Chapter 8 Function/Function block libraries

8. Function/Function block libraries

8.1 Function libraries

This chapter describes function libraries.

Point Please refer to below description when the function error occurs.

□ Function error

When the error occurs during the function, ENO will be 0 and error flag(_ERR, _LER) will be 1.
ENO of the function without error outputs EN input. EN and ENO are used only in LD(Ladder Diagram).

□ Error flag

_ERR (Error)

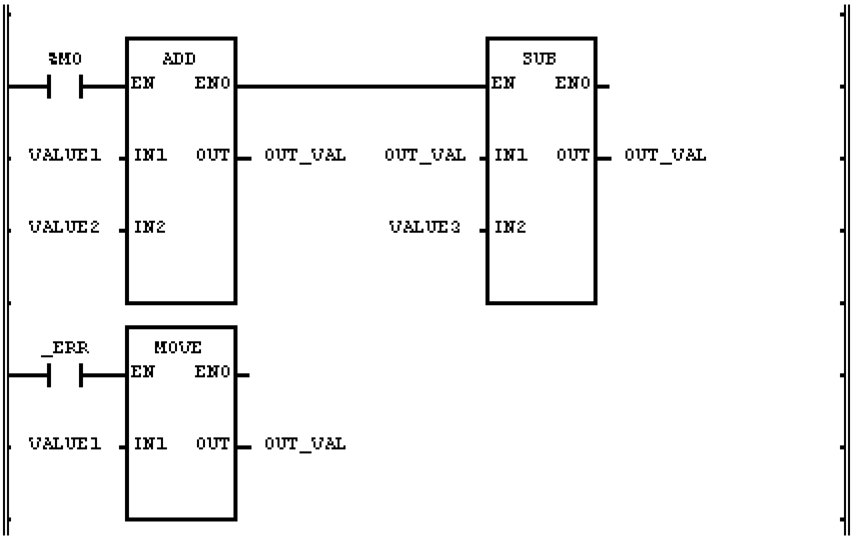
- _ERR value will be changed as below after operating the function marking no error.
  (The function marking no error maintains _ERR status before operation.)
- For the operation error, _ERR value will be 1.
- Except the operation error, _ERR value will be 0.

_LER (Latched Error)

- _LER will be 1 for the error after operation and will be maintained till the current program block is completed.
- 0 can be writable by program.

■ Program example

Program that does not execute SUB function while ADD function error and stores VALUE1 to OUT_VAL.



(1) If two inputs of function(ADD) are as below, the function error occurs.

Input(IN1) : VALUE1(SINT) = 100(16#64)
(IN2) : VALUE2(SINT) = 50(16#32)
Output(OUT) : OUT_VAL(SINT) = -106(16#96)

(2) The output exceeds the range of output data type and OUT_VAL(SINT) stores abnormal value.
ENO of function(ADD) will be 0 and the function(SUB) is not executed and the error flag _ERR and _LER will be on.

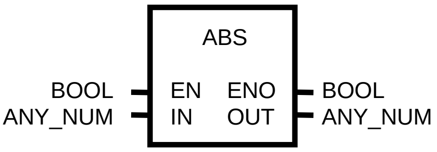
(3) _ERR is on and the function(MOVE) will be executed.

Input(IN1): VALUE1(SINT) = 100(16#64)
Output(OUT): OUT_VAL(SINT) = 100(16#64)

ABS

Absolute value operation

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
	**Input** EN : Execute the function in case of 1 IN : Input value of absolute operation **Output** ENO : Output 1 in case of no error OUT : Absolute value IN and OUT shall be same data type.

■ Function

Convert IN value to absolute and output to OUT.

X of absolute□X□will be

□X□= X if X>=0,

□X□= -X if X<0.

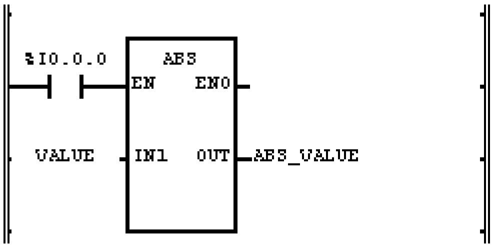
OUT = □IN□

■ Error

If IN value is lower limit of minus value of given data type, _ERR and _LER flag will be set.

Ex) If the data type is SINT and IN value is -128, it is error.

■ Program example

LD	IL
	LD %I0.0.0 JMPN AL LD VALUE ABS ST ABS_VALUE AL :

(1) If the execution condition(% I0.0.0) is On, the function ABS is executed.

(2) If VALUE = -7, ABS_VALUE = |-7| = 7.
If VALUE = 200, ABS_VALUE = |200| = 200.

Input(IN) : VALUE(INT) = -7

1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1

(16#FFF9)

↓ (ABS)

Output(OUT) : ABS_VALUE (INT) = 7

0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

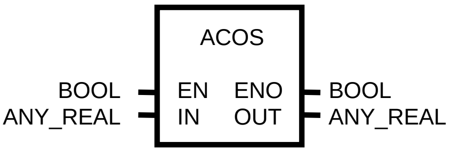
(16#0007)

Note Negative expression of INT type is described by 2□S Complement form(Refer to 3.2.4. Data type structure)

ACOS

Arc Cosine operation

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●			

Function	Description
 ACOS BOOL ANY_REAL EN IN ENO OUT BOOL ANY_REAL	**Input** EN : Execute the function in case of 1 IN : Input value of Arc Cosine operation **Output** ENO : Output 1 in case of no error OUT : Radian value of the result IN and OUT shall be same data type.

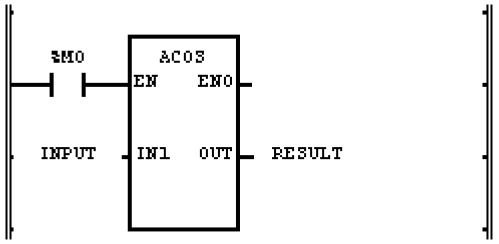
Function

Calculate IN's Arc Cosine and output to OUT. The output value will be between 0 and □.
OUT = ACOS (IN)

Error

If IN1 exceeds the range from -1.0 to 1.0, _ERR and _LER flag is set.

Program example

LD	IL
	LD %M0 JMPN LL LD INPUT ACOS ST RESULT LL :

(1) If the execution condition(%M0) is On, Arc Cosine operation function ACOS is executed.

(2) If INPUT variable is 0.8660 .... ($\sqrt{3}$ /2), the result will be 0.5235 ... (□/6 rad = 30°).

ACOS($\sqrt{3}$ /2) = □/6

(COS □/6 = ($\sqrt{3}$ /2)

Input(IN1) : INPUT(REAL) = 0.866

↓ (ACOS)

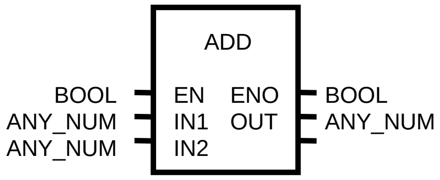
Output(OUT) : RESULT (REAL) = 5.23499966E-01

Note Expression of REAL type mark is based on IEEE Standard 754-1984 (Refer to 3.2.4. Data type structure)

ADD

Add

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
 ADD BOOL ANY_NUM ANY_NUM EN IN1 IN2 ENO OUT BOOL ANY_NUM	**Input** EN : Execute the function in case of 1 IN1 : Augend IN2 : Addend Can be extended to 8 inputs **Output** ENO : Output 1 in case of no error OUT : ADD Result Variables connected to IN1, IN2, ..., OUT shall be same data type.

■ Function

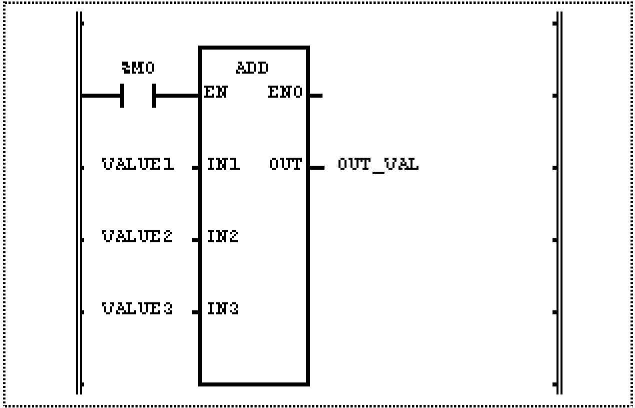
Add IN1, IN2,..., INn (n: input number) and output to OUT.
OUT = IN1 + IN2 + ... + INn

■ Error

If the output exceeds the range of given data type, _ERR and _LER flag will be set.

■ Program example

LD



IL

```
LD                  %M0
JMPN                CA
LD                  VALUE1
ADD       IN1:=     CURRENT RESULT
          IN2:=     VALUE2
          IN3:=     VALUE3
ST               OUT_VAL
CA :
```

(1) When the execution condition(%M0) is On, ADD function is executed.

(2) If VALUE1 = 300, VALUE2 = 200, VALUE3 = 100,
OUT_VAL = 300 + 200 + 100 = 600.

Input(IN1) : VALUE1(INT) = 300(16#012C)

0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0

+ (ADD)

(IN2) : VALUE2(INT) = 200(16#00C8)

0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0

+ (ADD)

(IN2) : VALUE3(INT) = 100(16#0064)

0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0

↓

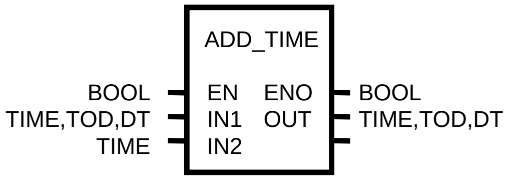
Output(OUT) : OUT_VAL(INT) = 600(16#0258)

0	0	0	0	0	0	1	0	0	1	0	1	1	0	0	0

ADD_TIME

Add time

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
 ADD_TIME BOOL TIME,TOD,DT TIME EN IN1 IN2 ENO OUT BOOL TIME,TOD,DT	**Input** EN : Execute the function in case of 1 IN1 : Reference time, time of day or date IN2 : Time to be added **Output** ENO : Output 1 in case of no error OUT : Add result of day or time or date OUT type depends on input IN1. If IN1 type is TIME_OF_DAY, OUT type will be also TIME_OF_DAY.

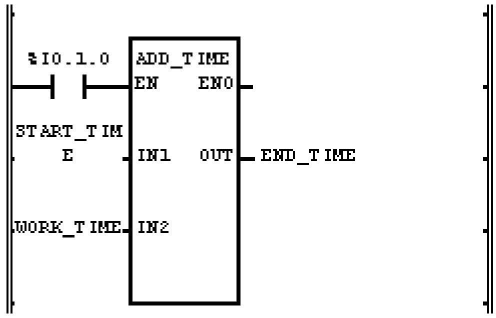
■ Function

- □ If IN1 is TIME, added TIME will be output.
- □ If IN1 is TIME_OF_DAY, add the TIME to reference TIME_OF_DAY and output the TIME_OF_DAY.
- □ If IN1 is DATE_AND_TIME, add the TIME to reference DATE_OF_TIME and output the DATE_AND_TIME.

■ Error

- □ If the output exceeds the range of given data type, _ERR and _LER flag will be set.
- □ If the result of adding TIME exceeds the range of TIME data type, T#49D17H2M47S295MS, the result of adding TOD and TIME exceeds 24 hours or the result of adding, DT and time exceeds 2083 YEAR, it will be error.

■ Program example

LD	IL
 %I0.1.0 ADD_TIME EN ENO START_TIME IN1 OUT END_TIME WORK_TIME IN2	LD %I0.1.0 JMPN ABC LD START_TIME ADD_TIME IN1:= CURRENT RESULT IN2:= WORK_TIME ST END_TIME ABC :

(1) If the execution condition(%I0.1.0) is On, time ADD function, ADD_TIME, is executed.

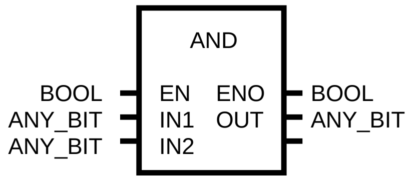
(2) If START_TIME is TOD#08:30:00 and WORK_TIME is T#2H10M20S500MS, TOD#10:40:20.5 will be output to END_TIME.

Input(IN1) : START_TIME(TOD) = TOD#08:30:00
+ (ADD_TIME)
(IN2) : WORK_TIME(TIME) = T#2H10M20S500MS
↓
Output(OUT) : END_TIME(TOD) = TOD#10:40:20.5

AND

Logical AND

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

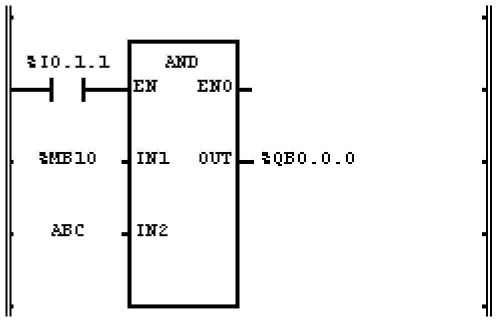
Function	Description
 AND BOOL ANY_BIT ANY_BIT EN IN1 IN2 ENO OUT BOOL ANY_BIT	**Input** EN : Execute the function in case of 1 IN1 : Input1 IN2 : Input2 Can be extended to 8 inputs. **Output** ENO : Output EN value itself OUT : AND result IN1, IN2 and OUT shall be same type.

■ Function

Execute AND IN1 to IN2 by bit and output the result to OUT.

IN1	1111 0000
&	
IN2	1010 1010
OUT	1010 0000

■ Program example

LD	IL
 %I0.1.1 AND EN ENO %MB10 IN1 OUT %QB0.0.0 ABC IN2	LD %I0.1.1 JMPN AA LD %MB10 AND IN1:= CURRENT RESULT IN2:= ABC ST %QB0.0.0 AA :

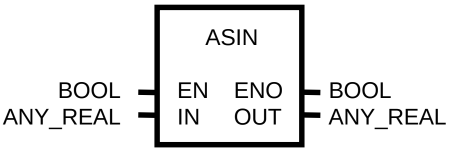
(1) If the execution condition(%I0.1.1) is On, the function AND is executed.
(2) The AND result of INI= %MB10 and IN2 = ABC is output to OUT = %QB0.0.0.

Input(IN1) : %MB10 (BYTE) = 16#CC	1	1	0	0	1	1	0	0
	& (AND)							
(IN2) : ABC(BYTE) = 16#F0	1	1	1	1	0	0	0	0
	↓							
Output(OUT) : %QB0.0.0(BYTE) = 16#C0	1	1	0	0	0	0	0	0

ASIN

Arc Sine operation

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●			

Function	Description
 ASIN BOOL ANY_REAL EN IN ENO OUT BOOL ANY_REAL	**Input** EN : Execute the function in case of 1 IN : Input value of Arc Sine operation **Output** ENO : Output 1 in case of no error OUT : Radian result of operation result IN and OUT shall be same type.

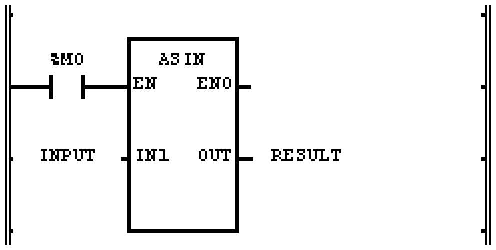
■ Function

Output IN's Arc Sine value to OUT. The output value is between -□/2 to □/2 .
OUT = ASIN (IN)

■ Error

If the input value exceeds the range from -1.0 to 1.0, _ERR and _LER flag is set.

■ Program example

LD	IL
	LD %M0 JMPN AAA LD INPUT ASIN ST RESULT AAA :

(1) If the execution condition(%M0) is On, Arc Sine operation function ASIN is executed.

(2) If INPUT variable is 0.8660 .... ($\sqrt{3}$ /2), RESULT declared as output variable will be 1.0471 .... (□/3 rad = 60°).

ASIN ($\sqrt{3}$ / 2) = □/3

(SIN(□/3) = ($\sqrt{3}$ /2)

Input(IN1) : INPUT(REAL) = 0.866

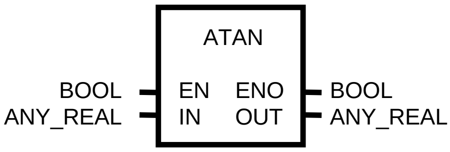
↓ (ASIN)

Output(OUT) : RESULT(REAL) = 1.04714680E+00

ATAN

Arc Tangent operation

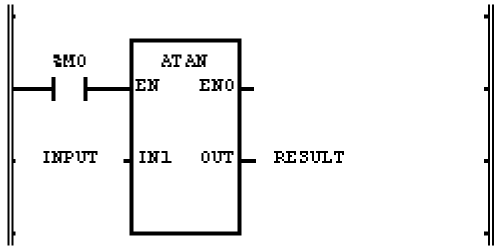
Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●			

Function	Description
 ATAN BOOL ANY_REAL EN IN ENO OUT BOOL ANY_REAL	**Input** EN : Execute the function in case of 1 IN : Input value of Arc Tangent operation **Output** ENO : Output EN value itself OUT : Radian output value of operation result IN and OUT shall be same type.

■ Function

Output IN's Arc Tangent value to OUT. The output value is between -□/2 and □/2.
OUT = ATAN (IN)

■ Program example

LD	IL
	LD %M0 JMPN AA LD INPUT ATAN ST RESULT AA :

(1) If the execution condition(%M0) is On, Arc Tangent operation function ASIN is executed.

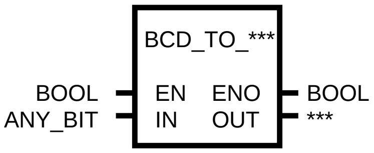
(2) If INPUT variable is 1.0, RESULT declared as output variable will be □/4 = 0.7853 ....
ATAN (1) = □/4
(TAN(□/4) = 1)

Input(IN1) : INPUT(REAL) = 1.0
↓ (ATAN)
Output(OUT) : RESULT(REAL) =7.85398185E-01

BCD_TO_***

Convert BCD type to integer

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
 BCD_TO_*** BOOL ANY_BIT EN IN ENO OUT BOOL ***	**Input** EN : Execute the function in case of 1 IN : ANY_BIT input with BCD type data **Output** ENO : Output EN value itself OUT : Type converted data

■ Function

Convert IN to OUT data type.

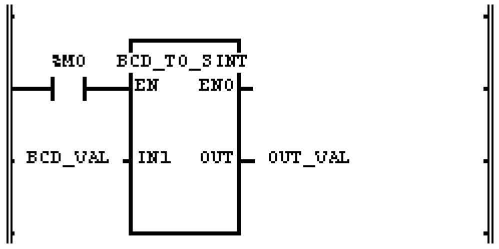
FUNCTION	Input type	Output type	Description
BCD_TO_SINT	BYTE	SINT	Convert BCD to output data type. Normal conversion is executed only if the input is BCD value. (If input data type is WORD, 0～16#9999 value is normally converted.)
BCD_TO_INT	WORD	INT	
BCD_TO_DINT	DWORD	DINT	
BCD_TO_LINT	LWORD	LINT	
BCD_TO_USINT	BYTE	USINT	
BCD_TO_UINT	WORD	UINT	
BCD_TO_UDINT	DWORD	UDINT	
BCD_TO_ULINT	LWORD	ULINT	

■ Error

If IN is not BCD type data, the output will be 0 and _ERR and _LER flag is set.

■ Program example

LD



IL

```
LD              %M0
JMPN            ABC
LD              BCD_VAL
BCD_TO_SINT
ST          OUT_VAL
ABC :
```

(1) If the execution condition(%M0) is On, the function BCD_TO_*** is executed.

(2) If BCD_VAL(BYTE type) = 16#22(2#0010_ 0010), OUT_VAL(SINT type) = 22(2#0001_ 0110) declared as output variable will be output.

Input(IN1) :BCD_VAL(BYTE) = 16#22

0	0	1	0	0	0	1	0

↓ (BCD_TO_SINT)

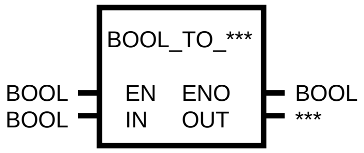
Output(OUT): OUT_VAL(SINT) = 22

0	0	0	1	0	1	1	0

BOOL_TO_***

BOOL type conversion	

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
	**Input** EN : Execute the function in case of 1 IN : Bit to be converted(1 bit) **Output** ENO : Output EN value itself OUT : Type converted data

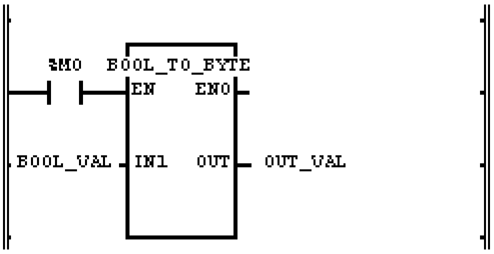
■ Function

Convert IN to OUT data type and output.

FUNCTION	Output type	Description
BOOL_TO_SINT	SINT	If BOOL input is 2#0, output integer '0' and if it is 2#1, output integer '1', according to output data type.
BOOL_TO_INT	INT	
BOOL_TO_DINT	DINT	
BOOL_TO_LINT	LINT	
BOOL_TO_USINT	USINT	
BOOL_TO_UINT	UINT	
BOOL_TO_UDINT	UDINT	
BOOL_TO_ULINT	ULINT	
BOOL_TO_BYTE	BYTE	Convert BOOL to output data type filling upper bit with 0.
BOOL_TO_WORD	WORD	
BOOL_TO_DWORD	DWORD	
BOOL_TO_LWORD	LWORD	
BOOL_TO_STRING	STRING	Convert BOOL to STRING type. Convert it to '0' or '1'.

■ Program example

LD



IL

```
LD              %M0
JMPN            ABC
LD              BOOL_VAL
BOOL_TO_BYTE
ST           OUT_VAL
ABC :
```

(1) If the execution condition(%M0) is On, the function BOOL_TO_*** is executed.

(2) If BOOL_VAL(BOOL type) = 2#1, OUT_VAL(BYTE type) = 2#0000_ 0001 declared as output variable will be output.

Input(IN1) : BOOL_VAL(BOOL) = 2#1

1

(BOOL_TO_SINT)↓

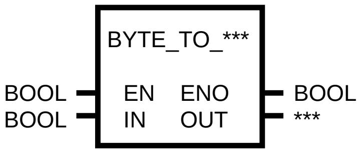
Output(OUT): OUT_VAL(BYTE) = 16#1

0	0	0	0	0	0	0	1

BYTE_TO_***

BYTE type conversion

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

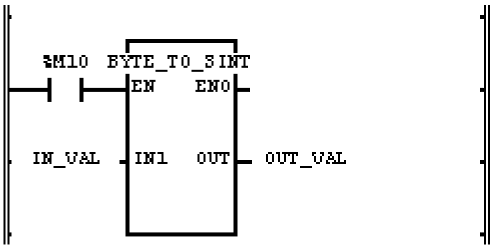
Function	Description
 BYTE_TO_*** BOOL BOOL EN IN ENO OUT BOOL ***	**Input** EN : Execute the function in case of 1 IN : Bit string to be converted(8bit) **Output** ENO : Output EN value itself OUT : Type converted data

■ Function

Convert IN to OUT data type and output.

FUNCTION	Output type	Description
BYTE _TO_SINT	SINT	Convert internal bit array to SINT type without conversion.
BYTE _TO_INT	INT	Convert INT to output data type filling upper bit with 0.
BYTE _TO_DINT	DINT	Convert DINT to output data type filling upper bit with 0.
BYTE _TO_LINT	LINT	Convert LINT to output data type filling upper bit with 0.
BYTE _TO_USINT	USINT	Convert internal bit array to USINT type without conversion.
BYTE _TO_UINT	UINT	Convert UINT to output data type filling upper bit with 0.
BYTE _TO_UDNT	UDINT	Convert UDINT to output data type filling upper bit with 0.
BYTE _TO_ULINT	ULINT	Convert ULINT to output data type filling upper bit with 0.
BYTE _TO_BOOL	BOOL	Convert lower 1 bit to BOOL type.
BYTE _TO_WORD	WORD	Fill upper bit with 0 to convert it to WORD type.
BYTE _TO_DWORD	DWORD	Fill upper bit with 0 to convert it to DWORD type.
BYTE _TO_LWORD	LWORD	Fill upper bit with 0 to convert it to LWORD type.
BYTE _TO_STRING	STRING	Convert input value to STRING type.

■ Program example

LD	IL
 %M10 BYTE_TO_SINT EN ENO IN_VAL IN1 OUT OUT_VAL	LD %M10 JMPN LLL LD IN_VAL BYTE_TO_SINT ST OUT_VAL LLL :

(1) If the execution condition(%M10) is On, the function BYTE_TO_SINT is executed.

(2) If IN_VAL(BYTE type) = 2#0001_1000, OUT_VAL(SINT type) = 24(2#0001_1000).

Input(IN1) : IN_VAL(BYTE) = 16#18

0	0	0	1	1	0	0	0

↓ (BYTE_TO_SINT)

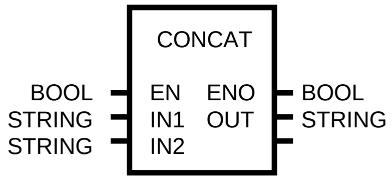
Output(OUT) : OUT_VAL(SINT) = 24

0	0	0	1	1	0	0	0

CONCAT

Character string concatenation

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
	**Input** EN : Execute the function in case of 1 IN1 : Character string input IN2 : Character string input Can be extended to 8 inputs. **Output** ENO : Output 1 in case of no error OUT : Character string output

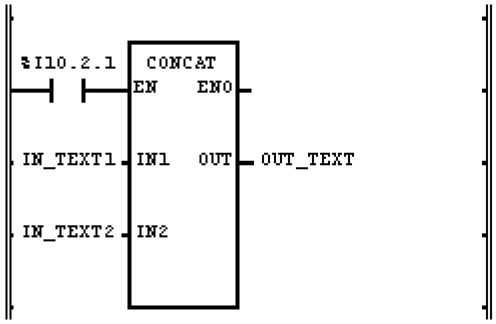
Function

Concatenates input character string in order of IN1, IN2, IN3,...., INn(n: input number) and outputs to the output character string OUT.

Error

If (character sum of all input character string) > 30, just 30 characters of concatenated each input character strings are output and _ERR and _LER flag is set.

Program example

LD	IL
	LD %0.2.1 JMPN THERE LD IN_TEXT1 CONCAT IN1:= CURRENT RESULT IN2:= IN_TEXT2 ST OUT_TEXT THERE :

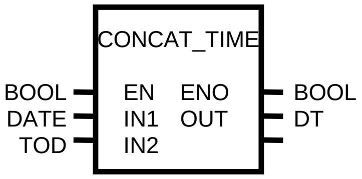
(1) If the execution condition(%I0.2.1) is On, the function CONCAT is executed.
(2) If IN_TEXT1=`ABCD` and IN_TEXT2=`DEF`, OUT_TEXT=`ABCDDEF`.

Input(IN1) : IN_TEXT1(STRING) = `ABCD`
(CONCAT)
(IN2) : IN_TEXT2(STRING) = `DEF`
↓
Output(OUT) : OUT_TEXT(STRING) = 'ABCDDEF'

CONCAT_TIME

DATE and TOD concatenation

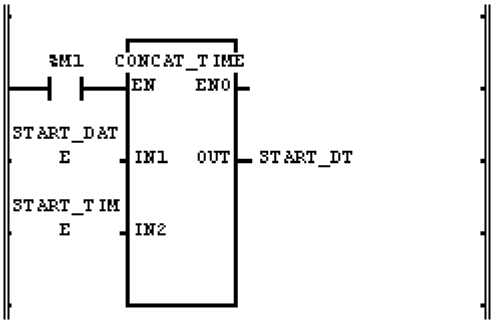
Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
	**Input** EN : Execute the function in case of 1 IN1 : Date data input IN2 : DOT data input **Output** ENO : Output EN value itself OUT : Output the date and DOT

■ Function

Concatenates IN1(DATE) and IN2(TOD) and outputs the resulting DT to OUT.

■ Program example

LD	IL
	LD %M1 JMPN AA LD START_DATE CONCAT_TIME IN1:= CURRENT RESULT IN2:= START_TIME ST START_DT AA :

(1) If the execution condition(%M1) is On, the function CONCAT_TIME is executed.

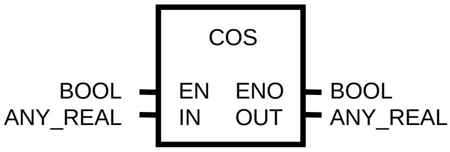
(2) If the operation start data is START_DATE = D#1995-12-06 and operation start time is START_TIME = TOD#08:30:00, START_DT outputs DT#1995-12-06-08:30:00.

Input(IN1) : START_DATE(DATE) = D#1995-12-06
(CONCAT_TIME)
(IN2) : START_TIME(TOD) = TOD#08:30:00
↓
Output(OUT) : START_DT(DT) = DT#1995-12-06-08:30:00

COS

Cosine operation

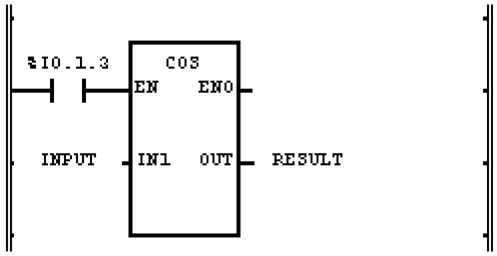
Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●			

Function	Description
 COS BOOL ANY_REAL EN IN ENO OUT BOOL ANY_REAL	**Input** EN : Execute the function in case of 1 IN : Radian value of Cosine operation **Output** ENO : Output EN value itself OUT : Cosine result IN and OUT shall be same data type.

Function

Calculate IN's Cosine value and output the result to OUT.
OUT = COS (IN)

Program example

LD	IL
 %I0.1.3 COS EN ENO INPUT IN1 OUT RESULT	LD %I0.1.3 JMPN CCC LD INPUT COS ST RESULT CCC :

(1) If the execution condition(%I0.1.3) is On, the function COS is executed.

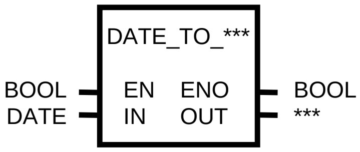
(2) If INPUT variable is 0.5235 (□/6 rad = 30°), output variable RESULT will be 0.8660 .... ($\sqrt{3}$ /2).
COS (□/6) = $\sqrt{3}$ /2 = 0.866

Input(IN1) : INPUT(REAL) = 0.5235
↓ (COS)
Output(OUT) : RESULT(REAL) = 8.66074800E-01

DATE_TO_***

DATE type conversion

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

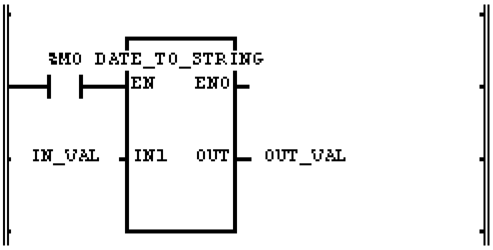
Function	Description
 DATE_TO_*** BOOL DATE EN IN ENO OUT BOOL ***	**Input** EN : Execute the function in case of 1 IN : DATE data to be converted **Output** ENO : Output EN value itself OUT : Type converted data

Function

Convert IN to OUT data type.

FUNCTION	Output type	Description
DATE_TO_UINT	UINT	Convert DATE to UINT type.
DATE_TO_WORD	WORD	Convert DATE to WORD type.
DATE_TO_STRING	STRING	Convert DATE to STRING type.

Program example

LD	IL
 %M0 DATE_TO_STRING EN ENO IN_VAL IN1 OUT OUT_VAL	LD %M0 JMPN LL LD IN_VAL DATE_TO_STRING ST OUT_VAL LL :

(1) If the execution condition(%M0) is On, the function DATE_TO_STRING is executed.

(2) If INPUT variable IN_VAL(DATE type) isD#1995-12-01, output variable OUT_VAL (STRING type) will be 'D#1995-12-01'.

Input(IN1) : IN_VAL(DATE) = D#1995-12-01

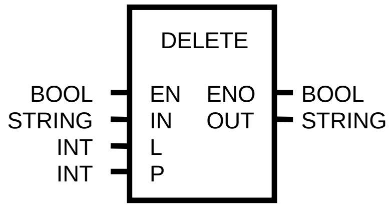
↓ (DATE_TO_STRING)

Output(OUT) : OUT_VAL(STRING) = 'D#1995-12-01'

DELETE

Character string deletion

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
 DELETE BOOL STRING INT INT EN IN L P ENO OUT BOOL STRING	**Input** EN : Execute the function in case of 1 IN : Character string input L : Character string length to be deleted P : Delete position of character string **Output** ENO : Output 1 in case of no error OUT : Character string output

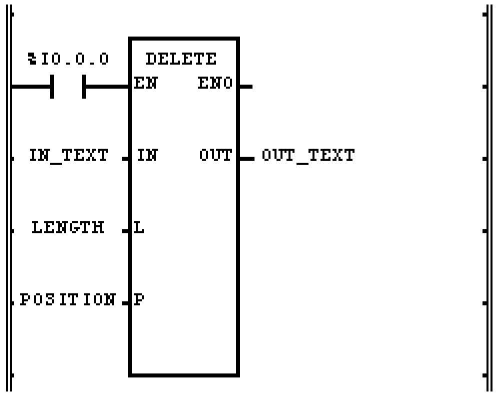
Function

After deleting L characters from P of Character string IN, output it to the character string OUT.

Error

If P≤0 or L<0 or P> (Character number of IN1 input character string), _ERR,_LER flag is set.

Program example

LD	IL
 %I0.0.0 DELETE EN ENO IN_TEXT IN OUT OUT_TEXT LENGTH L POSITION P	LD %I0.0.0 JMPN KKK LD IN_TEXT DELETE IN:= CURRENT RESULT L:= LENGTH P:=POSITION ST OUT_TEXT KKK :

(1) If the execution condition(%I0.0.0) is On, the character string deletion DELETE is executed.

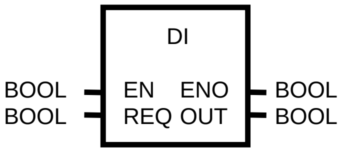
(2) If INPUT variable IN_TEXT(input character)=`ABCDEF` and LENGTH(Character string length to be deleted)=3 and POSITION(Deletion position of character string)=3, output variable OUT_TEXT(STRING type) will be `ABF`.

Input(IN) : IN_TEXT(STRING) = `ABCDEF`
(L) : LENGTH(INT) = 3
(P) : POSITION(INT) = 3
↓ (DELETE)
Output(OUT) : OUT_VAL(STRING) = `ABF`

DI

Prohibits task program operation

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
 DI BOOL BOOL EN REQ ENO OUT BOOL BOOL	**Input** EN : Execute the function in case of 1 REQ : Reguest to prohibit task program operation **Output** ENO : Output EN value itself OUT : Output 1 in case of DI execution

■ Function

□ If EN is 1 and REQ has 1, prohibit the driving the task program(interval, interrupt) programmed by the user.

□ If the normal task program operation is required. Please use 'EI' function.

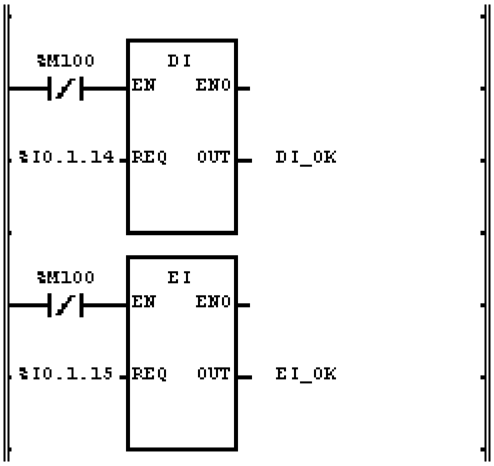
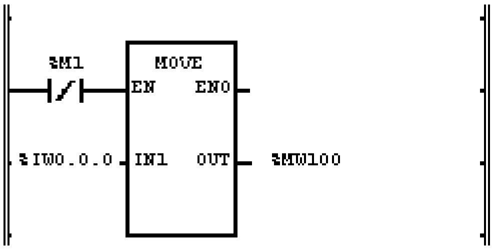
□ If the normal task program operation is required, please use ‘EI’ function.

□ The task generated during task program operation is prohibited is executed as below.

– Interval task, interrupt : These are executed after EI' function execution or completion of current task program. But, if the task is generated more than twice, the task collision error (TASK_ERR) occurs and counts the collision time(TC_CNT)

■ Program example

Program to control task program increasing the value every second using task program driving prohibit function DI and task program driving allowance function E1

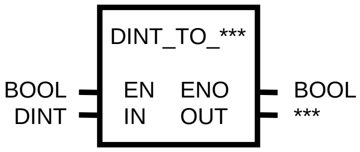
LD	IL
(1) Scan program(TASK program control)  %M100 DI EN ENO %I0.1.14 REQ OUT DI_OK %M100 EI EN ENO %I0.1.15 REQ OUT EI_OK	(1)Scan program controls(TASK program) LDN %M100 JMPN KK LD %I0.1.14 DI ST DI_OK KK : LDN %M100 JMPN LL LD %I0.1.15 EI ST EI_OK LL :
(2) Task program increasing the value every second  %M1 MOVE EN ENO %IW0.0.0 IN1 OUT %MW100	(2)Task program increasing the value every second LDN %M1 JMPN MM LD %IW0.0.0 MOVE ST %MW100 MM :

(1) If REQ(Direct variable %I0.1.14) of DI driving prohibit request is On, the function DI is executed and DI_OK will be 1.
(2) When function DI is executed, the task program executing every second will be stopped.
(3) If REQ(Direct variable %I0.1.15) of EI driving prohibit request is On, the function EI is executed and EI_OK will be 1.
(4) When the function EI is executed, the task program stopped by function DI will be executed again.

DINT_TO_***

DINT type conversion

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
 DINT_TO_*** BOOL DINT EN IN ENO OUT BOOL ***	**Input** EN : Execute the function in case of 1 IN : Double Integer value to be converted **Output** ENO : Output 1 in case of no error OUT : Type converted data

■ Function

Convert IN type and output it to OUT.

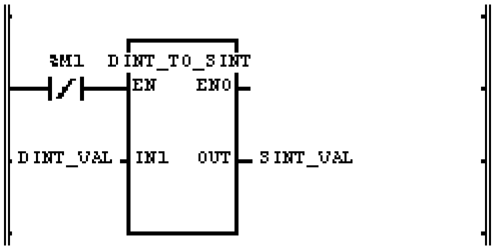
FUNCTION	Output type	Description
DINT_TO_SINT	SINT	If the input is -128～127, it is normally converted but the others cause the error.
DINT_TO_INT	INT	If the input is -32768～32767, it is normally converted but the others cause the error.
DINT_TO_LINT	LINT	Convert LINT type normally.
DINT_TO_USINT	USINT	If the input is 0～255, it is normally converted but the others cause the error.
DINT_TO_UINT	UINT	If the input is 0～65535, it is normally converted but the others cause the error.
DINT_TO_UDINT	UDINT	If the input is 0～2^{32}-1, it is normally converted but the others cause the error.
DINT_TO_ULINT	ULINT	If the input is 0～2^{32}-1, it is normally converted but the others cause the error.
DINT_TO_BOOL	BOOL	Convert lower 1 bit to BOOL type.
DINT_TO_BYTE	BYTE	Convert lower 8 bit to BYTE type.
DINT_TO_WORD	WORD	Convert lower 16 bit to WORD type.
DINT_TO_DWORD	DWORD	Convert internal bit array to DWORD type without conversion.
DINT_TO_LWORD	LWORD	Fill upper bit with 0 to convert it to LWORD type.
DINT_TO_BCD	DWORD	If the input is 0～99,999,999, it is normally converted but the others cause the error.
DINT_TO_REAL	REAL	Convert DINT to REAL type. Conversion error rate is depend on precision.
DINT_TO_LREAL	LREAL	Convert DINT to LREAL type. Conversion error rate is depend on precision.

■ Error

When the conversion error occurs, _ERR, _LER flag is set.

Note When the error occurs, outputs internal bit array without the conversion by taking from lower bit as much as output type bit.

■ Program example

LD	IL
	LD %M1 JMPN LSB LD DINT_VAL DINT_TO_SINT ST SINT_VAL LSB :

(1) When the execution condition(%M1) is On, the data type conversion function DINT_TO_SINT is executed.
(2) If INI = DINT_VAL(DINT type) = -77, SINT_VAL(SINT type) = -77.

Input(IN1) : DINT_VAL(DINT) = -77

Upper	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Lower	1	1	1	1	1	1	1	1	1	0	1	1	0	0	1	1

↓ (DINT_TO_SINT)

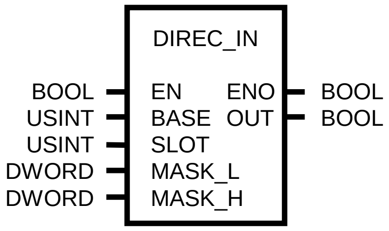
Output(OUT) : OUT_VAL(SINT) = -77

1	0	1	1	0	0	1	1

DIREC_IN

Instant refresh of input data

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	

Function	Description
 DIREC_IN BOOL USINT USINT DWORD DWORD EN BASE SLOT MASK_L MASK_H ENO OUT BOOL BOOL	**Input** EN : Execute the function in case of 1 BASE : The base number of input module is located SLOT : The slot number of input module on the base MASK_L : Mask data for bits that never wanted to refresh among lower 32 bits input data MASK_H : Mask data for bits that never wanted to refresh among upper 32 bits input data **Output** ENO : Output 1 in case of no error OUT : Output 1 if the input data is completely refreshed.

■ Function

□ When EN of DIREC_IN is 1 during scanning, read 64 bit data of input module at allocated location of BASE and SLOT and refresh input image by this data.

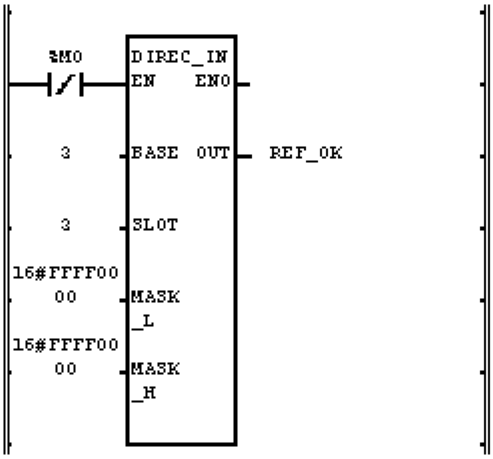
□ Refreshed image region is limited by contact points of input module installed at respective slot.

□ Function DIREC_IN is available to change inputs(%I) On/Off status during scanning.

□ As the scan synchronization batch processing processes input data reading and output data writing after completing scan program, the input data during 1Scan can not be refreshed. Function DIREC_IN can refreshes the relating input during executing the program.

■ Program example

1. Program that instartly refreshes lower 16 bits of assigned input image region when (16 point input module is at 4th slot of 4th base) and input data is 2#1010_1010_1110_1011.

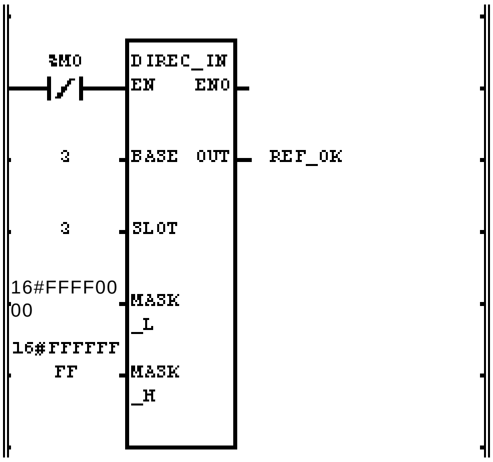
LD	IL
	LD %M0 JMPN ABC LD 3 DIREC_INBASE:= CURRENT RESULT SLOT:= 3 MASK_L:= 16#FFFF0000 MASK_H:= 16#FFFF0000 ST REF_OK ABC :

(1) When the input condition(%M0) is On, DIREC_IN function is executed.

(2) As the installed module is 16 point module, update image region will be %IW3.3.0 and as lower 16Bit is set to refresh in MASK_L(input lower 32Bit), %IW3.3.0 is updated to #1010_1010_1110_1011 during DIREC_IN execution.

(3) MASK_H(input upper 32Bit) value is ignored since 16 point module is installed at assigned slot

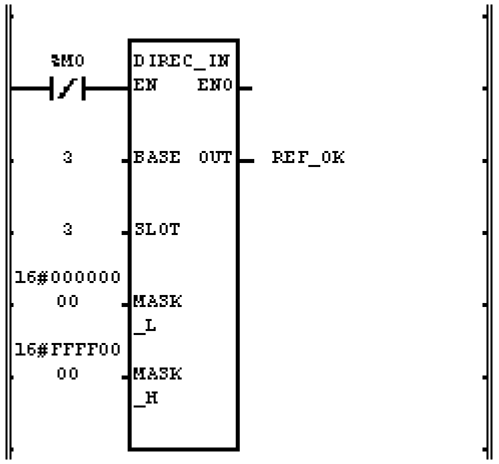
2. Program that instartly refreshes lower 16 bit of input image when 32 points (16 point input module is at 4th slot of 4th base) 2#0000_0000_1111_1111_1100_1100_0011_0011.

LD	IL
	LD %M0 JMPN ABC LD 3 DIREC_INBASE:= CURRENT RESULT SLOT:= 3 MASK_L:= 16#00000000 MASK_H:= 16#FFFFFFFF ST REF_OK ABC :

(1) If the input condition(%M0) is on, DIREC_IN function is executed.

(2) As the installed module is 32 point, the refreshed image region is %ID3.3.0 but as lower 16Bits of MASK_L(input lower 32Bit) is allowed to update, %IW3.3.0 is refreshed to 2#1100_1100_0011_0011.

3. Program that updates lower 48Bits of 64Bits input image promptly when 64 point module is at 4th slot of 4th base and input data is 16#0000_FFFF_AAAA_7777 (2#0000_0000_0000_1111_1111_1111_1111_1010_ 1010_1010_ 1010_0111_0111_ 0111_0111).

LD	IL
 %M0 DIREC_IN EN ENO 3 BASE OUT REF_OK 3 SLOT 16#000000 00 MASK _L 16#FFFF00 00 MASK _H	LD %M0 JMPN ABC LD 3 DIREC_INBASE:= CURRENT RESULT SLOT:= 3 MASK_L:= 16#00000000 MASK_H:= 16#FFFF0000 ST REF_OK ABC :

(1) If input condition(#M0) is on, DIREC_IN(input data prompt update) function is executed.

(2) As the installed module is 64 point, update image region will be %IL3.3.0, i.e., %ID3.3.0 and %ID3.3.1.
As all lower 32Bit(MASK_L) is allowed to update, %ID3.3.0 will be updated.
As lower 16Bits of upper 32Bit(MASK_H) is allowed to update, %IW.3.3.2 is updated but %IW3.3.3 is not updated.
Therefore, the data update of image region is as below.

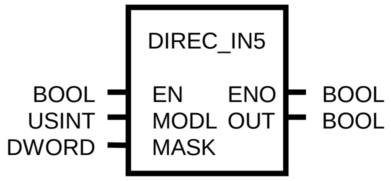
%IL3.3.0 ┌ %ID3.3.0 ┌ %IW.3.3.0:2#0111_0111_0111_0111
         │   └ %IW.3.3.1:2#1010_1010_1010_1010
         └ %ID3.3.1 ┌ %IW3.3.2:2#1111_1111_1111_1111
                    └ %IW3.3.3: maintain previous value

(3) When the input refresh is completed, REF_OK(input data refresh completion) outputs 1.

DIREC_IN5

Input data prompt update

Product	GM1	GM2	GM3	GM4	GM5
Applicable					●

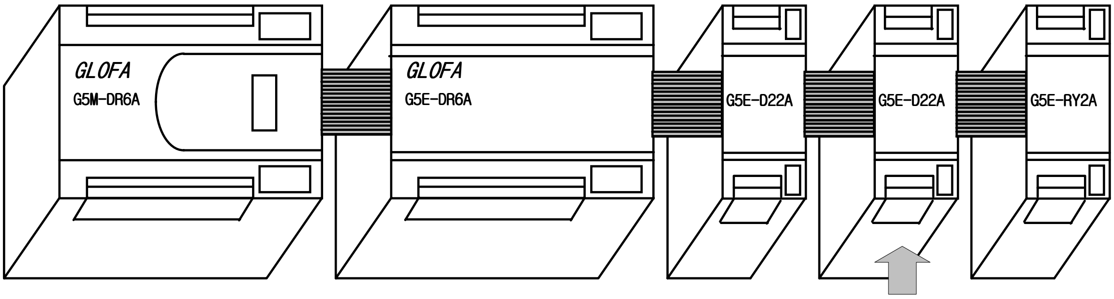
Function	Description
 DIREC_IN5 BOOL USINT DWORD EN MODL MASK ENO OUT BOOL BOOL	**Input** EN : Execute the function in case of 1 MODL : Location number of input module MASK : Bit assignment if input lower 32Bit data is not updated **Output** ENO : Output 1 in case of no error OUT : Output 1 if input data is completely updated

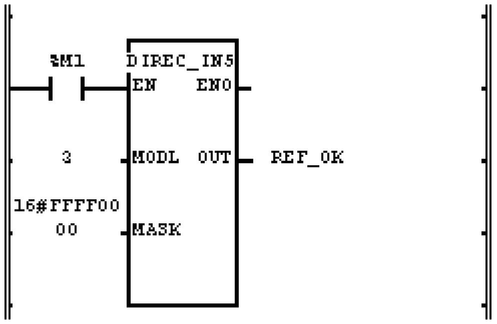
■ Function

- □ EN of DIREC_IN5 is 1 during scanning, read data of input module at allocated location and update it to input image region.
- □ Refreshed image region is limited by contact points of input module installed at respective slot.
- □ Function DIREC_IN5 is available to change input(%I) On/Off status during scanning.
- □ As the scan synchronization batch processing processes input data reading and output data writing after completing scan program, the input data from outside during 1Scan can not be updated. Function DIREC_IN5 can updates the relating input during executing the program.

■ Program example

Program that promptly refreshes lower 16bits of assigned input image when 4th module's input data is 2#1010_1010_1010_1010.



LD	IL
 %M1 DIREC_IN5 EN ENO 3 MODL OUT REF_OK 16#FFFF00 00 MASK	LD %M1 JMPN AAA LD 3 DIREC_IN5 MODL:= CURRENT RESULT MASK:= 16#FFFF0000 ST AAA :

(1) As the installation position is third expansion, set the location number MODL of input module to 3.

(2) As the input module is 16 point, lower 16Bit of MASK value is allowed to refresh.(16#FFFF 0000)

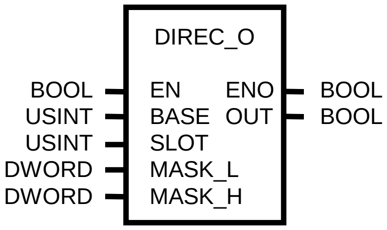
(3) If the execution condition(%M1) is On, DIREC_IN5(input data prompt upgrade) is executed and input data of module is updated promptly.

DIREC_O

Instant refresh of output data

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	

Function



Description

Input
EN : Execute the function in case of 1
BASE : The base number of input module is located
SLOT : The slot number of input module on the base
MASK_L : Mask data for bits that never wanted to refresh among lower 32 bits input data
MASK_H : Mask data for bits that never wanted to refresh among upper 32 bits input data

Output
ENO : Output 1 in case of no error
OUT : Output 1 if output data is completely updated

■ Function

□ When EN of DIREC_O(input data prompt update) is 1 during scanning, read 64bits data from assigned output image and outputs these bits instantly, but masked bits by mask data (MASK_L, MASK_H) are not refreshed. '1' means masked.

□ Function DIREC_0 is available to change output(%Q) On/Off status during scanning.

□ As the batch processing processes the input data reading and output data writing after completing scan program, the data refresh to output module during scan is not possible.

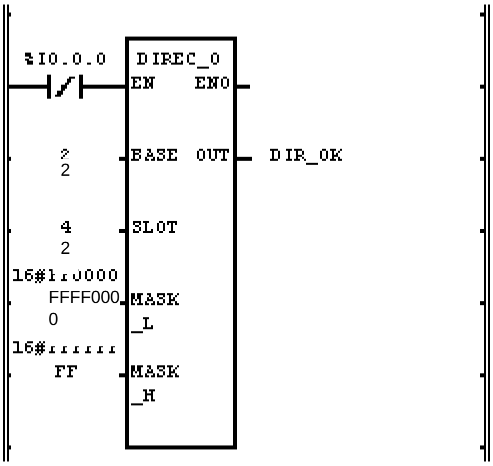
□ Function DIREC_O can output the bit data during scan.

□ If different type module is inserted or data is not written to output module, output EN0 and OUT to 0.(Normal operation : output 1)

■ Program example

1. Program that instantly outputs the output data of 2#0111_0111_0111_0111 to 16 point relay output module installed at third slot of 3rd base.

LD

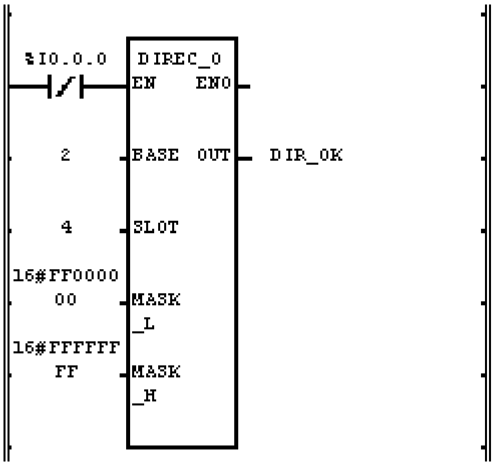


IL

```
LD                       %I0.0.0
JMPN                     AAA
LD                       2
DIREC_O     BASE: =      CURRENT RESULT
            SLOT:=       2
            MASK_L:=     16#FFFF0000
            MASK_H:=     16#FFFFFFFF
ST                    REF_OK
AAA :
```

(1) Input BASE number 2 and SLOT number 4.
(2) As the data wanted to output is 16bits, set lower 16Bit of MASK_L to enable output.(16#FFFF0000)
(3) If the execution condition(%I0.0.0) is On, DIREC_O function is executed and the output module has 2#0111_0111_0111_0111 during scanning.

2. Program that instantly outputs the lower 24Bit of 32 point TR installed at 5th slot of 3rd base, output data is 2#1111_0000_1111_0000_1111_0000.

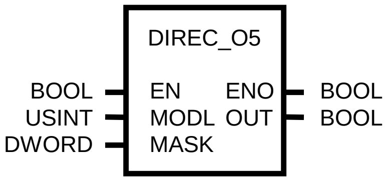
LD	IL
 %I0.0.0 DIREC_O EN ENO 2 BASE OUT DIR_OK 4 SLOT 16#FF0000 00 MASK _L 16#FFFFFF FF MASK _H	LD %I0.0.0 JMPN AAA LD 2 DIREC_O BASE: = CURRENT RESULT SLOT:= 4 MASK_L:= 16#FF000000 MASK_H:= 16#FFFFFFFF ST REF_OK AAA :

(1) Input BASE number 2 and SLOT number 4.
(2) As the data wanted to output is 24bits, lower 24Bit of MASK_L value is allowed to output.(16# FF000000)
(3) If the execution condition(%I0.0.0) is off, function DIREC_0 is executed and output module has 2#□□□□_□□□□_1111_0000_1111_0000_1111_0000.
Previous value

DIREC_O5

Instant refresh of output data

Product	GM1	GM2	GM3	GM4	GM5
Applicable					●

Function	Description
 DIREC_O5 BOOL USINT DWORD EN MODL MASK ENO OUT BOOL BOOL	**Input** EN : Execute the function in case of 1 MODL : Position number of output module MASK : Bit assignment not to be updated among output lower 32Bit data **Output** ENO : Output 1 in case of no error OUT : Output 1 if the output data is updated

■ Function

□ When EN of DIREC_05(input data prompt update) is 1 during scanning, read 64bits data from assigned output image and outputs these bits instantly, but masked bits by mask data (MASK_L, MASK_M) are not refreshed. ‘1’ means masked.

□ Function DIREC_05 is available to change output(%Q) On/Off status during 1scanning.

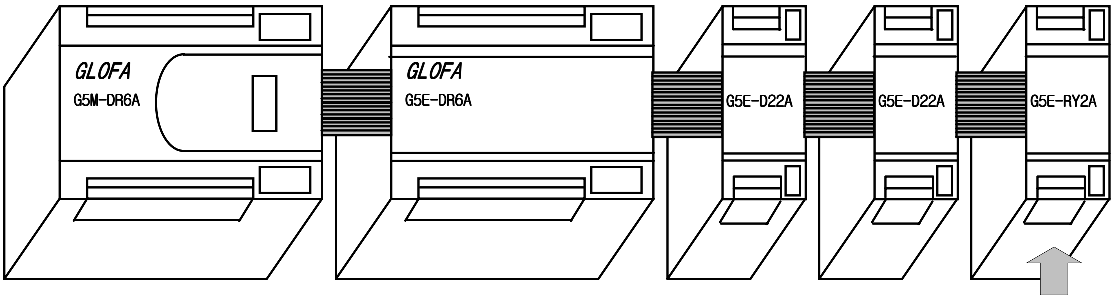
□ As the batch processing processes the input data reading and output data writing after completing scan program, the data refresh to output module during scan is not possible.

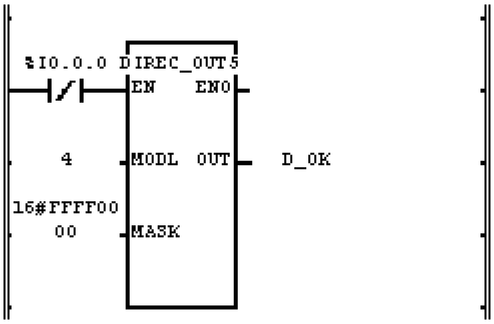
□ Function DIREC_O5 can output the bit data to outside during scan.

□ If different type module is inserted or data is not written normally to output module, output EN0 and OUT to 0.(Normal operation : output 1)

■ Program example

Program that fifth installed output module outputs the output data of 2#1111_0000_1111_0000 under below GM5 system configuration.



LD	IL
 %I0.0.0 DIREC_OUT5 EN ENO 4 MODL OUT D_OK 16#FFFF00 00 MASK	LD %I0.0.0 JMPN CCC LD 4 DIREC_O5 MODL:= CURRENT RESULT MASK:= 16#FFFF0000 ST D_OK CCC :

(1) As the output module is located at expansion fifth, input the location number of output module with 4.

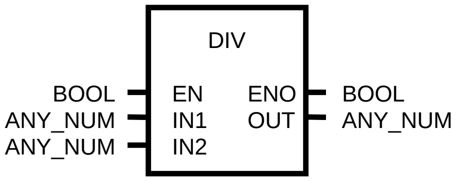
(2) As the module point to output the data during scanning is 16, set MASK value to update allowance of lower 16Bit only.(16#$\underbrace{\text{FFFF}}_{\text{Update prohibit}}$ $\underbrace{\text{0000}}_{\text{Update allowance}}$)

(3) If the execution condition(%I0.0.0) is OFF, DIREC_05 is executed and fifth expanded output module has refreshed data during scan.

DIV

Divide

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
 DIV BOOL ANY_NUM ANY_NUM EN IN1 IN2 ENO OUT BOOL ANY_NUM	**Input** EN : Execute the function in case of 1 IN1 : Dividend IN2 : Divisor **Output** ENO : Output 1 in case of no error OUT : Result(Quotient) Variable connected to IN1, IN2 and OUT shall be same data type.

■ Function

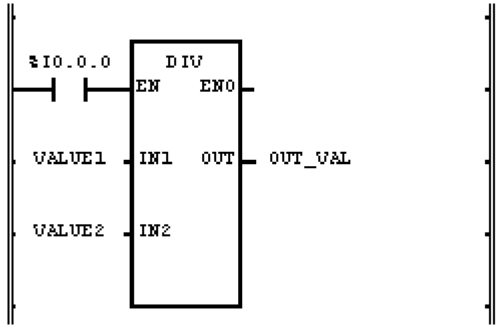
Divide IN1 by IN2 and output the quotient excluding the value below decimal point to OUT.
OUT = IN1/IN2

IN1	IN2	OUT	Remark
7 7 -7 -7	2 -2 2 -2	3 -3 -3 3	Delete the value below decimal point
7	0	×	Error

■ Error

If the divisor is '0', _ERR and _LER flag is set.

■ Program example

LD	IL
 %I0.0.0 DIV EN ENO VALUE1 IN1 OUT OUT_VAL VALUE2 IN2	LD %I0.0.0 JMPN LL LD VALUE1 DIV IN1:= CURRENT RESULT IN2:= VALUE2 ST OUT_VAL LL :

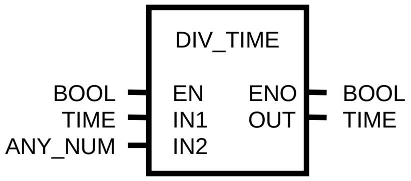
(1) If the execution condition(%I0.0.0) is On, division function DIV is executed.
(2) If input variable VALUE1 = 300 and VALUE2 = 100, output OUT_VAL = 300/100 = 3.

Input(IN1) : VALUE1(INT) = 300(16#012C)	0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0
	/ (DIV)															
(IN2): VALUE2(INT) = 100(16#0064)	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0
	↓															
Output(OUT) : OUT_VAL(INT) = 3(16#3)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

DIV_TIME

Time divide

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
 DIV_TIME BOOL TIME ANY_NUM EN IN1 IN2 ENO OUT BOOL TIME	**Input** EN : Execute the function in case of 1 IN1 : Dividing time IN2 : Dividing value **Output** ENO : Output 1 in case of no error OUT : Result

■ Function

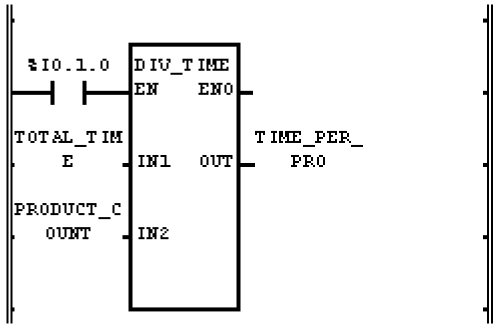
Divide IN1(time) by IN2(number) and output the result to OUT.

■ Error

If divisor(IN2) is 0, _ERR and _LER flag is set.

■ Program example

Program that calculates the time to manufacture one product if one-day work time is 12 hours 24 minutes 24 seconds and one-day production capacity is 12 products.

LD	IL
 %I0.1.0 DIV_TIME EN ENO TOTAL_TIM E IN1 OUT TIME_PER_ PRO PRODUCT_C OUNT IN2	LD %I0.0.0 JMPN SS LD TOTAL_TIME DIV_TIME IN1:= CURRENT RESULT IN2:= PRODUCT_COUNT ST TIME_PER_PRO SS :

(1) If the execution condition(%I0.1.0) is On, the time division function DIV_TIME is executed.

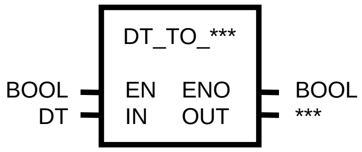
(2) Dividing TOTAL_TIME(T#12H24M24S) by PRODUCT_COUNT(12), TIME_PER_PRO(T#1H2M2S), 1 hour 2 minutes 2 seconds, is output.

Input(IN1) : TOTAL_TIME(TIME) = T#12H24M24S
/ (DIV_TIME)
(IN2) : PRODUCT_COUNT(INT) = 12
↓
Output(OUT) : TIME_PER_PRO(TIME) = T#1H2M2S

DT_TO_***

DT type conversion

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

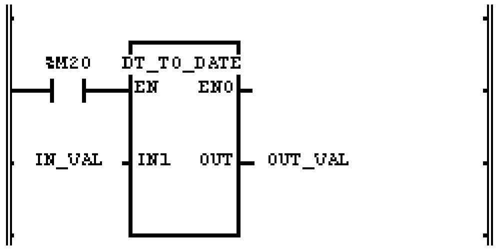
Function	Description
	**Input** EN : Execute the function in case of 1 IN : DATE_AND_TIME data to be converted **Output** ENO : Output EN value itself OUT : Type converted data

■ Function

Convert IN to OUT data type.

FUNCTION	Output type	Description
DT_TO_LWORD	LWORD	Convert DT to LWORD type. (Reverse conversion is available since inter data is not converted.)
DT_TO_DATE	DATE	Convert DT to DATE type.
DT_TO_TOD	TOD	Convert DT to TOD type.
DT_TO_STRING	STRING	Convert DT to STRING type.

■ Program example

LD	IL
	LD %M20 JMPN L LD IN_VAL DT_TO_DATE ST OUT_VAL L :

(1) If the execution condition(%M20) is On, DT type conversion function DT_TO_DATE is executed.

(2) If IN_VAL(DT type) = DT#1995-12-01-12:00:00, OUT_VAL(DATE type) = D#1995-12-01.

Input(IN1) : IN_VAL(DT) = DT#1995-12-01-12:00:00

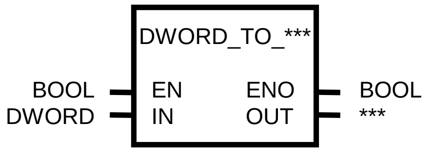
↓ (DT_TO_DATE)

Output(OUT) : OUT_VA(DATE) = D#1995-12-01

DWORD_TO_***

DWORD type conversion

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

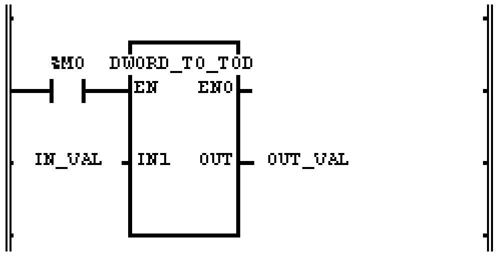
Function	Description
 DWORD_TO_*** BOOL DWORD EN IN ENO OUT BOOL ***	**Input** EN : Execute the function in case of 1 IN : Bit array to be converted(32Bit) **Output** ENO : Output EN value itself OUT : Type converted data

Function

Convert IN to OUT data type.

FUNCTION	Output type	Description
DWORD _TO_SINT	SINT	Convert lower 8Bit to SINT type.
DWORD _TO_INT	INT	Convert lower 16Bit to INT type.
DWORD _TO_DINT	DINT	Convert internal bit array to DINT type without conversion.
DWORD _TO_LINT	LINT	Fill upper bit with 0 to convert it to LINT type.
DWORD _TO_USINT	USINT	Convert lower 8Bit to USINT type.
DWORD _TO_UINT	UINT	Convert lower 16Bit to UINT type.
DWORD _TO_UDINT	UDINT	Convert internal bit array to UDINT type without conversion.
DWORD _TO_ULINT	ULINT	Fill upper bit with 0 to convert it to ULINT type.
DWORD _TO_BOOL	BOOL	Convert lower 1Bit to BOOL type.
DWORD _TO_BYTE	BYTE	Convert lower 8Bit to BYTE type.
DWORD _TO_WORD	WORD	Convert lower 16Bit to WORD type.
DWORD _TO_LWORD	LWORD	Convert upper bit to LWORD type filled with 0.
DWORD _TO_REAL	REAL	Convert internal bit array to REAL type without conversion.
DWORD _TO_TIME	TIME	Convert internal bit array to TIME type without conversion.
DWORD _TO_TOD	TOD	Convert internal bit array to TOD type without conversion.
DWORD _TO_STRING	STRING	Convert input value to decimal and STRING type.

Program example

LD	IL
 %M0 DWORD_TO_TOD EN ENO IN_VAL IN1 OUT OUT_VAL	LD %M0 JMPN AA LD IN_VAL DWORD_TO_WORD ST OUT_VAL AA :

(1) If the execution condition(%M0) is On, type conversion function DWORD_TO_TOD is executed.
(2) If IN_VAL(DWORD type) = 16#3E8(1000), OUT_VAL(TOD type) = TOD#1S.

Input(IN1) : IN_VAL(DWORD) = 16#3E8(1000)	Upper	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Lower	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0

↓ Data type is converted without the data change (internal bit array status)

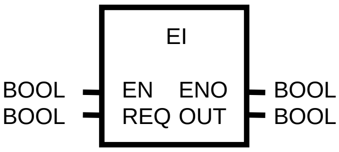
Output(OUT) : OUT_VAL(TOD) = TOD#1S	Upper	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Lower	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0

Note TIME and TOD calculates decimal value by ms unit, i.e., 1000 is calculated as 1000ms = 1s.
(Refer to 3.2.4. Data type structure)

EI

Task program operation allow

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
	**Input** EN : Execute the function in case of 1 REQ : Task program driving allowance request **Output** ENO : Output EN value itself OUT : Output 1 if EI is executed

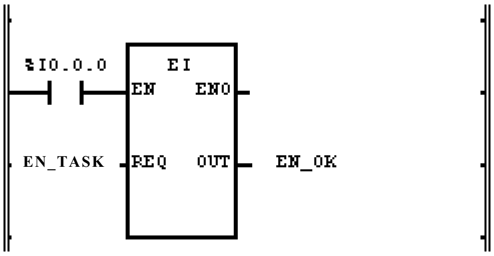
■ Function

□ If EN is 1 and REQ has 1, task program blocked by 'DI' function is operated normally.

□ Once 'EI' command is executed, the task program is operated normally though REQ input is 0.

□ Tasks generated at the driving prohibit status of task program is executed after executing 'EI' function execution or completing current task program.

■ Program example(Refer to DI)

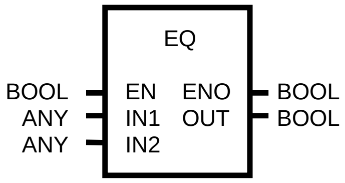
LD	IL
 %I0.0.0 EI EN ENO EN_TASK REQ OUT EN_OK	LD %I0.0.0 JMPN LSB LD EN_TASK EI ST EN_OK LSB :

If EN_TASK is 1, the task program is normally operated.
When the task execution is allowed by 'EI' function, EN_OK outputs 1.

EQ

'Equal' comparison

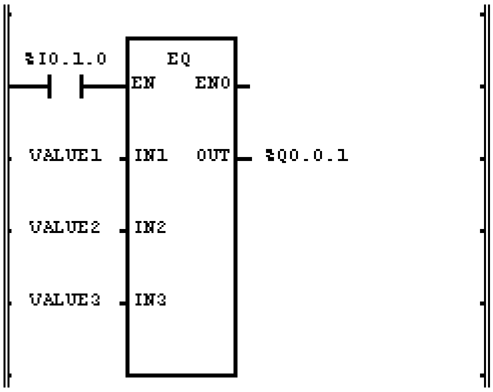
Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
 EQ BOOL ANY ANY EN IN1 IN2 ENO OUT BOOL BOOL	**Input** EN : Execute the function in case of 1 IN1 : Value to be compared IN2 : Comparing value Can be extended to 8 inputs. IN1, IN2, ... shall be same type. **Output** ENO : Output EN value itself OUT : Comparison result

Function

If IN1=IN2=IN3...=INn(n: input number), OUT outputs 1.
Otherwise, OUT outputs 0.

Program example

LD	IL
 %I0.1.0 EQ EN ENO VALUE1 IN1 OUT %Q0.0.1 VALUE2 IN2 VALUE3 IN3	LD %I0.1.0 JMPN AA LD VALUE1 EQ IN1:= CURRENT RESULT IN2:= VALUE2 IN3:= VALUE3 ST %Q0.0.1 AA :

(1) If the execution condition(%I0.1.0) is On, comparison function 'EQ' is executed.

(2) If VALUE1 = 300 and VALUE2 = 300 and VALUE3 = 300, output %Q0.0.1 = 1 since comparison result VALUE1 = VALUE2 = VALUE3.

Input(IN1) : VALUE1(INT) = 300(16#012C)

0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0

= (EQ)

(IN2) : VALUE2(INT) = 300(16#012C)

0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0

= (EQ)

(IN3) : VALUE3(INT) = 300(16#012C)

0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0

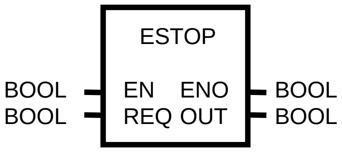
↓

Output(OUT) : %Q0.0.1(BOOL) = 1(16#1) $\boxed{1}$

ESTOP

Emergency stop by program

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
 ESTOP BOOL BOOL EN ENO REQ OUT BOOL BOOL	**Input** EN : Execute the function in case of 1 REQ : Emergency stop request **Output** ENO : Output EN value itself OUT : Output 1 if ESTOP is executed

■ Function

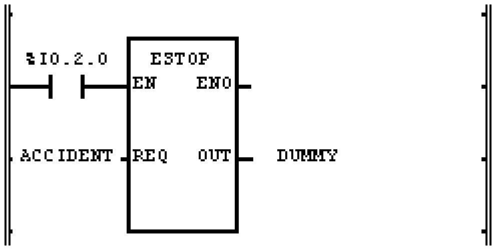
□ If the function execution condition EN is 1 and emergency stop request signal REQ is 1, stop current executing program promptly and go to STOP mode.

□ In case of stop by 'ESTOP' function, the operation is not available though the power is supplied again.

□ Set the operation mode to STOP and from STOP to RUN, the operation starts again.

□ Since 'ESTOP' function stops the program, there may be error of data continuity if not cold restart mode during restarting.

■ Program example

LD	IL
 %I0.2.0 ESTOP EN ENO ACCIDENT REQ OUT DUMMY	LD %I0.2.0 JMPN SSS LD ACCIDENT ESTOP (ST DUMMY) SSS :

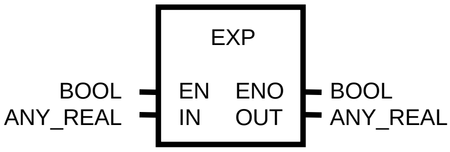
(1) If the execution condition(%I0.2.0) is On, 'ESTOP' function is executed.
(2) If ACCIDENT is 1, the running program stops promptly and go to STOP mode.

Note In case of emergency situation, ESTOP function can be used as redundancy safety device with mechanical interrupt.

EXP

Natural exponent operation

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●			

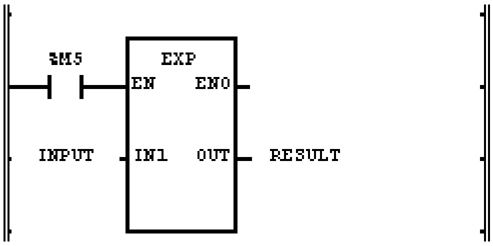
Function	Description
 EXP BOOL ANY_REAL EN IN ENO OUT BOOL ANY_REAL	**Input** EN : Execute the function in case of 1 IN : Input value of exponent operation **Output** ENO : Output EN value itself OUT : Exponent operation result IN and OUT shall be same data type.

Function

Calculate INs exponent value and output it to OUT.

OUT = e^{IN}

Program example

LD	IL
	LD %M5 JMPN JJ LD INPUT EXP ST RESULT JJ :

(1) If the execution condition(%M5) is On, natural exponent function 'EXP' is executed.

(2) If input variable INPUT is 2.0, output variable RESULT will be 7.3890 .....

$e^{2.0}$ = 7.3890.....

Input(IN1) : INPUT(REAL) = 2.0

Upper	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Lower	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0

(16#40000000)

(EXP)

Output(OUT) : RESULT(REAL) = 7.38905621E+00

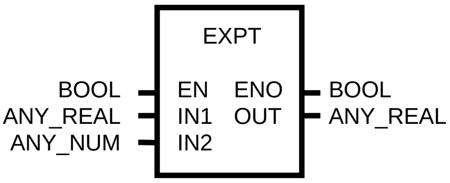
Upper	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Lower	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0

(16#40EC7326)

EXPT

Exponent operation

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●			

Function	Description
 EXPT BOOL ANY_REAL ANY_NUM EN IN1 IN2 ENO OUT BOOL ANY_REAL	**Input** EN : Execute the function in case of 1 IN1 : Real IN2 : Exponent **Output** ENO : Output 1 in case of no error OUT : Result Variable connected to IN1 and OUT shall be same data type.

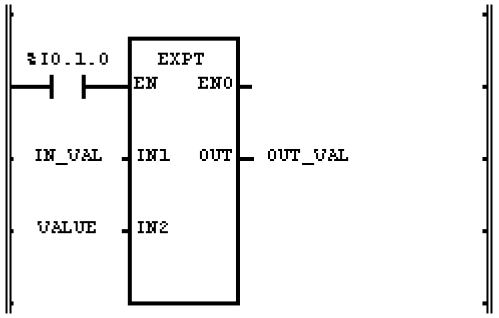
■ Function

Exponent IN1 by IN2 and output it to OUT.
$OUT = INI^{IN2}$

■ Error

If the output exceeds the range of related data type, _ERR and _LER flag is set.

■ Program example

LD	IL
 %I0.1.0 EXPT EN ENO IN_VAL IN1 OUT OUT_VAL VALUE IN2	LD %I0.1.0 JMPN LSB LD IN_VAL EXPT IN1:= CURRENT RESULT IN2:= VALUE ST OUT_VAL LSB :

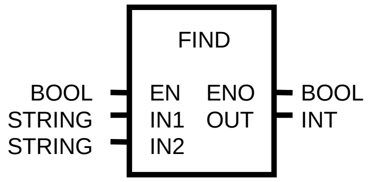
(1) If the execution condition(%I0.1.0) is On, natural exponent function 'EXPT' is executed.
(2) If IN_VAL = 1.5 and VALUE = 3, OUT_VAL = 1.53 = 1.5 × 1.5 × 1.5 = 3.375.

Input(IN1) : IN_VAL(REAL) = 1.5
(IN2) : VALUE(INT) = 3
↓ (EXPT)
Output(OUT) : OUT_VAL(REAL) = 3.37500000E+00

FIND

Find character string

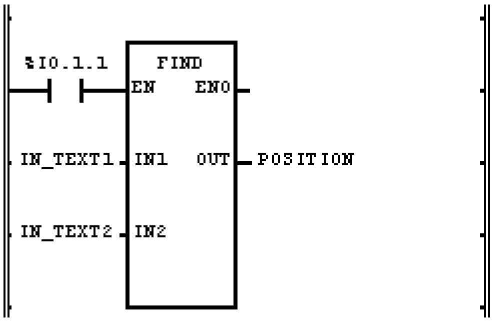
Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
 FIND BOOL STRING STRING EN IN1 IN2 ENO OUT BOOL INT	**Input** EN : Execution the function in case of 1 IN1 : Character string input IN2 : Character to be found **Output** ENO : Output EN value itself OUT : Location of found character string

■ Function

Find character string IN2 in input character string IN1. If find, output the character location of IN2 in IN1 to OUT and if not, output 0 to OUT.

■ Program example

LD	IL
 %I0.1.1 FIND EN ENO IN_TEXT1 IN1 OUT POSITION IN_TEXT2 IN2	LD %I0.1.1 JMPN XYZ LD IN_TEXT1 FIND IN1:= CURRENT RESULT IN2:= IN_TEXT2 ST POSITION XYZ :

(1) If the execution condition(%I0.1.1) is On, execute FIND(character string find) function.

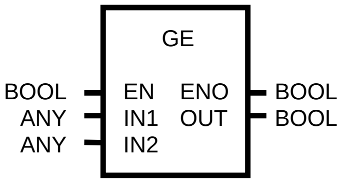
(2) If IN_TEXT1=`ABCEF and IN_TEXT2=`BC`, output variable POSITION=2 is declared.
(The location of IN_TEXT2=`BC` in input character string IN_TEXT1=`ABCEF` is second)

Input(IN1) : IN_TEXT1(STRING) = 'ABCEF'
(FIND)
(IN2) : IN_TEXT2(STRING) = 'BC'
↓
Output(OUT) : POSITION(INT) = 2

GE

'Greater than or equal' comparison

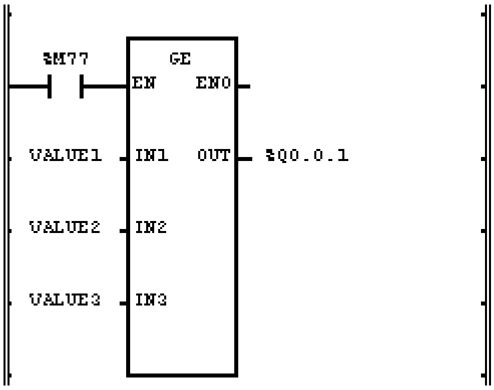
Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
 GE BOOL ANY ANY EN IN1 IN2 ENO OUT BOOL BOOL	**Input** EN : Execute the function in case of 1 IN1 : Value to be compared IN2 : Comparing value Can be extended to 8 inputs. IN1, IN2, ... shall be same type. **Output** ENO : Output EN value itself OUT : Comparison result

■ Function

If IN1 ≥ IN2 ≥ IN3... ≥ INn(n: Input number), OUT outputs 1.
Otherwise, OUT outputs 0.

■ Program example

LD	IL
 %M77 GE EN ENO VALUE1 IN1 OUT %Q0.0.1 VALUE2 IN2 VALUE3 IN3	LD %M77 JMPN YY LD VALUE1 GE IN1:= CURRENT RESULT IN2:= VALUE2 IN3:= VALUE3 ST %Q0.0.1 YY:

(1) If the execution condition(%M77) is On, GE(comparison: larger or equal) function is executed.

(2) If input variable VALUE1=300 VALUE2=200 and VALUE3=100, output result %Q0.01 will be 1 since comparison result VALUE1 ≥ VALUE2 ≥ VALUE3.

Input(IN1) : VALUE1(INT) = 300(16#012C) 0 0 0 0 0 0 0 1 0 0 1 0 1 1 0 0

≥ (GE)

(IN2) : VALUE2(INT) = 200(16#00C8) 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0

≥ (GE)

(IN3) : VALUE3(INT) = 100(16#0064) 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0

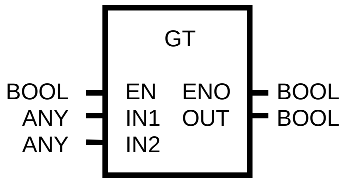
↓

Output(OUT) : %Q0.0.1(BOOL) = 1(16#1) 1

GT

'Greater than' comparison

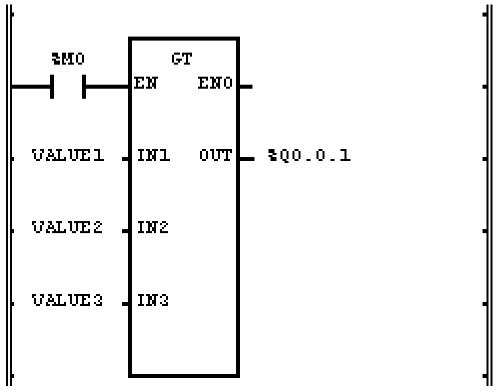
Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
 GT BOOL ANY ANY EN IN1 IN2 ENO OUT BOOL BOOL	**Input** EN : Execute the function in case of 1 IN1 : Value to be compared IN2 : Comparing value Can be extended to 8 inputs. IN1, IN2, ... shall be same type. **Output** ENO : Output EN value itself OUT : Comparison result

■ Function

If IN1>IN2>IN3...>INn(n: input number), OUT outputs 1.
Otherwise, OUT outputs 0.

■ Program example

LD	IL
 %M0 GT EN ENO VALUE1 IN1 OUT %Q0.0.1 VALUE2 IN2 VALUE3 IN3	LD %M0 JMPN AAA LD VALUE1 GT IN1:= CURRENT RESULT IN2:= VALUE2 IN3:= VALUE3 ST %Q0.0.1 AAA :

(1) If the execution condition(%M0) is On, T(Comparison: larger) function is executed.
(2) If input variable VALUE1 = 300, VALUE2 = 200 and VALUE3 = 100, output result %Q0.0.1 will be 1 since comparison result VALUE1 > VALUE2 > VALUE3.

Input(IN1) : VALUE1(INT) = 300(16#012C)

0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0

> (GT)

(IN2) : VALUE2(INT) = 200(16#00C8)

0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0

> (GT)

(IN3) : VALUE3(INT) = 100(16#0064)

0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0

↓

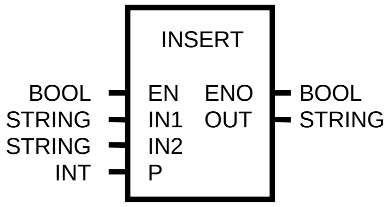
Output(OUT) : %Q0.0.1(BOOL) = 1(16#1)

1

INSERT

Character string insertion

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
 INSERT BOOL STRING STRING INT EN IN1 IN2 P ENO OUT BOOL STRING	**Input** EN : Execute the function in case of 1 IN1 : Character string to be added IN2 : Adding character string P : Character string insert position **Output** ENO : Output 1 in case of no error OUT : Output character string

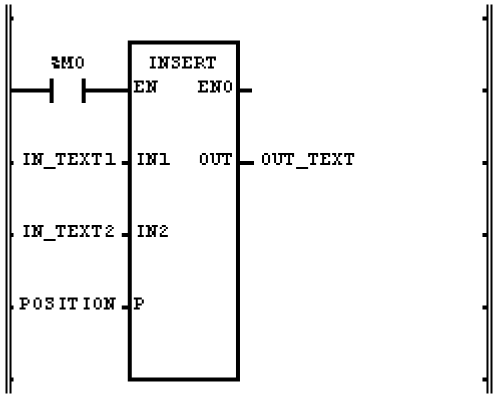
Function

Insert character string IN2 to position of P-th character of IN1 and output it to OUT.

Error

If P≤0 or (character number of variable IN1) < P or if the character number of result exceeds 30, _ERR and _LER flag is set and just 30 characters are output to OUT.

Program example

LD	IL
 %M0 INSERT EN ENO IN_TEXT1 IN1 OUT OUT_TEXT IN_TEXT2 IN2 POSITION P	LD %M0 JMPN AA LD IN_TEXT1 INSERT IN1:= CURRENT RESULT IN2:= IN_TEXT2 P:= POSITION ST OUT_TEXT AA :

(1) If the execution(%M0) is On, INSERT(character string insert) function is executed.

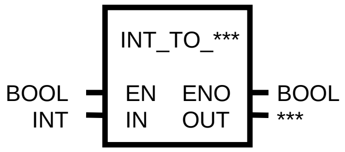
(2) If input variable IN_TEXT1= `ABCD` and IN_TEXT2=`XY` and POSITON=2, output variable OUT_TEXT= `ABXYCD`.

Input(IN1) : IN_TEXT1(STRING) = 'ABCD'
(IN2) : IN_TEXT2(STRING) = 'XY'
(P) : POSITION(INT) = 2
↓ (INSERT)
Output(OUT) : OUT_TEXT = `ABXYCD`

INT_TO_***

INT type conversion

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
	**Input** EN : Execute the function in case of 1 IN : Integer to be converted **Output** ENO : Output 1 in case of no error OUT : Type converted data

■ Function

Convert IN to OUT data type.

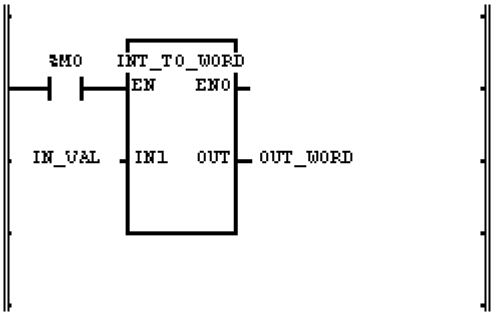
FUNCTION	Output type	Description
INT_TO_SINT	SINT	If input is -128～127, convert integer normally and for other value, the error occurs.
INT_TO_DINT	DINT	Convert to DINT type normally.
INT_TO_LINT	LINT	Convert to LINT type normally.
INT_TO_USINT	USINT	If input is 0～255, convert integer normally and for other value, the error occurs.
INT_TO_UINT	UINT	If input is 0～32767, convert integer normally and for other value, the error occurs.
INT_TO_UDINT	UDINT	If input is 0～32767, convert integer normally and for other value, the error occurs.
INT_TO_ULINT	ULINT	If input is 0～32767, convert integer normally and for other value, the error occurs.
INT_TO_BOOL	BOOL	Convert lower 1Bit to BOOL type.
INT_TO_BYTE	BYTE	Convert lower 8Bit to BYTE type.
INT_TO_WORD	WORD	Convert internal bit array to WORD type without conversion.
INT_TO_DWORD	DWORD	Convert upper bit filled with 0 to DWORD type.
INT_TO_LWORD	LWORD	Convert upper bit filled with 0 to LWORD type.
INT_TO_BCD	WORD	If input is 0～9,999, convert integer normally and for other value, the error occurs.
INT_TO_REAL	REAL	Convert INT to REAL type normally.
INT_TO_LREAL	LREAL	Convert INT to LREAL type normally.

■ Error

When conversion error occurs, _ERR _LER flag is set.

Note When error occurs, outputs bits from lower bit of IN as much as output type bit number without conversion of internal bit array.

■ Program example

LD	IL
 %M0 INT_TO_WORD EN ENO IN_VAL IN1 OUT OUT_WORD	LD %M0 JMPN AAA LD IN_VAL INT_TO_WORD ST OUT_WORD AAA :

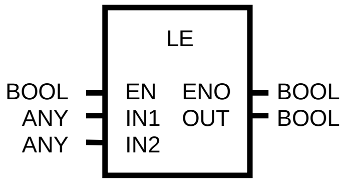
(1) If the execution(%M0) is On, INT_TO_WORD function is executed.
(2) If input variable IN_VAL(INT type) = 512(16#200), output variable OUT_WORD(WORD type) = 16#200.

Input(IN1) : IN_VAL(INT) = 512(16#200)	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
									↓ (INT_TO_WORD)							
Output(OUT) : OUT_WORD(WORD) = 16#200	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

LE

'Less than or equal' comparison

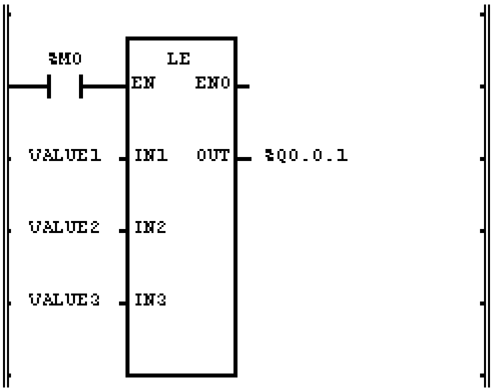
Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
 LE BOOL EN ENO BOOL ANY IN1 OUT BOOL ANY IN2	**Input** EN : Execute the function in case of 1 IN1 : Value to be compared IN2 : Comparing value Can be extended to 8 inputs. IN1, IN2, ... shall be same type. **Output** ENO : Output EN value itself OUT : Comparison result

■ Function

If IN1 ≤ IN2 ≤ IN3... ≤ INn (n: input number), OUT outputs 1.
Otherwise, OUT outputs 0.

■ Program example

LD	IL
 %M0 LE EN ENO VALUE1 IN1 OUT %Q0.0.1 VALUE2 IN2 VALUE3 IN3	LD %M0 JMPN BBB LD VALUE1 LE IN1:= CURRENT RESULT IN2:= VALUE2 IN3:= VALUE3 ST %Q0.0.1 BBB :

(1) If the execution(%M0) is On, LE(Comparison: less or equal) function is executed.
(2) If input variable VALUE1=150 and VALUE2=200 and VALUE3 = 250, output result %Q0.0.1 will be 1.

Input(IN1) : VALUE1(INT) = 150(16#0096)

0	0	0	0	0	0	0	0	1	0	0	1	0	1	1	0

≤ (LE)

(IN2) : VALUE2(INT) = 200(16#00C8)

0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0

≤ (LE)

(IN3) : VALUE1(INT) = 250(16#00FA)

0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0

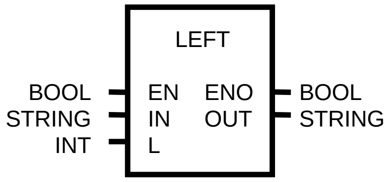
↓

Output(OUT) : %Q0.0.1(BOOL) = 1(16#1) 1

LEFT

Left of character string

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
 LEFT BOOL STRING INT EN IN L ENO OUT BOOL STRING	**Input** EN : Execute the function in case of 1 IN : Character string input L : Character string length to be output **Output** ENO : Output 1 in case of no error OUT : Character string output

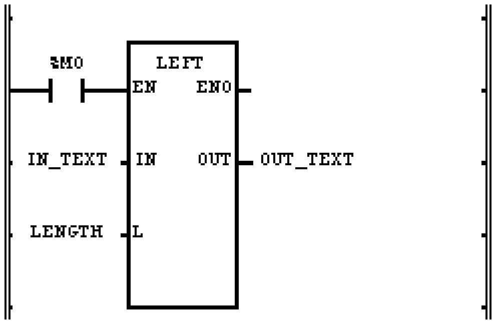
■ Function

Outputs L characters from left character of In to OUT.

■ Error

If L < 0, _ERR and _LER flag is set.

■ Program example

LD	IL
 %M0 LEFT EN ENO IN_TEXT IN OUT OUT_TEXT LENGTH L	LD %M0 JMPN FF LD IN_TEXT LEFT IN:= CURRENT RESULT L:= LENGTH ST OUT_TEXT FF:

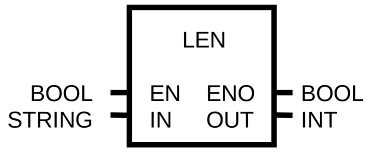
(1) If the execution condition(%M0) is On, LEFT(get left of character string) function is executed.
(2) If IN_TEXT=`ABCDEFG` and LENGTH=3, output character string variable OUT_TEXT=`ABC`.

Input(IN1) : IN_TEXT(STRING) = 'ABCDEFG'
(IN2) : LENGTH(INT) = 3
↓ (LEFT)
Output(OUT) : OUT_TEXT(STRING) = `ABC`

LEN

Character string length

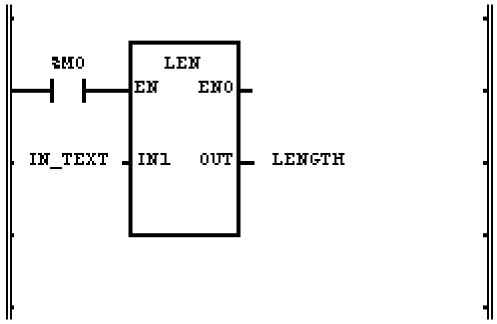
Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
 LEN BOOL STRING EN IN ENO OUT BOOL INT	**Input** EN : Execute the function in case of 1 IN : Character string input **Output** ENO : Output EN value itself OUT : Character string length

Function

Output the length of input character string(IN) to OUT.

Program example

LD	IL
	LD %M0 JMPN II LD IN_TEXT LEN IN:= CURRENT RESULT ST LENGTH II:

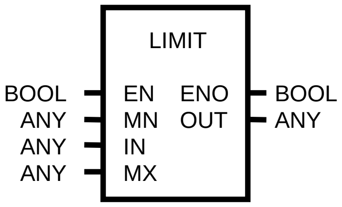
(1) If the execution condition(%M0) is On, LEN(character string length) function is executed.
(2) If input variable IN_TEXT=`ABCD`, output length LENGTH=4.

Input(IN1) : IN_TEXT(STRING) = 'ABCD'
↓ (LEN)
Output(OUT) : LENGTH(INT) = 4

LIMIT

Upper/Lower limit

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
 LIMIT BOOL EN ENO BOOL ANY MN OUT ANY ANY IN ANY MX	**Input** EN : Execute the function in case of 1 MN: Minimum value IN : Limit value MX : Maximum value **Output** ENO : Output EN value itself OUT : Value in the range MN, IN, MX and OUT shall be same data type.

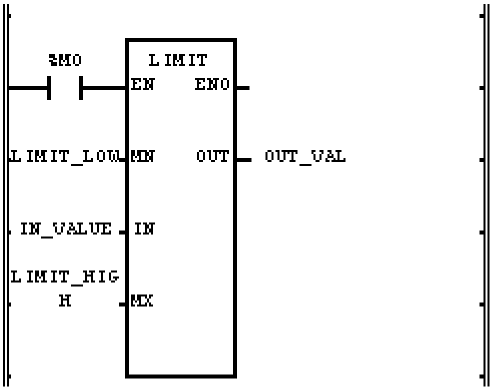
■ Function

□ If input value IN is between MN and MX, OUT outputs IN. Therefore, if MN □ IN □ MX, OUT = IN

□ If input value IN is less than MN, OUT outputs MN. Therefore, if IN < MN, OUT = MN

□ If input value IN is greater than MX, OUT outputs MX. Therefore, if IN > MX, OUT = MX.

■ Program example

LD	IL
 %M0 LIMIT EN ENO LIMIT_LOW MN OUT OUT_VAL IN_VALUE IN LIMIT_HIG H MX	LD %M0 JMPN MM LD LIMIT_LOW LIMIT MN:= CURRENT RESULT IN := IN_VALUE MX:= LIMIT_HIGH ST OUT_VAL MM:

(1) If the execution condition(%M0) is On, LIMIT(upper/lower limit) function is executed.

(2) The output variable(OUT_VAL) on lower limit input variable(LIMIT_LOW), upper limit input variable(LIMIT_HIGH) and limited value input variable(IN_VALUE) is as below.

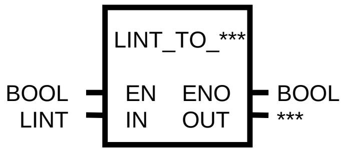
LIMIT_LOW	IN_VALUE	LIMIT_HIGH	OUT_VAL
1000	2000	3000	2000
1000	500	3000	1000
1000	4000	3000	3000

Input(MN) : LIMIT_LOW (INT) = 1000
(IN) : IN_VALUE (INT) = 4000
(MX) : IN_VALUE (INT) = 3000
↓ (LIMIT)
Output(OUT) : OUT_VAL (INT) = 3000

LINT_TO_***

LINT type conversion

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●			

Function	Description
	**Input** EN : Execute the function in case of 1 IN : Long Integer to be converted **Output** ENO : Output 1 in case of no error OUT : Type converted data

■ Function

Convert IN to OUT data type.

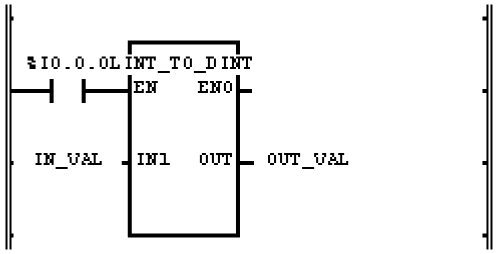
FUNCTION	Output type	Description
LINT_TO_SINT	SINT	If input is -128～127, convert it normally. Otherwise, the error occurs.
LINT_TO_INT	INT	If input is -32,768～32,767, convert it normally. Otherwise, the error occurs.
LINT_TO_DINT	DINT	If input is -2^{31}～2^{31}-1, convert it normally. Otherwise, the error occurs.
LINT_TO_USINT	USINT	If input is 0～255, convert it normally. Otherwise, the error occurs.
LINT_TO_UINT	UINT	If input is 0～65,535, convert it normally. Otherwise, the error occurs.
LINT_TO_UDINT	UDINT	If input is 0～2^{32}-1, convert it normally. Otherwise, the error occurs.
LINT_TO_ULINT	ULINT	If input is 0～2^{64}-1, convert it normally. Otherwise, the error occurs.
LINT_TO_BOOL	BOOL	Convert lower 1Bit to BOOL type.
LINT_TO_BYTE	BYTE	Convert lower 8Bit to BOOL type.
LINT_TO_WORD	WORD	Convert lower 16Bit to BOOL type.
LINT_TO_DWORD	DWORD	Convert lower 32Bit to BOOL type.
LINT_TO_LWORD	LWORD	Convert LINT to LWORD without conversion of internal bit array.
LINT_TO_BCD	LWORD	If input is 0~9,999,999,999,999,999, convert it normally. Otherwise, the error occurs.
LINT_TO_REAL	REAL	Convert LINT to REAL type. Conversion error rate is depend on precision.
LINT_TO_LREAL	LREAL	Convert LINT to LREAL type. Conversion error rate is depend on precision.

■ Error

When conversion error occurs, _ERR and _LER flag is set.

Note When error occurs, outputs bits from lower bit of IN as much as output type bit number without conversion of internal bit array.

■ Program example

LD	IL
 %I0.0.0 LINT_TO_DINT EN ENO IN_VAL IN1 OUT OUT_VAL	LD %I0.0.0 JMPN AAA LD IN_VAL LINT_TO_DINT ST OUT_VAL AAA:

(1) If the execution condition(%I0.0.0) is On, LINT_TO_DINT function is executed.

(2) The output variable IN_VAL(LINT type) = 123_456_789, OUT_VAL(DINT type) = 123_456_789.

Input(IN1) : IN_VAL(LINT) = 123,456,789
(16#75BCD15)

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1	0	1	0	1	1	0	1	1
1	1	0	0	1	1	0	1	0	0	0	1	0	1	0	1

↓ (LINT_TO_DINT)

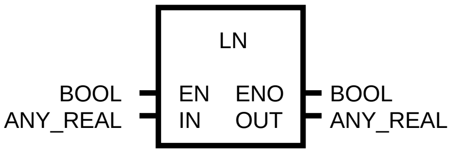
Output(OUT) : OUT_VAL(DINT) = 123,456,789
(16#75BCD15)

0	0	0	0	0	1	1	1	0	1	0	1	1	0	1	1
1	1	0	0	1	1	0	1	0	0	0	1	0	1	0	1

LN

Natural logarithm operation

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●			

Function	Description
 LN BOOL ANY_REAL EN IN ENO OUT BOOL ANY_REAL	**Input** EN : Execute the function in case of 1 IN : Input value of natural logarithm operation **Output** ENO : Output 1 in case of no error OUT : Natural logarithm value IN and OUT shall be same data type.

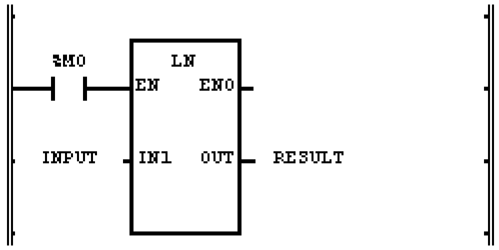
■ Function

Output IN's natural logarithm value to OUT.
OUT = ln IN

■ Error

If the input value is 0 or negative, _ERR and _LER flag is set.

■ Program example

LD	IL
	LD %M0 JMPN AE LD INPUT LN ST RESULT AE:

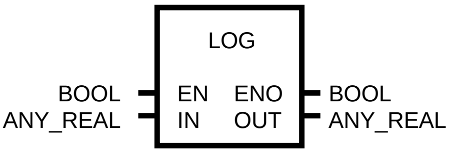
(1) If the execution condition(%M0) is On, LN(natural logarithm operation) function is executed.
(2) The output variable INPUT value is 2.0, output variable RESULT is 0.6931 ....
ln (2.0) = 0.6931....

Input(IN1) : INPUT(REAL) = 2.0
↓ (LN)
Output(OUT) : RESULT(REAL) = 6.93147182E-01

LOG

Logarithm operation

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●			

Function	Description
 LOG BOOL ANY_REAL EN IN ENO OUT BOOL ANY_REAL	**Input** EN : Execute the function in case of 1 IN : Input value of logarithm operation **Output** ENO : Output 1 in case of no error OUT : Commercial logarithm value IN and OUT shall be same data type.

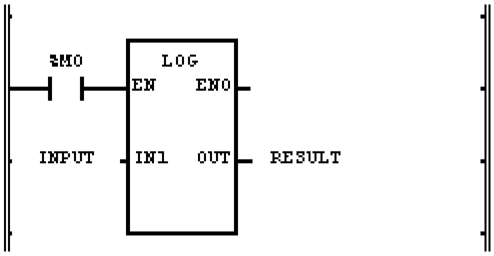
■ Function

Output IN's logarithm value to OUT.
OUT = $\log_{10}$ IN = log IN

■ Error

If the input value is 0 or negative, _ERR and _LER flag is set.

■ Program example

LD	IL
	LD %M0 JMPN BB LD INPUT LOG ST RESULT BB:

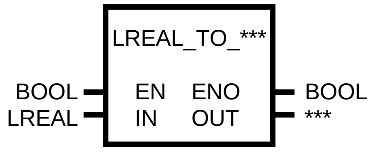
(1) If the execution condition(%M0) is On, LOG(commerical logarithm operation) function is executed.
(2) The output variable INPUT value is 2.0, output variable RESULT is 0.3010 ....
$\log_{10}(2.0) = 0.3010....$

Input(IN1) : INPUT(REAL) = 2.0
↓ (LOG)
Output(OUT) : RESULT(REAL) = 3.01030010E-01

LREAL_TO_***

LREAL type conversion

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●			

Function	Description
 LREAL_TO_*** BOOL LREAL EN IN ENO OUT BOOL ***	**Input** EN : Execute the function in case of 1 IN : LREAL value to be converted **Output** ENO : Execute 1 in case of no error OUT : Type converted data

■ Function

Convert IN to OUT data type.

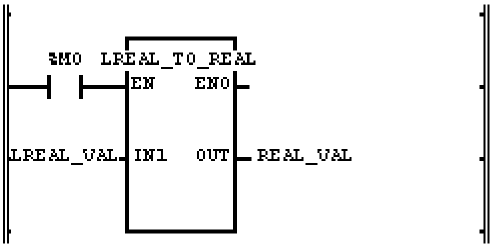
FUNCTION	Output type	Description
LREAL_TO_SINT	SINT	If input integer is -128～127, convert it normally. Otherwise, the error occurs.(round to decimal point)
LREAL_TO_INT	INT	If input integer is -32768～32767, convert it normally. Otherwise, the error occurs.(round to decimal point)
LREAL_TO_DINT	DINT	If input integer is -2^{31}～2^{31}-1, convert it normally. Otherwise, the error occurs.(round to decimal point)
LREAL_TO_LINT	LINT	If input integer is -2^{31}～2^{31}-1, convert it normally. Otherwise, the error occurs.(round to decimal point)
LREAL_TO_USINT	USINT	If input integer is 0～255, convert it normally. Otherwise, the error occurs.(round to decimal point)
LREAL_TO_UINT	UINT	If input integer is 0～65,535, convert it normally. Otherwise, the error occurs.(round to decimal point)
LREAL_TO_UDINT	UDINT	If input integer is 0～2^{32}-1, convert it normally. Otherwise, the error occurs.(round to decimal point)
LREAL_TO_ULINT	ULINT	If input integer is 0～2^{64}-1, convert it normally. Otherwise, the error occurs.(round to decimal point)
LREAL_TO_LWORD	LWORD	Convert LREAL to LWORD type without conversion of internal bit array.
LREAL_TO_REAL	REAL	Convert LREAL to REAL normally. Conversion error rate is depend on precision.

■ Error

If the overflow occurs because input is larger than the storage capacity of output type, _ERR and _LER flag is set.

Note When the error occurs, output 0.

■ Program example

LD	IL
	LD %M0 JMPN LLL LD LREAL_VAL LREAL_TO_REAL ST REAL_VAL LLL:

(1) If the execution condition(%M0) is On, LREAL_TO_REAL function is executed.

(2) The input variable LREAL_VAL(LREAL type) = -1.34E-12, output variable REAL_VAL (REAL type)=-1.34E-12.

Input(IN1) : LREAL_VAL (LREAL) = -1.34E-12

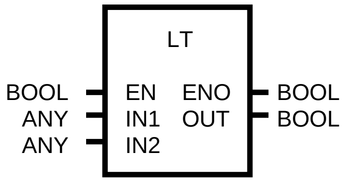
↓ (LREAL_TO_REAL)

Output(OUT) : REAL_VAL (REAL) = -1.34E-12

LT

'Less than' comparison

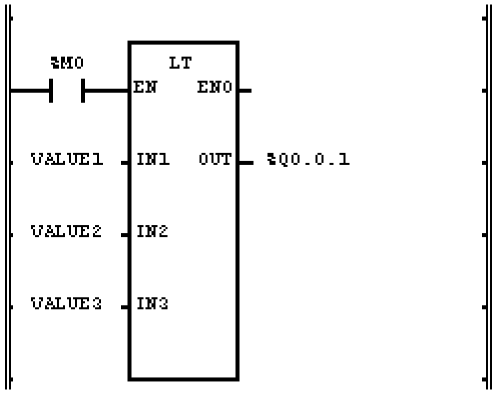
Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
 LT BOOL ANY ANY EN IN1 IN2 ENO OUT BOOL BOOL	**Input** EN : Execute the function in case of 1 IN1 : Value to be compared IN2 : Comparing value Can be extended to 8 inputs. IN1, IN2, ... shall be same type. **Output** ENO : Execute EN itself OUT : Comparison result

■ Function

If IN1<IN2<IN3...<INn(n: input number), OUT outputs 1.
Otherwise, OUT outputs 0.

■ Program example

LD	IL
 %M0 LT EN ENO VALUE1 IN1 OUT %Q0.0.1 VALUE2 IN2 VALUE3 IN3	LD %M0 JMPN AA LD VALUE1 LT IN1:= CURRENT RESULT IN2:= VALUE2 IN3:= VALUE3 ST %Q0.0.1 AA:

(1) If the execution condition(%M0) is On, LT(comparison:less) function is executed.

(2) The input variable VALUE1 = 100 and VALUE2 = 200 and VALUE3 = 300, the output result %Q0.0.1 will be 1 since comparison result VALUE1 < VALUE2 < VALUIE3.

Input(IN1) : VALUE1(INT) = 100(16#0064)

0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0

< (LT)

(IN2) : VALUE2(INT) = 200(16#00C8)

0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0

< (LT)

(IN3) : VALUE3(INT) = 300(16#012C)

0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0

↓

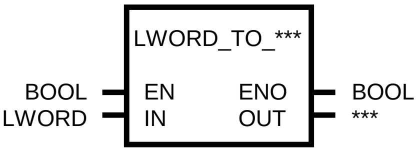
Output(OUT) : %Q0.0.1(BOOL) = 1(16#1)

$\boxed{1}$

LWORD_TO_***

LWORD type conversion

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●			

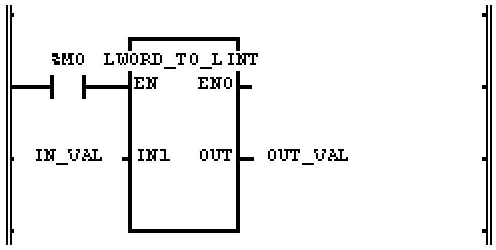
Function	Description
 LWORD_TO_*** BOOL LWORD EN IN ENO OUT BOOL ***	**Input** EN : Execute the function in case of 1 IN : Bit array to be converted(64Bit) **Output** ENO : Output EN value itself OUT : Type converted data

■ Function

Convert IN to OUT data type.

FUNCTION	Output type	Description
LWORD _TO_SINT	SINT	Convert lower 8Bit to SINT type.
LWORD _TO_INT	INT	Convert lower 16Bit to INT type.
LWORD _TO_DINT	DINT	Convert lower 32Bit to DINT type.
LWORD _TO_LINT	LINT	Convert LWORD to LINT without conversion of internal bit array.
LWORD _TO_USINT	USINT	Convert lower 8Bit to USINT type.
LWORD _TO_UINT	UINT	Convert lower 16Bit to UINT type.
LWORD _TO_UDINT	UDINT	Convert lower 32Bit to UDINT type.
LWORD _TO_ULINT	ULINT	Convert LWORD to ULINT without conversion of internal bit array.
LWORD _TO_BOOL	BOOL	Convert lower 1Bit to BOOL type.
LWORD _TO_BYTE	BYTE	Convert lower 8Bit to BYTE type.
LWORD _TO_WORD	WORD	Convert lower 16Bit to WORD type.
LWORD _TO_DWORD	DWORD	Convert lower 32Bit to LWORD type.
LWORD _TO_LREAL	LREAL	Convert LWORD to LREAL type.
LWORD _TO_DT	DT	Convert LWORD to DT type without conversion of internal bit array.
LWORD _TO_STRING	STRING	Convert input value to STRING type.

■ Program example

LD	IL
 %M0 LWORD_TO_LINT EN ENO IN_VAL IN1 OUT OUT_VAL	LD %M0 JMPN PPP LD IN_VAL LWORD_TO_LINT ST OUT_VAL PPP:

(1) If the execution condition(%M0) is On, LWORD_TO_LINT function is executed.

(2) The input variable IN_VAL(LWORD type) = 16#FFFFFFFFFFFFFFFF, the output variable OUT_VAL(LINT type) = -1(16#FFFFFFFFFFFFFFFF).

Input(IN1) : IN_VAL(LWORD) = 16#FFFFFFFFFFFFFFFF

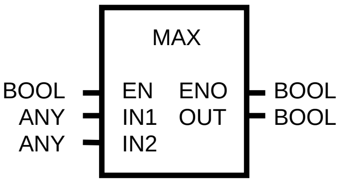
↓ (LWORD_TO_LINT)

Output(OUT) : OUT_VAL(LINT) = -1

MAX

Maximum value

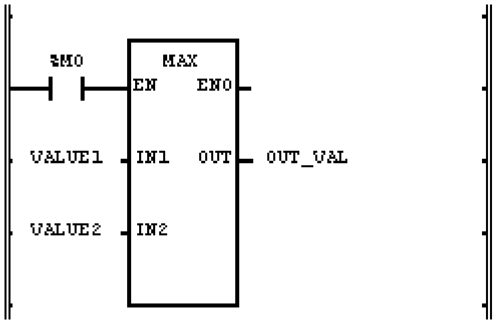
Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
 MAX BOOL ANY ANY EN IN1 IN2 ENO OUT BOOL BOOL	**Input** EN : Execute the function in case of 1 IN1 : Value to be compared IN2 : Comparing value Can be extended to 8 inputs. **Output** ENO : Output EN value itself OUT : Maximum value of input value IN1, IN2, ..., OUT shall be same type.

■ Function

Output maximum value of input among IN1, IN2, ..., INn(n: input number) to OUT.

■ Program example

LD	IL
 %M0 MAX EN ENO VALUE1 IN1 OUT OUT_VAL VALUE2 IN2	LD %M0 JMPN GG LD VALUE1 MAX IN1:= CURRENT RESULT IN2:= VALUE2 ST OUT_VALUE GG:

(1) If the execution condition(%M0) is On, MAX(maximum value) function is executed.

(2) Compare the input variable VALUE1 = 100 and VALUE2 = 200, output OUT_VALUE = 200 since maximum is 200.

Input(IN1) : VALUE1(INT) = 100(16#0064) 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0

(MAX)

(IN2) : VALUE2(INT) = 200(16#00C8) 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0

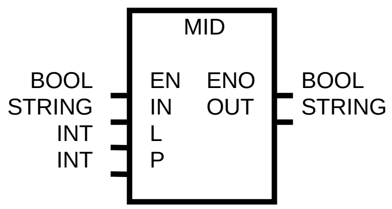
↓

Output(OUT): OUT_VAL(INT) = 200(16#00C8) 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0

MID

Middle of character string

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
	**Input** EN : Execute the function in case of 1 IN : Character string input L : Character string length to be output P : Start position of character string to be output **Output** ENO : Execute 1 in case of no error OUT : Character string output

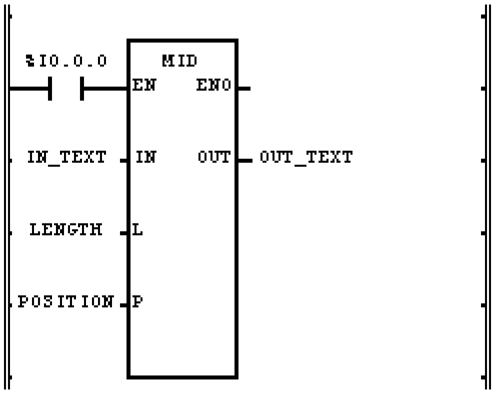
■ Function

Output the character string from P-th character of IN as many as length L to OUT.

■ Error

If (Character number of variable IN) < P or P <= 0 and L < 0, _ERR and _LER flag is set.

■ Program example

LD	IL
	LD %I0.0.0 JMPN MM LD IN_TEXT MID IN:= CURRENT RESULT L: = LENGTH P: = POSITION ST OUT_TEXT MM:

(1) If the execution condition(%I0.0.0) is On, MID(middle of character string) function is executed.

(2) If input character string is IN_TEXT=`ABCDEFG`, character string length is LENGTH=3 and start position of output character string is POSITION=2, output character string variable is OUT_TEXT=`BCD`.

Input(IN) : IN_TEXT1(STRING) = 'ABCDEFG'
(L) : LENGTH(INT) = 3
(P) : POSITION(INT) = 2
↓ (MID)
Output(OUT) : OUT_TEXT = `BCD`