents

Preface

Disclaimer

This reference guide provides specific information on the various function blocks and operators available for use when writing control programs in the Control category of Crimson®.

While every effort has been made to ensure that this document is complete and accurate at the time of release, the information that it contains is subject to change. Red Lion Controls, Inc. is not responsible for any additions to or alterations of the original document. Industrial networks vary widely in their configurations, topologies, and traffic conditions. This document is intended as a general guide only. It has not been tested for all possible applications, and it may not be complete or accurate for some situations.

This guide is intended to be used by personnel responsible for configuring and commissioning Crimson devices for use in visualization, monitoring and control applications. Users of this document are urged to heed warnings and cautions used throughout the document.

Trademark Acknowledgments

Red Lion Controls, Inc. acknowledges and recognizes ownership of the following trademarked terms used in this document.

- Microsoft®, Windows®, and Windows 7™ are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

All other company and product names are trademarks of their respective owners.

Document History and Related Publications

The hard copy and electronic media versions of this document are revised only at major releases and therefore, may not always contain the latest product information. Tech Notes and/or product addendums will be provided as needed between major releases to describe any new information or document changes.

The latest online version of this document can be accessed through the Red Lion web site at https://www.redlion.net/red-lion-software/crimson/crimson-32.

Additional Product Information

Additional product information can be obtained by contacting the local sales representative or Red Lion through the contact numbers and/or support e-mail address listed on the inside of the front cover.

Chapter 1 Introduction

Crimson® 3.2 is the latest version of Red Lion's widely-acclaimed Crimson device configuration software. This Function Block Reference Guide augments the Crimson 3.2 Software Guide, and Crimson 3.2 Reference Guide by describing the various functions and operations available for use when writing IEC 61131-3 control programs within the Control category of Crimson 3.2. For each item, information is provided on the inputs, the output and the item's operation.

System Requirements

Crimson 3.2 is designed to run on any version of Microsoft Windows, from Windows 7 onwards. Memory requirements are modest and any system that meets the minimum system requirements for its operating system will be able to run Crimson 3.2. About 1GB of free disk space will be needed for installation, and you should ideally have a display with sufficient resolution to allow the editing of display pages without having to scroll.

Checking for Updates

If you have an Internet connection, you can use the Check for Update command in the Help menu to scan Red Lion's website for a new version of Crimson 3.2. If a later version than the one you are using is found, Crimson will ask if it should download the upgrade and update your software automatically. You may also manually download the upgrade from the Red Lion website by visiting the Downloads page within the Support section.

Getting Assistance

If you experience a problem or need assistance, the following resources are available.

Technical Support

Technical assistance is available on the web at: support.redlion.net
You may also call:
Inside US: +1 (877) 432-9908
Outside US: +1 (717) 767-6511

Online Forums

A number of online forums exist to support users of PLCs and HMIs. Red Lion recommends the Q&A forum at http://www.plctalk.net/qanda/. The discussion board is populated by many experts who are willing to help, and Red Lion's own technical support staff monitors this forum for questions relating to our products.

Chapter 2 Function Blocks and Operators

This chapter describes the various function blocks and operators available for use when writing IEC 61131-3 control programs within the Control category of Crimson 3.2. For each item, information is provided on the inputs, the output and the item's operation.

ABS ABSL

Function – Returns the absolute value of the input.

Inputs

IN : ANY value.

Outputs

Q : ANY Result: absolute value of IN.

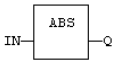
Remarks

In LD 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.

ST Language

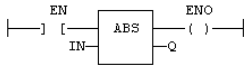
```
Q := ABS (IN);
```

FBD Language



LD Language

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



IL Language

```
Op1:    LD    IN
        ABS
        ST    Q (* Q is: ABS (IN) *)
```

See Also

TRUNC LOG POW SQRT

ACOS ACOSL

Function – Calculate an arc-cosine.

Inputs

IN : REAL/LREAL Real value.

Outputs

Q : REAL/LREAL Result: arc-cosine of IN.

Remarks

In LD 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.

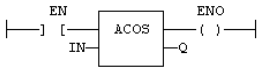
ST Language

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

FBD Language

LD Language

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



IL Language

```
Op1:    LD      IN
        ACOS
        ST      Q (* Q is: ACOS (IN) *)
```

See Also

SIN COS TAN ASIN ATAN ATAN2

+ ADD

Operator – Performs an addition of all inputs.

Inputs

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

Outputs

Q : ANY Result: IN1 + IN2.

Remarks

All inputs and the output must have the same type. In FBD language, the block may have up to 16 inputs. In LD language, the input rung (EN) enables the operation, and the output rung keeps the same value as the input rung. In 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.

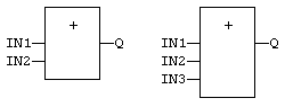
The addition can be used with strings. The result is the concatenation of the input strings.

ST Language

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

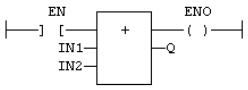
FBD Language

The block may have up to 16 inputs:



LD Language

The addition is executed only if EN is *TRUE*.
ENO is equal to EN.



IL Language

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

See Also

- SUB * MUL / DIV

ALARM_A

Function Block - Alarm with automatic reset.

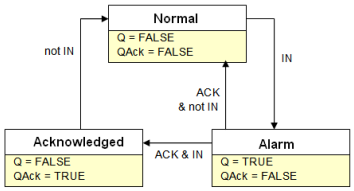
Inputs

IN : BOOL Process signal.
ACK : BOOL Acknowledge command.

Outputs

Q : BOOL *TRUE* if alarm is active.
QACK : BOOL *TRUE* if alarm is acknowledged.

Sequence



Remarks

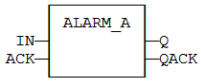
Combine this block with the LIM_ALRM block for managing analog alarms.

ST Language

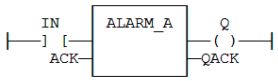
MyALARM is declared as an instance of ALARM_A function block.

```
MyALARM (IN, ACK, RST);
Q := MyALARM.Q;
QACK := MyALARM.QACK;
```

FBD Language



LD Language



IL Language

MyALARM is declared as an instance of ALARM_A function block.

```
Op1:    CAL   MyALARM (IN, ACK, RST)
        LD    MyALARM.Q
        ST    Q
        LD    MyALARM.QACK
        ST    QACK
```

See Also

ALARM_MLIM_ALRM

ALARM_M

Function Block – Alarm with manual reset.

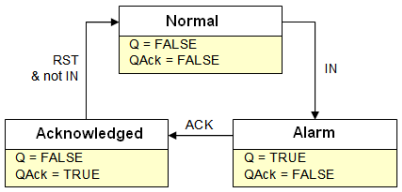
Inputs

IN : BOOL Process signal.
ACK : BOOL Acknowledge command.
RST : BOOL Reset command.

Outputs

Q : BOOL *TRUE* if alarm is active.
QACK : BOOL *TRUE* if alarm is acknowledged.

Sequence



Remarks

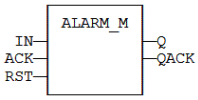
Combine this block with the LIM_ALRM block for managing analog alarms.

ST Language

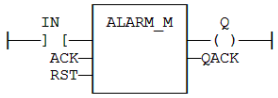
MyALARM is declared as an instance of ALARM_M function block.

```
MyALARM (IN, ACK, RST);
Q := MyALARM.Q;
QACK := MyALARM.QACK;
```

FBD Language



LD Language



IL Language

MyALARM is declared as an instance of ALARM_M function block.

```
Op1:    CALMyALARM (IN, ACK, RST)
LD      MyALARM.Q
ST      Q
LD      MyALARM.QACK
ST      QACK
```

See Also

ALARM_A LIM_ALRM

AND ANDN &

Operator – Performs a logical AND of all inputs.

Inputs

IN1 : BOOL First boolean input.
IN2 : BOOL Second boolean input.

Outputs

Q : BOOL Boolean AND of all inputs.

Truth table

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

Remarks

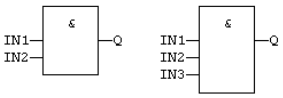
In FBD language, the block may have up to 16 inputs. The block is called "&" in FBD language. In LD language, an AND operation is represented by serialized contacts. In IL language, the AND instruction performs a logical AND between the current result and the operand. The current result must be boolean. The ANDN instruction performs an AND between the current result and the boolean negation of the operand. In ST and IL languages, "&" can be used instead of "AND".

ST Language

```
Q := IN1 AND IN2;
Q := IN1 & IN2 & IN3;
```

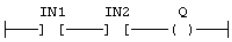
FBD Language

The block may have up to 16 inputs:



LD Language

Serialized contacts:



IL Language

```
Op1:    LD      IN1
        &       IN2 (* "&" or "AND" can be used *)
        ST      Q(* Q is equal to: IN1 AND IN2 *)
Op2:    LD      IN1
        AND     IN2
        &N      IN3(* "&N" or "ANDN" can be used *)
        ST      Q(* Q is equal to: IN1 AND IN2 AND (NOT IN3) *)
```

See Also

OR XOR NOT

ANY_TO_BOOL

Operator – Converts the input into boolean value.

Inputs

IN : ANY Input value.

Outputs

Q : BOOL Value converted to boolean.

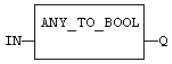
Remarks

For DINT, REAL and TIME input data types, the result is *FALSE* if the input is 0. The result is *TRUE* in all other cases. For STRING inputs, the output is *TRUE* if the input string is not empty, and *FALSE* if the string is empty. In LD language, the conversion is executed only if the input rung (EN) is *TRUE*. The output rung is the result of the conversion. In IL Language, the ANY_TO_BOOL function converts the current result.

ST Language

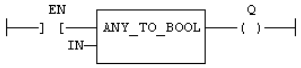
```
Q := ANY_TO_BOOL (IN);
```

FBD Language



LD Language

The conversion is executed only if EN is *TRUE*.
The output rung is the result of the conversion.
The output rung is *FALSE* if the EN is *FALSE*.



IL Language

```
Op1:    LD              IN
        ANY_TO_BOOL
        ST              Q
```

See Also

ANY_TO_SINT ANY_TO_INT ANY_TO_DINT ANY_TO_LINT
ANY_TO_REAL ANY_TO_LREAL ANY_TO_TIME ANY_TO_STRING

ANY_TO_DINT ANY_TO_UDINT

Operator – Converts the input into integer value.

Inputs

IN : ANY Input value.

Outputs

Q : DINT Value converted to integer.

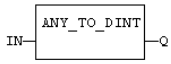
Remarks

For BOOL input data types, the output is 0 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 inputs, the output is the number represented by the string, or 0 if the string does not represent a valid number. In LD 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. In IL Language, the ANY_TO_DINT function converts the current result.

ST Language

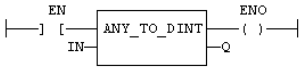
```
Q := ANY_TO_DINT (IN);
```

FBD Language



LD Language

The conversion is executed only if EN is *TRUE*.
ENO keeps the same value as EN.



IL Language

```
Op1:    LD              IN
        ANY_TO_DINT
        ST              Q
```

See Also

ANY_TO_BOOL ANY_TO_SINT ANY_TO_INT ANY_TO_LINT
ANY_TO_REAL ANY_TO_LREAL ANY_TO_TIME ANY_TO_STRING

ANY_TO_INT ANY_TO_UINT

Operator – Converts the input into 16 bit integer value.

Inputs

IN : ANY Input value.

Outputs

Q : INT Value converted to 16 bit integer.

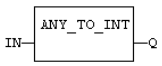
Remarks

For BOOL input data types, the output is 0 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 inputs, the output is the number represented by the string, or 0 if the string does not represent a valid number. In LD 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. In IL Language, the ANY_TO_INT function converts the current result.

ST Language

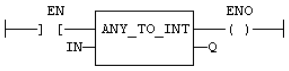
```
Q := ANY_TO_INT (IN);
```

FBD Language



LD Language

The conversion is executed only if EN is *TRUE*.
ENO keeps the same value as EN.



IL Language

```
Op1:    LD          IN
        ANY_TO_INT
        ST          Q
```

See Also

ANY_TO_BOOL ANY_TO_SINT ANY_TO_DINT ANY_TO_LINT ANY_TO_REAL
ANY_TO_LREAL ANY_TO_TIME ANY_TO_STRING

ANY_TO_LINT ANY_TO_ULINT

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

Inputs

IN : ANY Input value.

Outputs

Q : LINT Value converted to long (64 bit) integer.

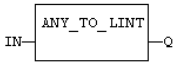
Remarks

For BOOL input data types, the output is 0 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 inputs, the output is the number represented by the string, or 0 if the string does not represent a valid number. In LD 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. In IL Language, the ANY_TO_LINT function converts the current result.

ST Language

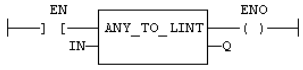
```
Q := ANY_TO_LINT (IN);
```

FBD Language



LD Language

The conversion is executed only if EN is *TRUE*.
ENO keeps the same value as EN.



IL Language

```
Op1:    LD              IN
        ANY_TO_LINT
        ST              Q
```

See Also

ANY_TO_BOOL ANY_TO_SINT ANY_TO_INT ANY_TO_DINT ANY_TO_REAL ANY_TO_LREAL
ANY_TO_TIME ANY_TO_STRING

ANY_TO_LREAL

Operator – Converts the input into double precision real value.

Inputs

IN : ANY Input value.

Outputs

Q : LREAL Value converted to double precision real.

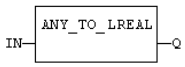
Remarks

For BOOL input data types, the output is 0.0 or 1.0. For DINT input data type, the output is the same number. For TIME input data types, the result is the number of milliseconds. For STRING inputs, the output is the number represented by the string, or 0.0 if the string does not represent a valid number. In LD 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. In IL Language, the ANY_TO_LREAL function converts the current result.

ST Language

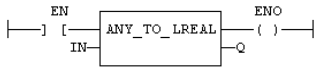
```
Q := ANY_TO_LREAL (IN);
```

FBD Language



LD Language

The conversion is executed only if EN is *TRUE*.
ENO keeps the same value as EN.



IL Language

```
Op1:    LD              IN
        ANY_TO_LREAL
        ST              Q
```

See Also

ANY_TO_BOOL ANY_TO_SINT ANY_TO_INT ANY_TO_DINT ANY_TO_LINT ANY_TO_REAL
ANY_TO_TIME ANY_TO_STRING

ANY_TO_REAL

Operator – Converts the input into real value.

Inputs

IN : ANY Input value.

Outputs

Q : REAL Value converted to real.

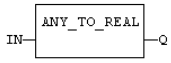
Remarks

For BOOL input data types, the output is 0.0 or 1.0. For DINT input data type, the output is the same number. For TIME input data types, the result is the number of milliseconds. For STRING inputs, the output is the number represented by the string, or 0.0 if the string does not represent a valid number. In LD 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. In IL Language, the ANY_TO_REAL function converts the current result.

ST Language

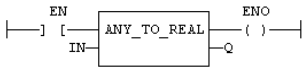
```
Q := ANY_TO_REAL (IN);
```

FBD Language



LD Language

The conversion is executed only if EN is *TRUE*.
ENO keeps the same value as EN.



IL Language

```
Op1:    LD              IN
        ANY_TO_REAL
        ST              Q
```

See Also

ANY_TO_BOOL ANY_TO_SINT ANY_TO_INT ANY_TO_DINT ANY_TO_LINT ANY_TO_LREAL
ANY_TO_TIME ANY_TO_STRING

ANY_TO_SINT ANY_TO_USINT

Operator – Converts the input into a small (8 bit) integer value.

Inputs

IN : ANY Input value.

Outputs

Q : SINT Value converted to a small (8 bit) integer.

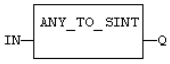
Remarks

For BOOL input data types, the output is 0 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 inputs, the output is the number represented by the string, or 0 if the string does not represent a valid number. In LD 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. In IL Language, the ANY_TO_SINT function converts the current result.

ST Language

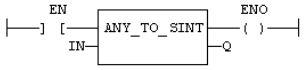
```
Q := ANY_TO_SINT (IN);
```

FBD Language



LD Language

The conversion is executed only if EN is *TRUE*.
ENO keeps the same value as EN.



IL Language

```
Op1:    LD          IN
        ANY_TO_SINT
        ST          Q
```

See Also

ANY_TO_BOOL ANY_TO_INT ANY_TO_DINT ANY_TO_LINT ANY_TO_REAL ANY_TO_LREAL
ANY_TO_TIME ANY_TO_STRING

ANY_TO_STRING

Operator – Converts the input into string value.

Inputs

IN : ANY Input value.

Outputs

Q : STRING Value converted to string.

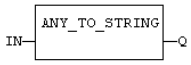
Remarks

For BOOL input data types, the output is 1 or 0 for *TRUE* and *FALSE* respectively. 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. In LD 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. In IL language, the ANY_TO_STRING function converts the current result.

ST Language

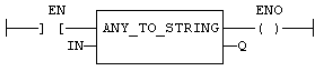
```
Q := ANY_TO_STRING (IN);
```

FBD Language



LD Language

The conversion is executed only if EN is *TRUE*.
ENO keeps the same value as EN.



IL Language

```
Op1:    LD              IN
        ANY_TO_STRING
        ST              Q
```

See Also

ANY_TO_BOOL ANY_TO_SINT ANY_TO_INT ANY_TO_DINT ANY_TO_LINT ANY_TO_REAL ANY_TO_LREAL ANY_TO_TIME

ANY_TO_TIME

Operator – Converts the input into time value.

Inputs

IN : ANY Input value.

Outputs

Q : TIME Value converted to time.

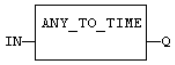
Remarks

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 inputs, the output is the time represented by the string, or t#0ms if the string does not represent a valid time. In LD 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. In IL Language, the ANY_TO_TIME function converts the current result.

ST Language

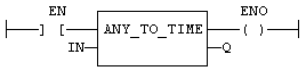
```
Q := ANY_TO_TIME (IN);
```

FBD Language



LD Language

The conversion is executed only if EN is *TRUE*.
ENO keeps the same value as EN.



IL Language

```
Op1:    LD              IN
        ANY_TO_TIME
        ST              Q
```

See Also

ANY_TO_BOOL ANY_TO_SINT ANY_TO_INT ANY_TO_DINT ANY_TO_LINT ANY_TO_REAL ANY_TO_LREAL ANY_TO_STRING

ASIN ASINL

Function – Calculate an arc-sine.

Inputs

IN : REAL/LREAL Real value.

Outputs

Q : REAL/LREAL Result: arc-sine of IN.

Remarks

In LD 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.

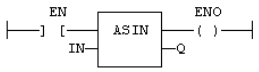
ST Language

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

FBD Language

LD Language

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



IL Language

```
Op1:    LD      IN
        ASIN
        ST      Q (* Q is: ASIN (IN) *)
```

See Also

SIN COS TAN ACOS ATAN ATAN2

ATAN ATANL

Function – Calculate an arc-tangent.

Inputs

IN : REAL/LREAL Real value.

Outputs

Q : REAL/LREAL Result: arc-tangent of IN.

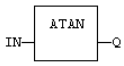
Remarks

In LD 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.

ST Language

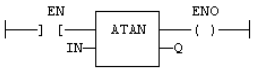
```
Q := ATAN (IN);
```

FBD Language



LD Language

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



IL Language

```
Op1:    LD      IN
        ATAN
        ST      Q (* Q is: ATAN (IN) *)
```

See Also

SIN COS TAN ASIN ACOS ATAN2

ATAN2 ATAN2L

Function – Calculate arc-tangent of Y/X.

Inputs

Y : REAL/LREAL Real value.
X : REAL/LREAL Real value.

Outputs

Q : REAL/LREAL Result: arc-tangent of Y / X.

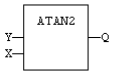
Remarks

In LD 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.

ST Language

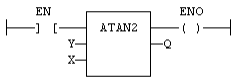
Q := ATAN2 (IN);

FBD Language



LD Language

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



See Also

SIN COS TAN ASIN ACOS ATAN

BLINK

Function Block – Blinker.

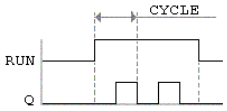
Inputs

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

Outputs

Q : BOOL Output blinking signal.

Time diagram



Remarks

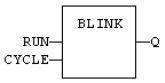
The output signal is *FALSE* when the RUN input is *FALSE*. The CYCLE input is the complete period of the blinking signal. In LD language, the input rung is the IN command. The output rung is the Q output signal.

ST Language

MyBlinker is a declared instance of BLINK function block.

```
MyBlinker (RUN, CYCLE);
Q := MyBlinker.Q;
```

FBD Language



LD Language

IL Language

MyBlinker is a declared instance of BLINK function block.

```
Op1:    CAL     MyBlinker (RUN, CYCLE)
        LD      MyBlinker.Q
        ST      Q
```

See Also

TONTOFTP

BLINKA

Function Block – Asymetric blinker.

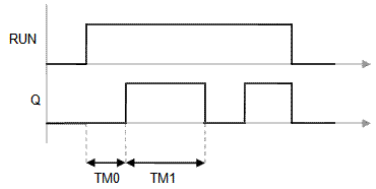
Inputs

RUN : BOOL Enabling command.
TM0 : TIME Duration of *FALSE* state on output.
TM1 : TIME Duration of *TRUE* state on output.

Outputs

Q : BOOL Output blinking signal.

Time diagram



Remarks

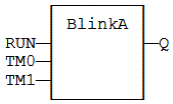
The output signal is *FALSE* when the RUN input is *FALSE*. In LD language, the input rung is the IN command. The output rung is the Q output signal.

ST Language

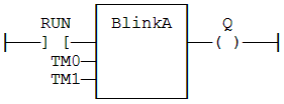
MyBlinker is a declared instance of BLINKA function block.

```
MyBlinker (RUN, TM0, TM1);
Q := MyBlinker.Q;
```

FBD Language



LD Language



IL Language

MyBlinker is a declared instance of BLINKA function block.

```
Op1:    CAL     MyBlinker (RUN, TM0, TM1)
        LD      MyBlinker.Q
        ST      Q
```

See Also

TONTOFTP

CMP

Function Block – Comparison with detailed outputs for integer inputs.

Inputs

IN1 : DINT First value.
IN2 : DINT Second value.

Outputs

LT : BOOL *TRUE* if IN1 < IN2.
EQ : BOOL *TRUE* if IN1 = IN2.
GT : BOOL *TRUE* if IN1 > IN2.

Remarks

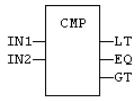
In LD language, the rung input (EN) validates the operation. The rung output is the result of LT (lower than comparison).

ST Language

MyCmp is declared as an instance of CMP function block:

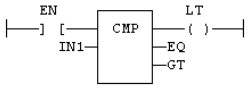
```
MyCMP (IN1, IN2);
bLT := MyCmp.LT;
bEQ := MyCmp.EQ;
bGT := MyCmp.GT;
```

FBD Language



LD Language

The comparison is performed only if EN is *TRUE*:



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
```

See Also

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

CONCAT

Function – Concatenate strings.

Inputs

IN_1 : STRING Any string variable or constant expression.
IN_N : STRING Any string variable or constant expression.

Outputs

Q : STRING Concatenation of all inputs.

Remarks

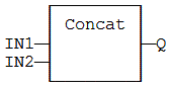
In FBD or LD language, the block may have up to 16 inputs. In IL or ST, the function accepts a variable number of inputs (at least 2).

Note that you also can use the "+" operator to concatenate strings.

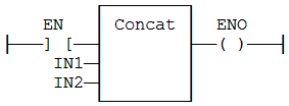
ST Language

```
Q := CONCAT ('AB', 'CD', 'E');(* now Q is 'ABCDE' *)
```

FBD Language



LD Language



IL Language

```
Op1:    LD          'AB'
        CONCAT      'CD'        'E'
        ST          Q (* Q is now 'ABCDE' *)
```

COS COSL

Function – Calculate a cosine.

Inputs

IN : REAL/LREAL Real value.

Outputs

Q : REAL/LREAL Result: cosine of IN.

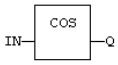
Remarks

In LD 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.

ST Language

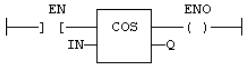
```
Q := COS (IN);
```

FBD Language



LD Language

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



IL Language

```
Op1:    LD      IN
        COS
        ST      Q       (* Q is: COS (IN) *)
```

See Also

SIN TAN ASIN ACOS ATAN ATAN2

CTD CTDr

Function Block – Down counter.

Inputs

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.

Outputs

Q : BOOL *TRUE* when counter is empty, i.e. when CV = 0.
CV : DINT Current value of the counter.

Remarks

The counter is empty (CV = 0) when the application starts. The counter does not include a pulse detection for CD input. Use R_TRIG or F_TRIG function block for counting pulses of CD input signal. In LD language, CD is the input rung. The output rung is the Q output.

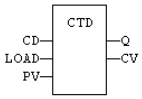
CTUr, CTDr, CTUDr function blocks operate exactly as other counters, except that all boolean inputs (CU, CD, RESET, LOAD) have an implicit rising edge detection included. Note that these counters may not be supported on some target systems.

ST Language

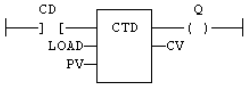
MyCounter is a declared instance of CTD function block.

```
MyCounter (CD, LOAD, PV);
Q := MyCounter.Q;
CV := MyCounter.CV;
```

FBD Language



LD Language



IL Language

MyCounter is a declared instance of CTD function block.

```
Op1:    CAL     MyCounter (CD, LOAD, PV)
        LD      MyCounter.Q
        ST      Q
        LD      MyCounter.CV
        ST      CV
```

See Also

CTU CTUD

CTU CTUr

Function Block – Up counter.

Inputs

CU : BOOL Enable counting. Counter is increased on each call when CU is *TRUE*.
RESET : BOOL Reset command. Counter is reset to 0 when called with RESET to *TRUE*.
PV : DINT Programmed maximum value.

Outputs

Q : BOOL *TRUE* when counter is full, i.e. when CV = PV.
CV : DINT Current value of the counter.

Remarks

The counter is empty (CV = 0) when the application starts. The counter does not include a pulse detection for CU input. Use R_TRIG or F_TRIG function block for counting pulses of CU input signal. In LD language, CU is the input rung. The output rung is the Q output.

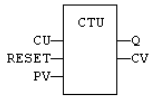
CTUr, CTDr, CTUDr function blocks operate exactly as other counters, except that all boolean inputs (CU, CD, RESET, LOAD) have an implicit rising edge detection included. Note that these counters may not be supported on some target systems.

ST Language

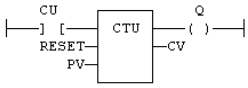
MyCounter is a declared instance of CTU function block.

```
MyCounter (CU, RESET, PV);
Q := MyCounter.Q;
CV := MyCounter.CV;
```

FBD Language



LD Language



IL Language

MyCounter is a declared instance of CTU function block.

```
Op1:    CAL     MyCounter (CU, RESET, PV)
        LD      MyCounter.Q
        ST      Q
        LD      MyCounter.CV
        ST      CV
```

See Also

CTD CTUD

CTUD CTUDr

Function Block – Up/down counter.

Inputs

CU : BOOL Enable counting. Counter is increased on each call when CU is *TRUE*.
CD : BOOL Enable counting. Counter is decreased on each call when CD is *TRUE*.
RESET : BOOL Reset command. Counter is reset to 0 called with RESET to *TRUE*.
LOAD : BOOL Re-load command. Counter is set to PV when called with LOAD to *TRUE*.
PV : DINT Programmed maximum value.

Outputs

QU : BOOL *TRUE* when counter is full, i.e. when CV = PV.
QD : BOOL *TRUE* when counter is empty, i.e. when CV = 0.
CV : DINT Current value of the counter.

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 R_TRIG or F_TRIG function blocks for counting pulses of CU or CD input signals. In LD language, CU is the input rung. The output rung is the QU output.

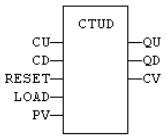
CTUr, CTDr, CTUDr function blocks operate exactly as other counters, except that all boolean inputs (CU, CD, RESET, LOAD) have an implicit rising edge detection included. Note that these counters may not be supported on some target systems.

ST Language

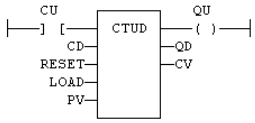
MyCounter is a declared instance of CTUD function block.

```
MyCounter (CU, CD, RESET, LOAD, PV);
QU := MyCounter.QU;
QD := MyCounter.QD;
CV := MyCounter.CV;
```

FBD Language



LD Language



IL Language

MyCounter is a declared instance of CTUD function block.

```
Op1:    CAL     MyCounter (CU, CD, RESET, LOAD, PV)
        LD      MyCounter.QU
        ST      QU
        LD      MyCounter.QD
        ST      QD
        LD      MyCounter.CV
        ST      CV
```

See Also

CTU CTD

DELETE

Function – Delete characters in a string.

Inputs

IN : STRING Character string.
NBC : DINT Number of characters to be deleted.
POS : DINT Position of the first deleted character (first character position is 1).

Outputs

Q : STRING Modified string.

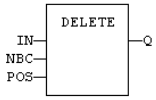
Remarks

The first valid character position is 1. In LD 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.

ST Language

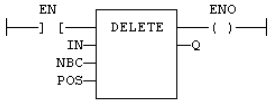
```
Q := DELETE (IN, NBC, POS);
```

FBD Language



LD Language

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



IL Language

```
Op1:    LD          IN
        DELETE      NBC, POS
        ST          Q
```

See Also

+ ADD MLEN INSERT FIND REPLACE LEFT RIGHT MID

/ DIV

Operator – Performs a division of inputs.

Inputs

IN1 : ANY_NUM First input.
IN2 : ANY_NUM Second input.

Outputs

Q : ANY_NUM Result: IN1 / IN2.

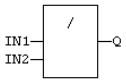
Remarks

All inputs and the output must have the same type. In LD language, the input rung (EN) enables the operation, and the output rung keeps the same value as the input rung. In 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.

ST Language

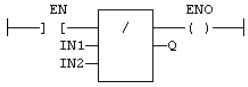
```
Q := IN1 / IN2;
```

FBD Language



LD Language

The division is executed only if EN is *TRUE*.
ENO is equal to EN.



IL Language

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

See Also

+ ADD NEG - * MUL

DTAt

Function Block – Generate a pulse at given date and time

Inputs

YEAR : DINT Wished year (e.g. 2006).
MONTH : DINT Wished month (1 = January).
DAY : DINT Wished day (1 to 31).
TMOFDAY : TIME Wished time.
RST : BOOL Reset command.

Outputs

QAT : BOOL Pulse signal.
QPAST : BOOL TRUE if elapsed.

Attention

Real Time clock may be not available on some targets. Please refer to OEM instructions for further details about available features.

Remarks

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*, and the output QAT is set to *TRUE* for one cycle only (pulse signal).

Highest units are ignored if set to 0. For instance, if arguments are year=0, month=0, day = 3, tmofday=t#10h then the block will trigger on the next 3rd day of the month at 10h.

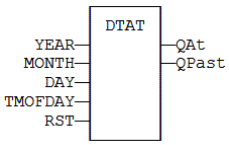
In LD language, the block is activated only if the input rung is *TRUE*.

ST Language

MyDTAT is a declared instance of DTAT function block.

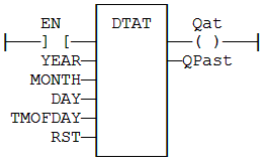
```
MyDTAT (YEAR, MONTH, DAY, TMOFDAY, RST);
QAT := MyDTAT.QAT;
QPAST := MyDTATA.QPAST;
```

FBD Language



LD Language

Called only if EN is *TRUE*.



IL Language

MyDTAT is a declared instance of DTAT function block.

```
Op1:    CAL     MyDTAT (YEAR, MONTH, DAY, TMOFDAY, RST)
        LD      MyDTAT.QAT
        ST      QAT
        LD      MyDTATA.QPAST
        ST      QPAST
```

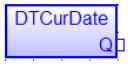
DTCurDate

Function: Get current date stamp

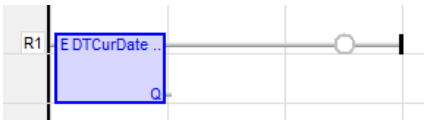
Q := DTCurDate ();

Q : DINT numerical stamp representing the current date.

FBD Language



LD Language



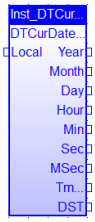
DTCurDateTime

Function: Get current time stamp (function block)

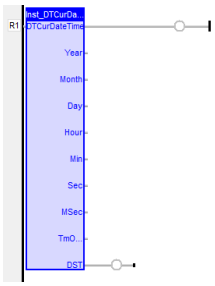
Inst_DTCurDateTime (bLocal);

bLocal :	BOOL *TRUE* if local time is requested (GMT if *FALSE*).
.Year :	DINT Output: current year
.Month :	DINT Output: current month
.Day :	DINT Output: current day
.Hour :	DINT Output: current time: hours
.Min :	DINT Output: current time: minutes
.Sec :	DINT Output: current time: seconds
.MSec :	DINT Output: current time: milliseconds
.TmOfDay :	TIME Output: current time of day (since midnight)
.DST :	BOOL Output: *TRUE* if Daylight Saving Time is active

FBD Language



LD Language



DTCurTime

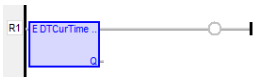
Function: get current time stamp

Q := DTCurTime ();

Q : DINT numerical stamp representing the current time of the day.

FBD Language

LD Language



DTDay

Function: Extract the day of the month from a date stamp

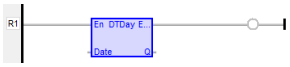
Q := DTDay (iDate);

IDATE : DINT numerical stamp representing a date.

Q : DINT day of the month of the date (1..31).

FBD Language

LD Language



DTFormat

Function – Format the current date/time to a string with a custom format

Inputs

FMT : STRING Format string.

Outputs

Q : STRING String containing formatted date or time.

Attention

Real Time clock may be not available on some targets. Please refer to OEM instructions for further details about available features.

Remarks

The format string may contain any character. Some special markers beginning with the '%' character indicates a date/time information:

%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)

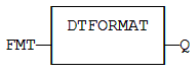
Example

```
(* let's say 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 *)
```

ST Language

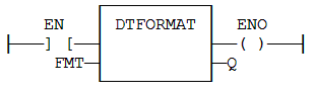
```
Q := DTFORMAT (FMT);
```

FBD Language



LD Language

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



IL Language

```
Op1:    LD FMT
        DTFORMAT
        ST Q
```

DTHour

Function: Extract the hours from a time stamp

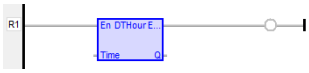
Q := DTHour (iTime);

ITIME : DINT numerical stamp representing a time.

Q : DINT Hours of the time (0..23).

FBD Language

LD Language



DTMin

Function: Extract the minutes from a time stamp

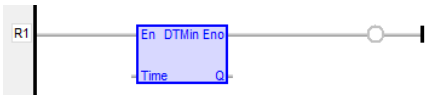
Q := DTMin (iTime);

ITIME : DINT numerical stamp representing a time.

Q : DINT minutes of the time (0..59).

FBD Language

LD Language



DTMonth

Function: Extract the month from a date stamp

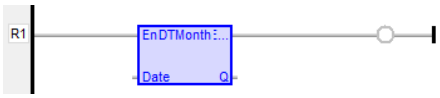
Q := DTMonth (iDate);

IDATE : DINT numerical stamp representing a date.

Q : DINT month of the date (1..12).

FBD Language

LD Language



DTMs

Function: Extract the milliseconds from a time stamp

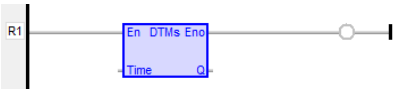
Q := DTMs (iTime);

ITIME : DINT numerical stamp representing a time.

Q : DINT Milliseconds of the time (0..999).

FBD Language

LD Language



DTSec

Function: Extract the seconds from a time stamp

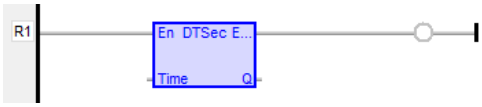
Q := DTSec (iTime);

ITIME : DINT numerical stamp representing a time.

Q : DINT Seconds of the time (0..59).

FBD Language

LD Language



DTYear

Function: Extract the year from a date stamp

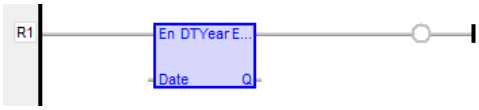
Q := DTYear (iDate);

IDATE : DINT numerical stamp representing a date.

Q : DINT year of the date (ex: 2004).

FBD Language

LD Language



= EQ

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

Inputs

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

Outputs

Q : BOOL *TRUE* if IN1 = IN2.

Remarks

Both inputs must have the same type. In LD language, the input rung (EN) enables the operation, and the output rung is the result of the comparison. In 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.

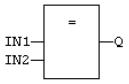
Comparisons can be used with strings. In that case, the lexical order is used for comparing the input strings. For instance, "ABC" is less than "ZX" ; "ABCD" is greater than "ABC".

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

ST Language

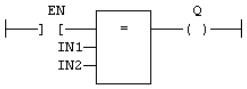
```
Q := IN1 = IN2;
```

FBD Language



LD Language

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



IL Language

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

See Also

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

EXP EXPL

Function – Calculates the natural exponential of the input.

Inputs

IN : REAL/LREAL Real value.

Outputs

Q : REAL/LREAL Result: natural exponential of IN.

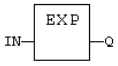
Remarks

In LD 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.

ST Language

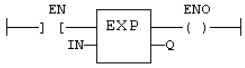
```
Q := EXP (IN);
```

FBD Language



LD Language

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



IL Language

```
Op1:    LD    IN
        EXP
        ST    Q (* Q is: EXP (IN) *)
```

See Also

ABS TRUNC POW SQRT

EXPT

Function – Calculates a power.

Inputs

IN : REAL Real value.
EXP : DINT Exponent.

Outputs

Q : REAL Result: IN at the 'EXP' power.

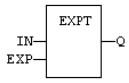
Remarks

In LD 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.

ST Language

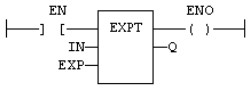
```
Q := EXPT (IN, EXP);
```

FBD Language



LD Language

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



IL Language

```
Op1:    LD      IN
        EXPT    EXP
        ST      Q (* Q is: (IN ** EXP) *)
```

See Also

ABS TRUNC LOG SQRT

F_TRIG

Function Block – Falling pulse detection.

Inputs

CLK : BOOL Boolean signal.

Outputs

Q : BOOL *TRUE* when the input changes from *TRUE* to *FALSE*.

Truth table

CLK	CLK prev	Q
0	0	0
0	1	1
1	0	0
1	1	0

Remarks

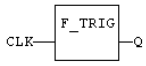
Although]P[an]N[contacts may be used in LD language, it is recommended to use declared instances of R_TRIG or F_TRIG function blocks in order to avoid unexpected behavior during an On Line change.

ST Language

MyTrigger is declared as an instance of F_TRIG function block:

```
MyTrigger (CLK);
Q := MyTrigger.Q;
```

FBD Language



LD Language

The input signal is the rung – the rung is the output:

IL Language

MyTrigger is declared as an instance of F_TRIG function block:

```
Op1:    CAL   MyTrigger (CLK)
        LD    MyTrigger.Q
        ST    Q
```

See Also

R_TRIG

FIND

Function – Find position of characters in a string.

Inputs

IN : STRING Character string.
STR : STRING String containing searched characters.

Outputs

POS : DINT Position of the first character of STR in IN, or 0 if not found.

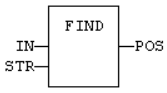
Remarks

The first valid character position is 1. A return value of 0 means that the STR string has not been found. Search is case sensitive. In LD 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.

ST Language

```
POS := FIND (IN, STR);
```

FBD Language



LD Language

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

IL Language

```
Op1:    LD      IN
        FIND    STR
        ST      POS
```

See Also

+ADD MLEN DELETE INSERT REPLACE LEFT RIGHT MID

>= GE

Operator – Test if first input is greater than or equal to second input.

Inputs

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

Outputs

Q : BOOL *TRUE* if IN1 >= IN2.

Remarks

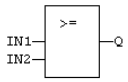
Both inputs must have the same type. In LD language, the input rung (EN) enables the operation, and the output rung is the result of the comparison. In IL language, the GE instruction performs the comparison between the current result and the operand. The current result and the operand must have the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

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".

ST Language

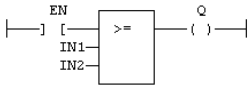
```
Q := IN1 >= IN2;
```

FBD Language



LD Language

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



IL Language

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

See Also

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

> GT

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

Inputs

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

Outputs

Q : BOOL *TRUE* if IN1 > IN2.

Remarks

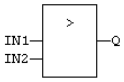
Both inputs must have the same type. In LD language, the input rung (EN) enables the operation, and the output rung is the result of the comparison. In IL language, the GT instruction performs the comparison between the current result and the operand. The current result and the operand must have the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

ST Language

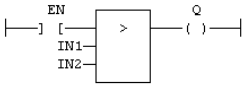
```
Q := IN1 > IN2;
```

FBD Language



LD Language

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



IL Language

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

See Also

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

GetSysInfo

Function – Returns system information.

Inputs

INFO : DINT Identifier of the requested information.

Outputs

Q : DINT Value of the requested information or 0 if error.

Remarks

The INFO parameter can be one of the following predefined values:

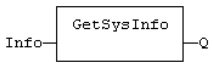
VALUE	DESCRIPTION
_SYSINFO_TRIGGER_MICROS	Programmed cycle time in micro-seconds.
_SYSINFO_TRIGGER_MS	Programmed cycle time in milliseconds.
_SYSINFO_CYCLETIME_MICROS	Duration of the previous cycle in micro-seconds.
_SYSINFO_CYCLETIME_MS	Duration of the previous cycle in milliseconds.
_SYSINFO_CYCLEMAX_MICROS	Maximum detected cycle time in micro-seconds.
_SYSINFO_CYCLEMAX_MS	Maximum detected cycle time in milliseconds.
_SYSINFO_CYCLESTAMP_MS	Time stamp of the current cycle in milliseconds (OEM dependent).
_SYSINFO_CYCLEOVERFLOWS	Number of detected cycle time overflows.
_SYSINFO_CYCLECOUNT	Counter of cycles.
_SYSINFO_APPVERSION	Version number of the application.
_SYSINFO_APPSTAMP	Compiling date stamp of the application.
_SYSINFO_CODECRC	CRC of the application code.
_SYSINFO_DATACRC	CRC of the application symbols.
_SYSINFO_FREEHEAP	Available space in memory heap (bytes).
_SYSINFO_DBSIZE	Space used in RAM (bytes).
_SYSINFO_ELAPSED	Seconds elapsed since startup.
_SYSINFO_CHANGE_CYCLE	Indicates a cycle just after an On Line Change.
_SYSINFO_WARMSTART	Non zero if RETAIN variables were loaded at the last start.
_SYSINFO_NBLOCKED	Number of locked variables.
_SYSINFO_NBBREAKPOINTS	Number of installed breakpoints.
_SYSINFO_BIGENDIAN	Non zero if the runtime processor is big endian.
_SYSINFO_DEMOAPP	Non zero if the application was compiled in DEMO mode.

In LD 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.

ST Language

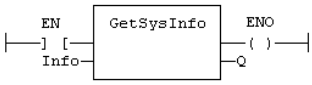
```
Q := GETSYSINFO (INFO);
```

FBD Language



LD Language

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



INSERT

Function – Insert characters in a string.

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).

Outputs

Q : STRING Modified string.

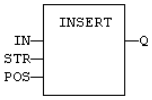
Remarks

The first valid character position is 1. In LD 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.

ST Language

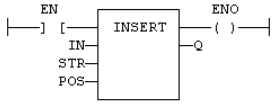
```
Q := INSERT (IN, STR, POS);
```

FBD Language



LD Language

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



IL Language

```
Op1:    LD          IN
        INSERTSTR,  POS
        ST          Q
```

See Also

+ ADD MLEN DELETE FIND REPLACE LEFT RIGHT MID

<= LE

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

Inputs

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

Outputs

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

Remarks

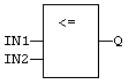
Both inputs must have the same type. In LD language, the input rung (EN) enables the operation, and the output rung is the result of the comparison. In IL language, the LE instruction performs the comparison between the current result and the operand. The current result and the operand must have the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

ST Language

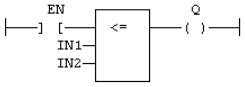
```
Q := IN1 <= IN2;
```

FBD Language



LD Language

The comparison is executed only if EN is *TRUE*:



IL Language

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

See Also

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

LEFT

Function – Extract characters of a string on the left.

Inputs

IN : STRING Character string.
NBC : DINT Number of characters to extract.

Outputs

Q : STRING String containing the first NBC characters of IN.

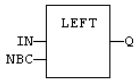
Remarks

In LD 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.

ST Language

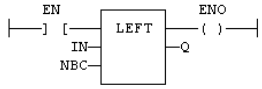
```
Q := LEFT (IN, NBC);
```

FBD Language



LD Language

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



IL Language

```
Op1:    LD      IN
        LEFT    NBC
        ST      Q
```

See Also

+ ADD MLEN DELETE INSERT FIND REPLACE RIGHT MID

LEN

Function – Get the number of characters in a string.

Inputs

IN : STRING Character string.

Outputs

NBC : INT Number of characters currently in the string. 0 if string is empty.

Remarks

In LD 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.

ST Language

```
NBC := LEN (IN);
```

LIMIT

Function – Bounds an integer between low and high limits.

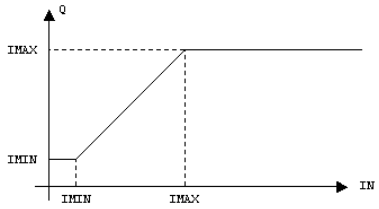
Inputs

IMIN : DINT Low bound.
IN : DINT Input value.
IMAX : DINT High bound.

Outputs

Q : DINT IMIN if IN < IMIN; IMAX if IN > IMAX; IN otherwise.

Function diagram



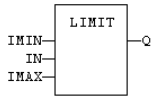
Remarks

In LD 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. Other inputs are operands of the function, separated by a coma.

ST Language

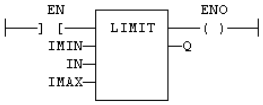
```
Q := LIMIT (IMIN, IN, IMAX);
```

FBD Language



LD Language

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



IL Language

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

See Also

MIN MAX MOD ODD

LN LNL

Function – Calculates the natural logarithm of the input.

Inputs

IN : REAL/LREAL Real value.

Outputs

Q : REAL/LREAL Result: natural logarithm of IN.

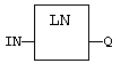
Remarks

In LD 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.

ST Language

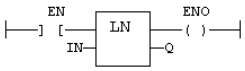
```
Q := LN (IN);
```

FBD Language



LD Language

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



IL Language

```
Op1:    LD    IN
        LN
        ST    Q (* Q is: LN (IN) *)
```

See Also

ABS TRUNC POW SQRT

LOG LOGL

Function – Calculates the logarithm (base 10) of the input.

Inputs

IN : REAL Real value.

Outputs

Q : REAL Result: logarithm (base 10) of IN.

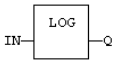
Remarks

In LD 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.

ST Language

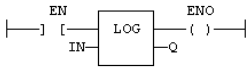
```
Q := LOG (IN);
```

FBD Language



LD Language

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



IL Language

```
Op1:    LD    IN
        LOG
        ST    Q (* Q is: LOG (IN) *)
```

See Also

ABS TRUNC POW SQRT

< LT

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

Inputs

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

Outputs

Q : BOOL *TRUE* if IN1 < IN2.

Remarks

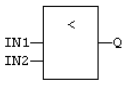
Both inputs must have the same type. In LD language, the input rung (EN) enables the operation, and the output rung is the result of the comparison. In IL language, the LT instruction performs the comparison between the current result and the operand. The current result and the operand must have the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

ST Language

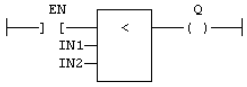
```
Q := IN1 < IN2;
```

FBD Language



LD Language

The comparison is executed only if EN is *TRUE*:



IL Language

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

See Also

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

MAX

Function – Get the maximum of two values.

Inputs

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

Outputs

Q : ANY IN1 if IN1 > IN2; IN2 otherwise.

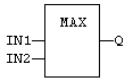
Remarks

In LD 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.

ST Language

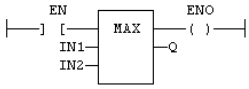
```
Q := MAX (IN1, IN2);
```

FBD Language



LD Language

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



IL Language

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

See Also

MIN LIMIT MOD ODD

MID

Function – Extract characters of a string at any position.

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).

Outputs

Q : STRING String containing the first NBC characters of IN.

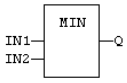
Remarks

The first valid position is 1. In LD 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 argument are operands of the function, separated by comas.

ST Language

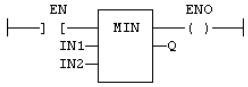
```
Q := MID (IN, NBC, POS);
```

FBD Language



LD Language

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



IL Language

```
Op1:    LD      IN
        MID     NBC, POS
        ST      Q
```

See Also

+ ADD MLEN DELETE INSERT FIND REPLACE LEFT RIGHT

MIN

Function – Get the minimum of two values.

Inputs

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

Outputs

Q : ANY IN1 if IN1 < IN2; IN2 otherwise.

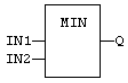
Remarks

In LD 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.

ST Language

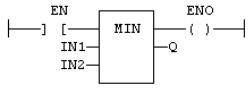
```
Q := MIN (IN1, IN2);
```

FBD Language



LD Language

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



IL Language

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

See Also

MAX LIMIT MOD ODD

MOD MODR MODLR

Function – Calculation of modulo.

Inputs

IN : DINT/REAL/LREAL Input value.
BASE : DINT/REAL/LREAL Base of the modulo.

Outputs

Q : DINT/REAL/LREAL Modulo: rest of the integer division (IN / BASE).

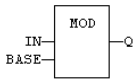
Remarks

In LD 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.

ST Language

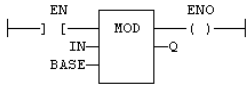
```
Q := MOD (IN, BASE);
```

FBD Language



LD Language

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



IL Language

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

See Also

MIN MAX LIMIT ODD

* MUL

Operator – Performs a multiplication of all inputs.

Inputs

IN1 : ANY_NUM First input.
IN2 : ANY_NUM Second input.

Outputs

Q : ANY_NUM Result: IN1 * IN2.

Remarks

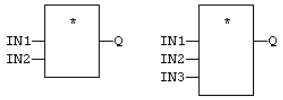
All inputs and the output must have the same type. In FBD language, the block may have up to 16 inputs. In LD language, the input rung (EN) enables the operation, and the output rung keeps the same value as the input rung. In IL language, the MUL instruction performs a multiplication between the current result and the operand. The current result and the operand must have the same type.

ST Language

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

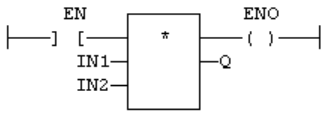
FBD Language

The block may have up to 16 inputs:



LD Language

The multiplication is executed only if EN is *TRUE*.
ENO is equal to EN.



IL Language

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

See Also

+ ADD - SUB / DIV

<> NE

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

Inputs

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

Outputs

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

Remarks

Both inputs must have the same type. In LD 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 type.

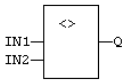
Comparisons can be used with strings. In that case, the lexical order is used for comparing the input strings. For instance, "ABC" is less than "ZX" ; "ABCD" is greater than "ABC".

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

ST Language

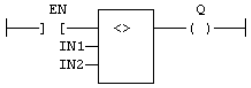
```
Q := IN1 <> IN2;
```

FBD Language



LD Language

The comparison is executed only if EN is *TRUE*:



IL Language

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

See Also

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

NEG -

Operator – Performs an integer negation of the input.

Inputs

IN : DINT Integer value.

Outputs

Q : DINT Integer negation of the input.

Truth table (examples)

IN	Q
0	0
1	-1
-123	123

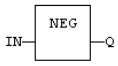
Remarks

In FBD and LD language, the block NEG can be used. In LD 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. This feature is not available in IL language. In ST language, "-" can be followed by a complex boolean expression between parenthesis.

ST Language

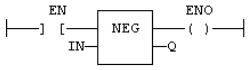
```
Q := -IN;
Q := - (IN1 + IN2);
```

FBD Language



LD Language

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



IL Language

Not available.

NOT

Operator – Performs a boolean negation of the input.

Inputs

IN : BOOL Boolean value.

Outputs

Q : BOOL Boolean negation of the input.

Truth table

IN	Q
0	1
1	0

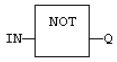
Remarks

In FBD language, the block NOT can be used. Alternatively, you can use a link terminated by a o negation. In LD language, negated contacts and coils can be used. In IL language, the N modifier can be used with instructions LD, AND, OR, XOR and ST. It represents a negation of the operand. In ST language, NOT can be followed by a complex boolean expression between parenthesis.

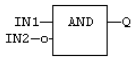
ST Language

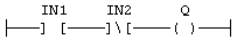
```
Q := NOT IN;
Q := NOT (IN1 OR IN2);
```

FBD Language



Explicit use of the NOT block:

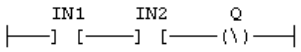




Use of a negated link: Q is IN1 AND NOT IN2:

LD Language

Negated contact: Q is: IN1 AND NOT IN2:



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

IL Language

```
Op1:    LDN  IN1
        OR   IN2
        ST   Q (* Q is equal to: (NOT IN1) OR IN2 *)
Op2:    LD   IN1
        AND  IN2
        STN  Q (* Q is equal to: NOT (IN1 AND IN2) *)
```

See Also

AND OR XOR

OR

Operator – Performs a logical OR of all inputs.

Inputs

IN1 : BOOL First boolean input.
IN2 : BOOL Second boolean input.

Outputs

Q : BOOL Boolean OR of all inputs.

Truth table

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

Remarks

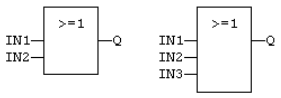
In FBD language, the block may have up to 16 inputs. The block is called >=1 in FBD language. In LD language, an OR operation is represented by contacts in parallel. In IL language, the OR instruction performs a logical OR between the current result and the operand. The current result must be boolean. The ORN instruction performs an OR between the current result and the boolean negation of the operand.

ST Language

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

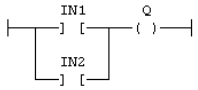
FBD Language

The block may have up to 16 inputs:



LD Language

Parallel contacts:



IL Language

```
Op1:    LD      IN1
        OR      IN2
        ST      Q (* Q is equal to: IN1 OR IN2 *)
Op2:    LD      IN1
        ORN     IN2
        ST      Q (* Q is equal to: IN1 OR (NOT IN2) *)
```

See Also

AND XOR NOT

PLS

Function Block – Pulse signal generator:

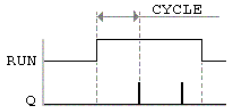
Inputs

RUN : BOOL Enabling command.
CYCLE : TIME Signal period.

Outputs

Q : BOOL Output pulse signal.

Time diagram



Remarks

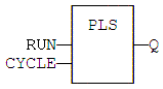
On every period, the output is set to *TRUE* during one cycle only. In LD language, the input rung is the IN command. The output rung is the Q output signal.

ST Language

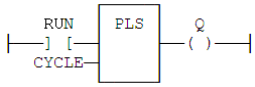
MyPLS is a declared instance of PLS function block:

```
MyPLS (RUN, CYCLE);
Q := MyPLS.Q;
```

FBD Language



LD Language



IL Language

MyPLS is a declared instance of PLS function block:

```
Op1:    CAL     MyPLS (RUN, CYCLE)
        LD      MyPLS.Q
        ST      Q
```

See Also

TON TOF TP

POW POWL

Function – Calculates a power.

Inputs

IN : REAL/LREAL Real value.
EXP : REAL/LREAL Exponent.

Outputs

Q : REAL/LREAL Result: IN at the 'EXP' power.

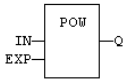
Remarks

Alternatively, in ST language, the ** operator can be used. In LD 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.

ST Language

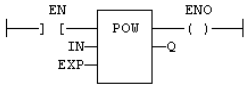
```
Q := POW (IN, EXP);
Q := IN ** EXP;
```

FBD Language



LD Language

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



IL Language

```
Op1:    LD      IN
        POW     EXP
        ST      Q (* Q is: (IN ** EXP) *)
```

See Also

ABS TRUNC LOG SQRT

QOR

Operator – Count the number of *TRUE* inputs.

Inputs

IN1 .. INn : BOOL Boolean inputs

Outputs

Q : DINT Number of inputs being *TRUE*

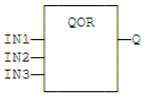
Remarks

The block accept a non-fixed number of inputs.

ST Language

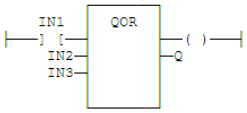
```
Q := QOR (IN1, IN2);
Q := QOR (IN1, IN2, IN3, IN4, IN5, IN6);
```

FBD Language



The block may have up to 16 inputs:

LD Language



The block may have up to 16 inputs:

IL Language

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

R_TRIG

Function Block – Rising pulse detection.

Inputs

CLK : BOOL Boolean signal.

Outputs

Q : BOOL *TRUE* when the input changes from *FALSE* to *TRUE*.

Truth table

CLK	CLK PREV	Q
0	0	0
0	1	0
1	0	1
1	1	0

Remarks

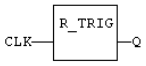
Although]P[an]N[contacts may be used in LD language, it is recommended to use declared instances of R_TRIG or F_TRIG function blocks in order to avoid unexpected behavior during an On Line change.

ST Language

MyTrigger is declared as an instance of R_TRIG function block:

```
MyTrigger (CLK);
Q := MyTrigger.Q;
```

FBD Language



LD Language

The input signal is the rung - the rung is the output:

IL Language

MyTrigger is declared as an instance of R_TRIG function block:

```
Op1:    CAL   MyTrigger (CLK)
        LD    MyTrigger.Q
        ST    Q
```

See Also

F_TRIG

REPLACE

Function – Replace characters in a string.

Inputs

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).

Outputs

Q : STRING Modified string.

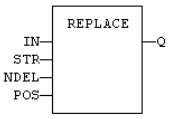
Remarks

The first valid character position is 1. In LD 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.

ST Language

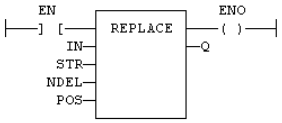
```
Q := REPLACE (IN, STR, NDEL, POS);
```

FBD Language



LD Language

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



IL Language

```
Op1:    LD        IN
        REPLACE   STR, NDEL, POS
        ST        Q
```

See Also

+ ADD MLEN DELETE INSERT FIND LEFT RIGHT MID

RIGHT

Function – Extract characters of a string on the right.

Inputs

IN : STRING Character string.
NBC : DINT Number of characters to extract.

Outputs

Q : STRING String containing the last NBC characters of IN.

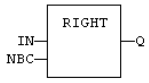
Remarks

In LD 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.

ST Language

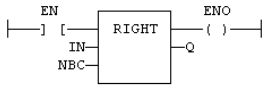
```
Q := RIGHT (IN, NBC);
```

FBD Language



LD Language

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



IL Language

```
Op1:    LD      IN
        RIGHT   NBC
        ST      Q
```

See Also

+ ADD MLEN DELETE INSERT FIND REPLACE LEFT MID

ROL

Function – Rotate bits of a register to the left.

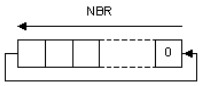
Inputs

IN : ANY register.
NBR : ANY Number of rotations (each rotation is 1 bit).

Outputs

Q : ANY Rotated register.

Diagram



Remarks

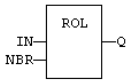
Arguments can be signed or unsigned integers from 8 to 32 bits.

In LD 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.

ST Language

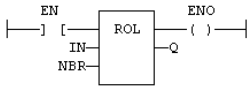
```
Q := ROL (IN, NBR);
```

FBD Language



LD Language

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



IL Language

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

See Also

SHL SHR ROR SHLb SHRb ROLb RORb SHLw SHRw ROLw RORw

ROOT

Function – Calculates the Nth root of the input.

Inputs

IN : REAL Real value
N : DINT Root level

Outputs

Q : REAL Result: Nth root of IN

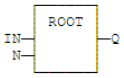
Remarks

In LD 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.

ST Language

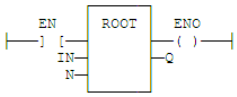
```
Q := ROOT (IN, N);
```

FBD Language



LD Language

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



IL Language

```
Op1:    LD      IN
        ROOT    N
        ST      Q (* Q is: ROOT (IN) *)
```

ROR

Function – Rotate bits of a register to the right.

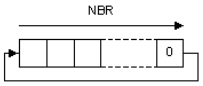
Inputs

IN : ANY register.
NBR : ANY Number of rotations (each rotation is 1 bit).

Outputs

Q : ANY Rotated register.

Diagram



Remarks

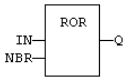
Arguments can be signed or unsigned integers from 8 to 32 bits.

In LD 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.

ST Language

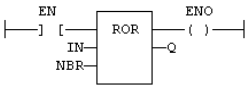
Q := ROR (IN, NBR);

FBD Language



LD Language

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



IL Language

Op1:	LD	IN
	ROR	NBR
	ST	Q

See Also

SHL SHR ROL SHLb SHRb ROLb RORb SHLw SHRw ROLw RORw

RS

Function Block – Reset dominant bistable.

Inputs

SET : BOOL Condition for forcing to *TRUE*.
RESET1 : BOOL Condition for forcing to *FALSE* (highest priority command).

Outputs

Q1 : BOOL Output to be forced.

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

Remarks

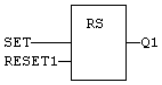
The output is unchanged when both inputs are *FALSE*. When both inputs are *TRUE*, the output is forced to *FALSE* (reset dominant).

ST Language

MyRS is declared as an instance of RS function block:

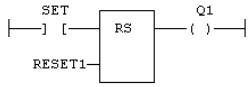
```
MyRS (SET, RESET1);
Q1 := MyRS.Q1;
```

FBD Language



LD Language

The SET command is the rung - the rung is the output:



IL Language

MyRS is declared as an instance of RS function block:

```
Op1:    CAL   MyRS (SET, RESET1)
        LD    MyRS.Q1
        ST    Q1
```

See Also

R S SR

ScaleLin

Function – Scaling – linear conversion.

Inputs

IN : REAL Real value.
IMIN : REAL Minimum input value.
IMAX : REAL Maximum input value.
OMIN : REAL Minimum output value.
OMAX : REAL Maximum output value.

Outputs

OUT : REAL Result: OMIN + IN * (OMAX - OMIN) / (IMAX - IMIN).

Truth table

INPUTS	OUT
IMIN >= IMAX	= IN
IN < IMIN	= OMIN
IN > IMAX	= OMAX
other	= OMIN + IN * (OMAX - OMIN) / (IMAX - IMIN)

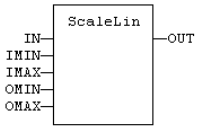
Remarks

In LD 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.

ST Language

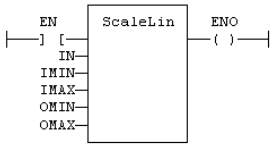
```
OUT := ScaleLin (IN, IMIN, IMAX, OMIN, OMAX);
```

FBD Language



LD Language

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



IL Language

```
Op1:    LD      IN
        ScaleLin IMAX, IMIN, OMAX, OMIN
        ST      OUT
```

SEL

Function – Select one of the inputs – 2 inputs.

Inputs

SELECT : BOOL Selection command
IN0 : ANY First input
IN1 : ANY Second input

Outputs

Q : ANY IN1 if SELECT is *FALSE*; IN2 if SELECT is *TRUE*

Truth table

SELECT	Q
0	IN0
1	IN1

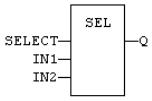
Remarks

In LD 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.

ST Language

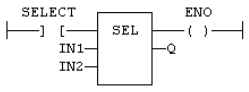
```
Q := SEL (SELECT, IN0, IN1);
```

FBD Language



LD Language

The input rung is the selector.
ENO has the same value as SELECT.



IL Language

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

See Also

MUX4 MUX8

SHL

Function – Shift bits of a register to the left.

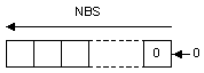
Inputs

IN : ANY register.
NBS : ANY Number of shifts (each shift is 1 bit).

Outputs

Q : ANY Shifted register.

Diagram



Remarks

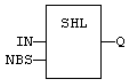
Arguments can be signed or unsigned integers from 8 to 32 bits.

In LD 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.

ST Language

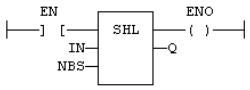
```
Q := SHL (IN, NBS);
```

FBD Language



LD Language

The shift is executed only if EN is *TRUE*.
ENO has the same value as EN.



IL Language

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

See Also

SHR ROL ROR SHLb SHRb ROLb RORb SHLw SHRw ROLw RORw

SHR

Function – Shift bits of a register to the right.

Inputs

IN : ANY register.
NBS : ANY Number of shifts (each shift is 1 bit).

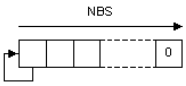
Outputs

Q : ANY Shifted register.

IMPORTANT!

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:



Remarks

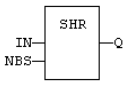
Arguments can be signed or unsigned integers from 8 to 32 bits.

In LD 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.

ST Language

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

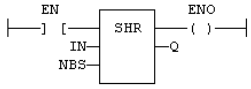
FBD Language



LD Language

The shift is executed only if EN is *TRUE*.

ENO has the same value as EN.



IL Language

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

See Also

SHL ROL ROR SHLb SHRb ROLb RORb SHLw SHRw ROLw RORw

SIN SINL

Function – Calculate a sine.

Inputs

IN : REAL/LREAL Real value.

Outputs

Q : REAL/LREAL Result: sine of IN.

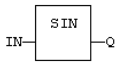
Remarks

In LD 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.

ST Language

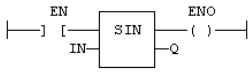
```
Q := SIN (IN);
```

FBD Language



LD Language

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



IL Language

```
Op1:    LD    IN
        SIN
        ST    Q (* Q is: SIN (IN) *)
```

See Also

COS TAN ASIN ACOS ATAN ATAN2

SQRT SQRTL

Function – Calculates the square root of the input.

Inputs

IN : REAL/LREAL Real value.

Outputs

Q : REAL/LREAL Result: square root of IN.

Remarks

In LD 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.

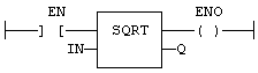
ST Language

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

FBD Language

LD Language

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



IL Language

```
Op1:    LD    IN
        SQRT
        ST    Q (* Q is: SQRT (IN) *)
```

See Also

ABS TRUNC LOG POW

SR

Function Block – Set dominant bistable.

Inputs

SET1 : BOOL Condition for forcing to *TRUE* (highest priority command).
RESET : BOOL Condition for forcing to *FALSE*.

Outputs

Q1 : BOOL Output to be forced.

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

Remarks

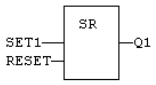
The output is unchanged when both inputs are *FALSE*. When both inputs are *TRUE*, the output is forced to *TRUE* (set dominant).

ST Language

MySR is declared as an instance of SR function block:

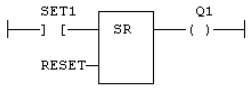
```
MySR (SET1, RESET);
Q1 := MySR.Q1;
```

FBD Language



LD Language

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



IL Language

MySR is declared as an instance of SR function block:

```
Op1:    CAL   MySR (SET1, RESET)
        LD    MySR.Q1
        ST    Q1
```

See Also

R S RS

- SUB

Operator – Performs a subtraction of inputs.

Inputs

IN1 : ANY_NUM / TIME First input.
IN2 : ANY_NUM / TIME Second input.

Outputs

Q : ANY_NUM / TIME Result: IN1 - IN2.

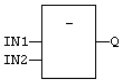
Remarks

All inputs and the output must have the same type. In LD language, the input rung (EN) enables the operation, and the output rung keeps the same value as the input rung. In 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.

ST Language

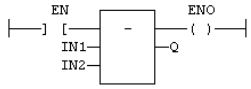
```
Q := IN1 - IN2;
```

FBD Language



LD Language

The subtraction is executed only if EN is *TRUE*.
ENO is equal to EN.



IL Language

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

See Also

+ ADD * MUL / DIV

TAN TANL

Function – Calculate a tangent.

Inputs

IN : REAL/LREAL Real value.

Outputs

Q : REAL/LREAL Result: tangent of IN.

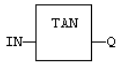
Remarks

In LD 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.

ST Language

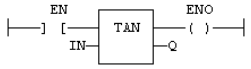
```
Q := TAN (IN);
```

FBD Language



LD Language

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



IL Language

```
Op1:    LD    IN
        TAN
        ST    Q (* Q is: TAN (IN) *)
```

See Also

SIN COS ASIN ACOS ATAN ATAN2

TMD

Function Block – Down-counting stop watch.

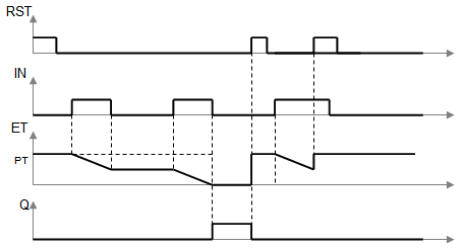
Inputs

IN : BOOL The time counts when this input is *TRUE*.
RST : BOOL Timer is reset to PT when this input is *TRUE*.
PT : TIME Programmed time.

Outputs

Q : BOOL Timer elapsed output signal.
ET : TIME Elapsed time.

Time diagram



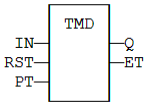
Remarks

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*.

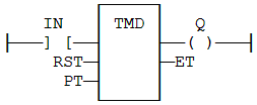
ST Language

```
MyTimer is a declared instance of TMD function block.
MyTimer (IN, RST, PT);
Q := MyTimer.Q;
ET := MyTimer.ET;
```

FBD Language



LD Language



IL Language

```
MyTimer is a declared instance of TMD function block.
Op1:    CAL   MyTimer (IN, RST, PT)
        LD    MyTimer.Q
        ST    Q
        LD    MyTimer.ET
        ST    ET
```

See Also

TMU

TMU

Function Block – Up-counting stop watch.

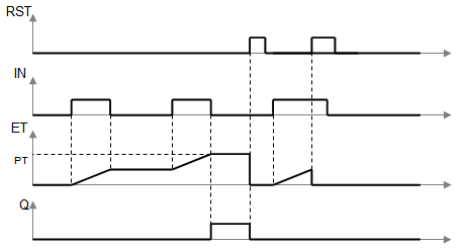
Inputs

IN : BOOL The time counts when this input is *TRUE*.
RST : BOOL Timer is reset to 0 when this input is *TRUE*.
PT : TIME Programmed time. (TMU)
PTsec : UDINT Programmed time. (TMUsec - seconds)

Outputs

Q : BOOL Timer elapsed output signal.
ET : TIME Elapsed time. (TMU)
ETsec : UDINT Elapsed time. (TMU - seconds)

Time diagram



Remarks

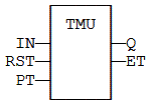
The timer counts up when the IN input is *TRUE*. It stops when the programmed time is elapsed. The timer is reset when the RST input is *TRUE*. It is not reset when IN is *FALSE*.

ST Language

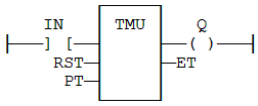
MyTimer is a declared instance of TMU function block.

```
MyTimer (IN, RST, PT);
Q := MyTimer.Q;
ET := MyTimer.ET;
```

FBD Language



LD Language



IL Language

MyTimer is a declared instance of TMU function block.

```
Op1:    CAL   MyTimer (IN, RST, PT)
        LD    MyTimer.Q
        ST    Q
        LD    MyTimer.ET
        ST    ET
```

See Also

TMD

TOF TOFR

Function Block – Off timer.

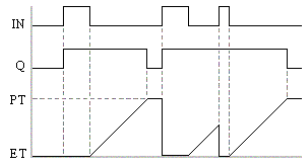
Inputs

IN : BOOL Timer command.
PT : TIME Programmed time.
RST : BOOL Reset (TOFR only).

Outputs

Q : BOOL Timer elapsed output signal.
ET : TIME Elapsed time.

Time diagram



Remarks

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. The output signal is set to *TRUE* on when the IN input rises to *TRUE*, reset to *FALSE* when programmed time is elapsed.

TOFR is same as TOF but has an extra input for resetting the timer.

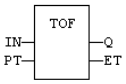
In LD language, the input rung is the IN command. The output rung is Q the output signal.

ST Language

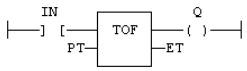
MyTimer is a declared instance of TOF function block.

```
MyTimer (IN, PT);
Q := MyTimer.Q;
ET := MyTimer.ET;
```

FBD Language



LD Language



IL Language

MyTimer is a declared instance of TOF function block.

```
Op1:    CAL   MyTimer (IN, PT)
        LD    MyTimer.Q
        ST    Q
        LD    MyTimer.ET
        ST    ET
```

See Also

TON TP BLINK

TON

Function Block – On timer.

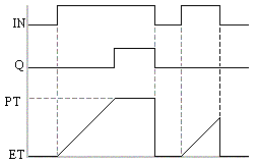
Inputs

IN : BOOL Timer command.
PT : TIME Programmed time.

Outputs

Q : BOOL Timer elapsed output signal.
ET : TIME Elapsed time.

Time diagram



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. The output signal is set to *TRUE* when programmed time is elapsed, and reset to *FALSE* when the input command falls.

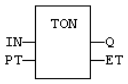
In LD language, the input rung is the IN command. The output rung is Q the output signal.

ST Language

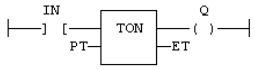
MyTimer is a declared instance of TON function block.

```
MyTimer (IN, PT);
Q := MyTimer.Q;
ET := MyTimer.ET;
```

FBD Language



LD Language



IL language

MyTimer is a declared instance of TON function block.

```
Op1:    CAL   MyTimer (IN, PT)
        LD    MyTimer.Q
        ST    Q
        LD    MyTimer.ET
        ST    ET
```

See Also

TOF TP BLINK

TP TPR

Function Block – Pulse timer.

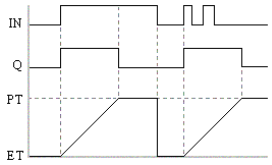
Inputs

IN : BOOL Timer command.
PT : TIME Programmed time.
RST : BOOL Reset (TPR only).

Outputs

Q : BOOL Timer elapsed output signal.
ET : TIME Elapsed time.

Time diagram



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, 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.

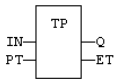
TPR is same as TP but has an extra input for resetting the timer.

In LD language, the input rung is the IN command. The output rung is Q the output signal.

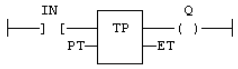
ST Language

```
MyTimer is a declared instance of TP function block.
MyTimer (IN, PT);
Q := MyTimer.Q;
ET := MyTimer.ET;
```

FBD Language



LD Language



IL Language

```
MyTimer is a declared instance of TP function block.
Op1:    CAL   MyTimer (IN, PT)
        LD    MyTimer.Q
        ST    Q
        LD    MyTimer.ET
        ST    ET
```

See Also

TON TOF BLINK

TRUNC TRUNCL

Function – Truncates the decimal part of the input.

Inputs

IN : REAL/LREAL Real value.

Outputs

Q : REAL/LREAL Result: integer part of IN.

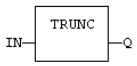
Remarks

In LD 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.

ST Language

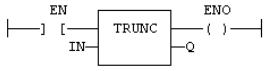
```
Q := TRUNC (IN);
```

FBD Language



LD Language

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



IL Language

```
Op1:    LD      IN
        TRUNC
        ST      Q (* Q is the integer part of IN *)
```

See Also

ABS LOG POW SQRT

XOR

Operator – Performs an exclusive OR of all inputs.

Inputs

IN1 : BOOL First boolean input.
IN2 : BOOL Second boolean input.

Outputs

Q : BOOL Exclusive OR of all inputs.

Truth table

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

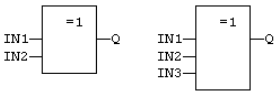
Remarks

The block is called =1 in FBD and LD languages. In IL language, the XOR instruction performs an exclusive OR between the current result and the operand. The current result must be Boolean.

ST Language

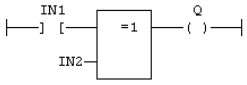
```
Q := IN1 XOR IN2;
Q := IN1 XOR IN2 XOR IN3;
```

FBD Language



LD Language

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



IL Language

```
Op1:    LD      IN1
        XOR     IN2
        ST      Q (* Q is equal to: IN1 XOR IN2 *)
```

See Also

AND OR NOT