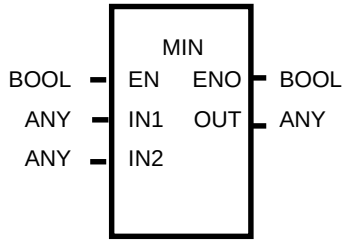
MIN

Minimum value

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

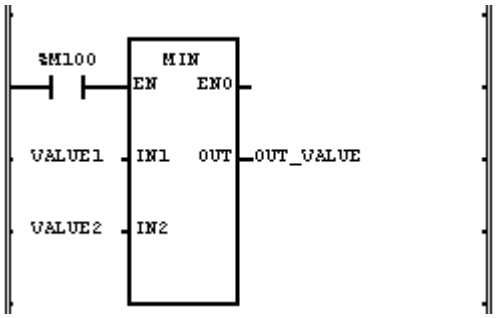
Function	Description
	**Input** EN : Execute the function in case of 1 IN1 : Value to be compared IN2 : Comparing value Input can be extended to 8 **Output** ENO : Output EN value itself OUT : Minimum value of input IN1, IN2, ..., OUT shall be same type.

■ Function

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

■ Program example

LD



IL

```
LD      %M100
JMPN    BBB
LD      VALUE1
MIN     IN1:= CURRENT RESULT
        IN2:= VALUE2
ST      OUT_VALUE
BBB:
```

(1) If the execution condition(%M100) is On, MIN(minimum value) function is executed.

(2) The input variable VALUE1 = 100 and VALUE2 = 200, the output variable OUT_VALUE will be 100 since minimum value is 100.

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

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

(MIN)

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

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

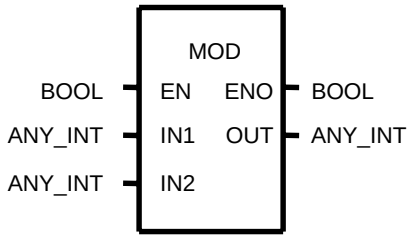
Output(OUT): OUT_VAL(INT) = 100(16#0064)

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

MOD

Remainder calculation

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

Function	Description
	**Input** EN : Execute the function in case of 1 IN1 : Dividend IN2 : Divisor **Output** ENO : Output EN value itself OUT : Dividing result(remainder) Variable connecting to IN1, IN2, OUT shall be same data type.

■ Function

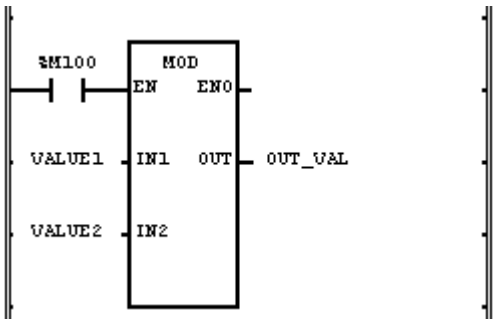
Divide IN1 by IN2 and output the remainder to OUT.

OUT = IN1 - (IN1/IN2) × IN2 (But, if IN2 = 0, OUT = 0)

IN1	IN2	OUT
7	2	1
7	-2	1
-7	2	-1
-7	-2	-1
7	0	0

■ Program example

LD



IL

```
LD      %M100
JMPN    BB
LD      VALUE1
MOD     IN1:=  CURRENT RESULT
        IN2:=  VALUE2
ST      OUT_VAL
BB:
```

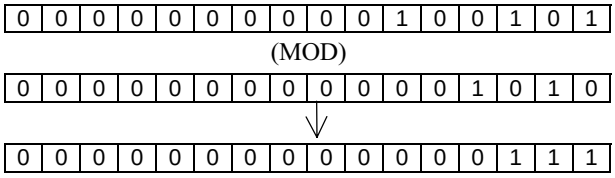
(1) If the execution condition(%M100) is On, MON(remainder calculation) function is executed.

(2) If the dividend VALUE1 = 37 and divisor VALUE2 = 10, the output variable OUT_VAL will be 7.

Input(IN1) : VALUE1(INT) = 37(16#0025)

(IN2) : VALUE2(INT) = 10(16#000A)

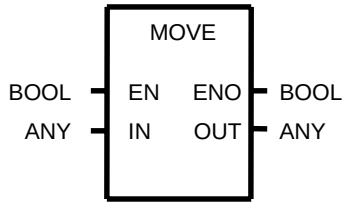
Output(OUT): OUT_VAL(INT) = 7(16#0007)



MOVE

Data move

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

Function	Description
	**Input** EN : Execute the function in case of 1 IN : Value to be moved **Output** ENO : Output EN value itself OUT : Moved value Variable connected to IN and OUT shall be same type.

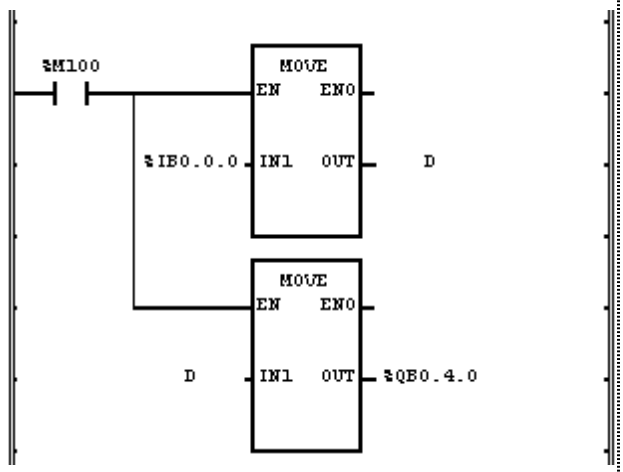
■ Function

Move IN to OUT.

■ Program example

Program that transfers 8 point input of input (%I0.0.0～%I0.0.7) to 8 point output %Q0.4.0～%Q0.4.7.

LD



IL

```
LD      %M100
JMPN    AAA
LD      %IB0.0.0
MOVE
ST      D
LD      D
MOVE
ST      %QB0.4.0
AAA :
```

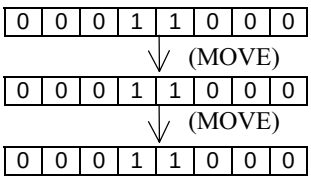
(1) If the execution condition(%M100) is On, MOVE(data copy) function is executed.

(2) Move 8 point input data of input module to variable D area by first MOVE function and output the input module status in variable D to the output module by second MOVE function.

Input(IN1) :%IB0.0.0(BYTE) = 16#18

D(BYTE) = 16#18

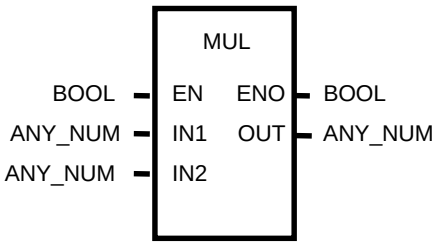
Output(OUT) : %QB0.4.0(BYTE) = 16#18



MUL

Multiply

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

Function	Description
	**Input** EN : Execute the function in case of 1 IN1 : Moltiplicand IN2 : Multiplier Can be extended to 8 inputs. **Output** ENO : Output EN value itself OUT : Multiplied value Variable connected to IN1, IN2, ..., OUT shall be same type.

■ Function

Multiply IN1, IN2,..., INn (n: input number) and output it to OUT.

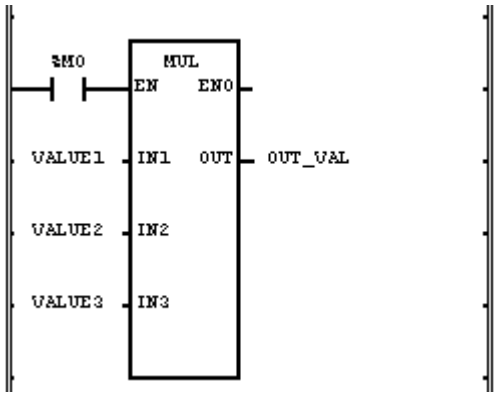
OUT = IN1 × IN2 × ... × INn

■ Error

If the output exceeds the range of respective data type, _ERR and _LER flags are set.

■ Program example

LD



IL

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

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

(2) The input variable VALUE1 = 30 and VALUE2 = 20 and VALUE3 = 10, the output variable OUT_VAL = 30 × 20 × 10 will be 6000.

Input(IN1) : VALUE1(INT) = 30(16#001E)

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

+ (MUL)

(IN2) : VALUE2(INT) = 20(16#0014)

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

+ (MUL)

(IN3) : VALUE3(INT) = 10(16#000A)

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

↓

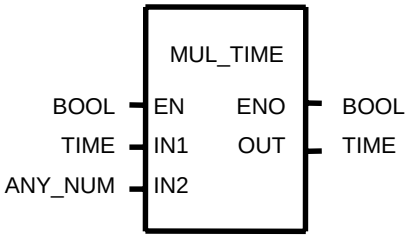
Output(OUT) : OUT_VAL(INT) = 6000(16#1770)

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

MUL_TIME

Time multiply

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

Function	Description
	**Input** EN : Execute the function in case of 1 IN1 : Time to be multiplied IN2 : Multiplying value **Output** ENO : Execute 1 in case of no error OUT : Multiplied result

■ **Function**

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

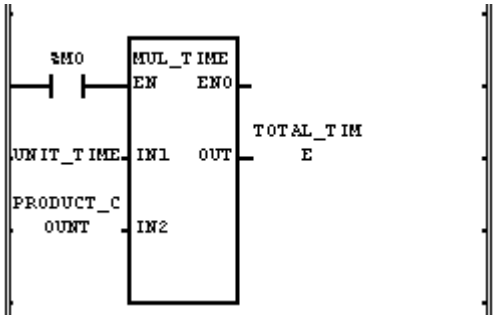
■ **Error**

If the output exceeds the range of TIME data type, _ERR and _LER flags are set.

■ **Program example**

Program that calculates the required work time if average production time is 20 minutes 2 seconds and 20 products shall be manufactured.

LD



IL

```
LD          %M0
JMPN        ABC
LD          UNIT_TIME
MUL_TIME    IN1:=  CURRENT RESULT
            IN2:=  PRODUCT_COUNT
ST          TOTAL_TIME
ABC:
```

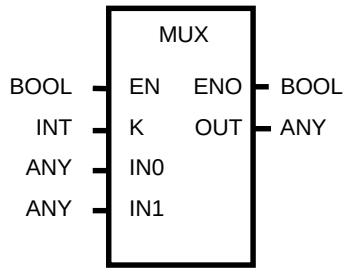
(1) Input UNIT_TIME:T#20M2S to the input variable(1N1:production time of unit product).
(2) Input PRODUCT_COUNT:20 to the input variable(1N2:production quantity).
(3) Input the output variable (OUT:total required work time) to TOTAL_TIME.
(4) If the execution condition(% M0) is On, TOTAL_TIME outputs T#6H40M40S.

Input(IN1) : UNIT_TIME(TIME) = T#20MS2S
(MUL_TIME)
(IN2) : PRODUCT_COUNT(INT) = 20
↓
Output(OUT): TOTAL_TIME(TIME) = T#6H40M40S

MUX

Multiplexer

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

Function	Description
	**Input** EN : Execute the function in case of 1 K : Selection IN0 : Value to be selected IN1 : Value to be selected Can be extended to 7 inputs(IN0, IN1,..., IN6). **Output** ENO : Output 1 in case of no error OUT : Selected value IN0, IN1, ..., OUT shall be same type.

■ Function

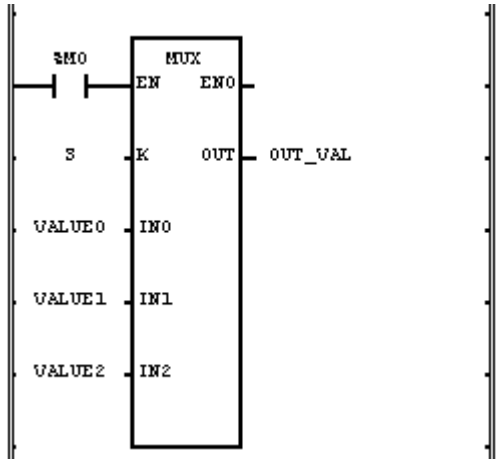
Select one of several input(IN0, IN1,..., INn) and output it to K.
If K = 0, OUT outputs IN0, if K = 1, OUT outputs IN1, if K = n, INn will be output to OUT.

■ Error

If K is larger than or equal to the number of input variable Inn, OUT outputs IN0 and _ERR and _LER flags are set.

■ Program example

LD



IL

```
LD          %M0
JMPN        ABC
LD          S
MUX   K   := CURRENT RESULT
      IN0 := VALUE0
      IN1 := VALUE1
      IN2 := VALUE2
ST          OUT_VAL
ABC:
```

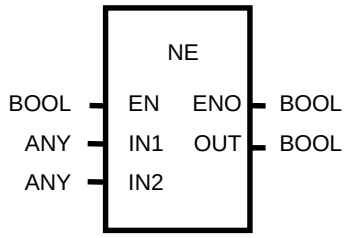
(1) If the execution condition(%M0) is On, MUX(select multiplex) function is executed.
(2) Select on of input variable VALUE0, 1, 2 and output it to OUT.

Input (K) : S(INT) = 2
(IN0) : VALUE0(WORD) = 16#11
(IN1) : VALUE1(WORD) = 16#22
(IN2) : VALUE2(WORD) = 16#33
↓ (MUX)
Output (OUT) : OUT_VAL(WORD) = 16#33

NE

'Not equal' comparison

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

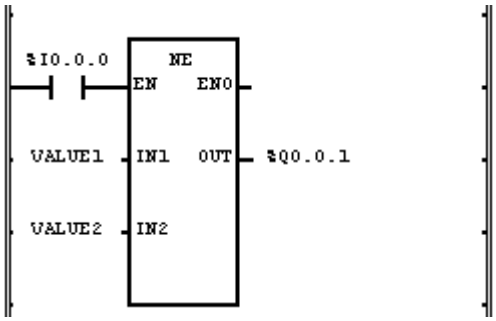
Function	Description
	**Input** EN : Execution allow IN1 : Value to be compared IN2 : Value to be compared IN1 and IN2 shall be same type. **Output** ENO : Execute EN value itself OUT : Comparison result

■ Function

If IN1 is not equal to IN2, OUT outputs 1.
If IN1 is equal to IN2, OUT outputs 0.

■ Program example

LD



IL

```
LD      %I0.0.0
JMPN    PP
LD      VALUE1
NE      IN1:=  CURRENT RESULT
        IN2:=  VALUE2
ST      %Q0.0.1
PP:
```

(1) If the execution condition(%I0.0.0) is On, NE(Comparison: not equal) function is executed.
(2) The input variable VALUE1 = 100 and VALUE2 = 200, the output result %Q0.0.1 will be 1 since VALUE1 is not equal to VALUE2.

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

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

(NE)

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

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

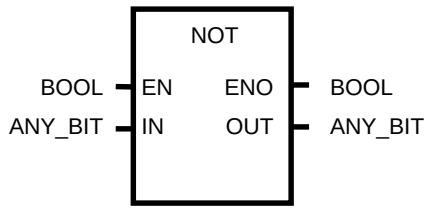
Output(OUT) : %Q0.0.1(BOOL) = 1(16#1)

1

NOT

Not

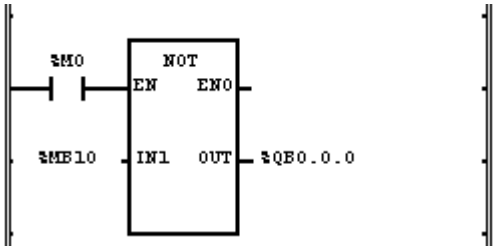
Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
	**Input** EN : Execute the function in case of 1 IN : Value to be NOT **Output** ENO : Output EN value itself OUT : NOT value IN1 and OUT shall be same type.

■ Function

Execute NOT(inversion) to IN and output the result to OUT.
IN 1100 1010
OUT 0011 0101

■ Program example

LD	IL
	LD %M0 JMPN AAA LD %MB10 NOT IN:= CURRENT RESULT ST %QB0.0.0 AAA:

(1) If the execution condition(%M0) is On, NOT function is executed.
(2) If NOT function is executed, invert input variable %MB10 and output the result to output variable %0B0.0.0.0.

Input(IN1) : %MB10(BYTE) = 16#CC

1	1	0	0	1	1	0	0

↓(NOT)

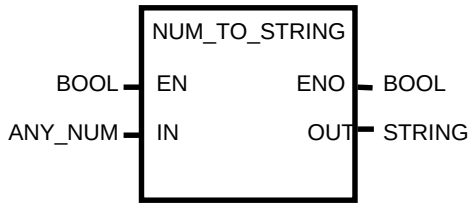
Output(OUT) : %QB0.0.0(BYTE) = 16#33

0	0	1	1	0	0	1	1

NUM_TO_STRING

Number to character string conversion

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

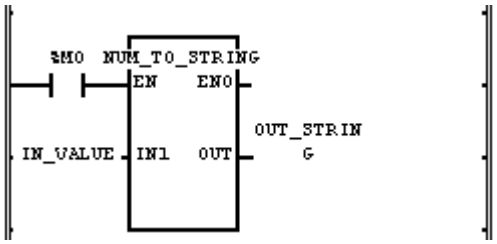
Function	Description
	**Input** EN : Execute the function in case of 1 IN : The number to be converted as STRING **Output** ENO : Output EN value itself OUT : Converted data

■ Function

Convert number to character string and output the result to OUT.

■ Program example

LD



IL

```
LD              %M0
JMPN            AAA
LD              IN_VALUE
NUM_TO_STRING
ST              OUT_STRING
AAA:
```

(1) If the execution condition(%M0) is On, NUM_TO_STRING(Number to character string conversion) function is executed.

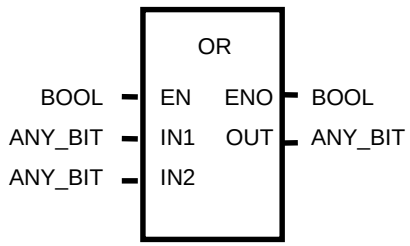
(2) If input variable of NUM_TO_STRING function IN_VALUE(INT type) is 123, output variable OUT_STRING will be '123' and if IN_VALUE(REAL type) is 123.0, OUT_STRING will be '1.23E2'.

Input(IN1) : IN_VALUE(INT) = 123
↓ (NUM_TO_STRING)
Output(OUT) : OUT_STRING(STRING) = '123'

OR

Logical OR

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

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

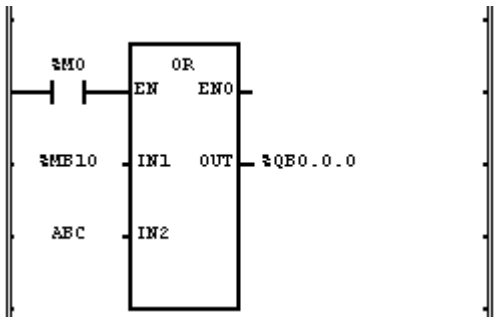
■ Function

Execute OR of IN1 and IN2 and output the result to OUT.

IN1 1111 0000
OR
IN2 1010 1010
OUT 1111 1010

■ Program example

LD



IL

```
LD          %M0
JMPN        AAA
LD          %MB10
OR    IN1:= CURRENT RESULT
      IN2:= ABC
ST          %QB0.0.0
AAA:
```

(1) If the execution condition(%M0) is On, OR function is executed.
(2) OR result of %MB10 = 11001100 and ABC = 11110000 is output to %QB0.0.0 = 11111100.

Input (IN1) : %MB10 (BYTE) = 16#CC

1	1	0	0	1	1	0	0

& (OR)

(IN2) : ABC(BYTE) = 16#F0

1	1	1	1	0	0	0	0

↓

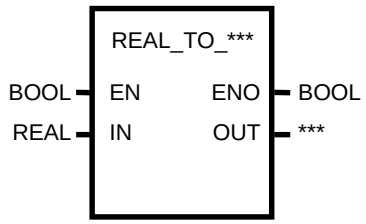
Output(OUT) : %QB0.0.0(BYTE) = 16#FC

1	1	1	1	1	1	0	0

REAL_TO_***

REAL type conversion

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●			

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

■ Function

Convert IN to OUT data type.

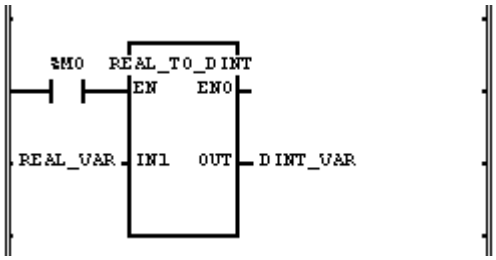
FUNCTION	Output type	Description
REAL_TO_SINT	SINT	If input integer is -128 ~ 127, convert it normally. Otherwise, the error occurs.(round to decimal point)
REAL_TO_INT	INT	If input integer is -32768 ~ 32767, convert it normally. Otherwise, the error occurs.(round to decimal point)
REAL_TO_DINT	DINT	If input integer is -2^{31} ~ $2^{31}-1$, convert it normally. Otherwise, the error occurs.(round to decimal point)
REAL_TO_LINT	LINT	If input integer is -2^{63} ~ $2^{63}-1$, convert it normally. Otherwise, the error occurs.(round to decimal point)
REAL_TO_USINT	USINT	If input integer is 0 ~ 255, convert it normally. Otherwise, the error occurs.(round to decimal point)
REAL_TO_UINT	UINT	If input integer is 0 ~ 65,535, convert it normally. Otherwise, the error occurs.(round to decimal point)
REAL_TO_UDINT	UDINT	If input integer is 0 ~ $2^{32}-1$, convert it normally. Otherwise, the error occurs.(round to decimal point)
REAL_TO_ULINT	ULINT	If input integer is 0 ~ $2^{64}-1$, convert it normally. Otherwise, the error occurs.(round to decimal point)
REAL_TO_DWORD	DWORD	Convert REAL to DWORD without conversion of internal bit array.
REAL_TO_LREAL	LREAL	Convert REAL to LREAL type.

■ Error

If the overflow occurs since input value is greater than the value to be stored at output type, _ERR and _LER flags are set.

Note If the error occurs, output 0.

■ **Program example**

LD	IL
	LD %M0 JMPN AAA LD REAL_VAL REAL_TO_DINT ST DINT_VAL AAA:

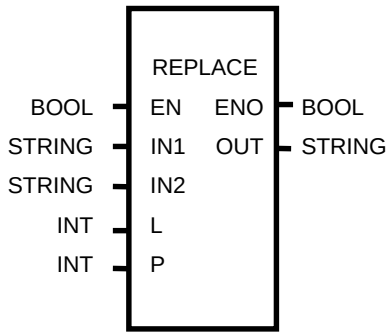
(1) If the execution condition(%M0) is On, REAL_TO_DINT function is executed.
(2) If REAL_VAL(REAL type) = 1.234E4, DINT_VAL(DINT type) = 12340.

Input(IN1) : REAL_VAL(REAL) = 1.234E4
↓(REAL_TO_DINT)
Output(OUT) : DINT_VAL(DINT) = 12340

REPLACE

Character string replacement

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

Function	Description
	**Input** EN : Execute the function in case of 1 IN1 : Character string to be replaced IN2 : Character string to be replaced L : Character string length to be replaced P : Character string position to be replaced **Output** ENO : Output 1 in case of no error OUT : Character string output

■ Function

Replace character of length L from Pth character in character string IN1 to character string IN2 and output the result to character string OUT.

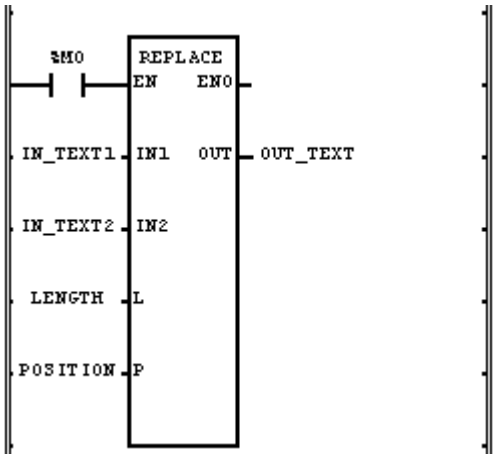
■ Error

For the below case, _ERR and _LER flags are set.

- ☐ P ≤ 0 or L < 0
- ☐ P > (character number of input character string IN1)
- ☐ Character number of operation result > 30

■ Program example

LD



IL

```
LD              %M0
JMPN            ABC
LD              IN_TEXT1
REPLACE  IN1:=  CURRENT RESULT
         IN2: = IN_TEXT2
         L : =  LENGTH
         P : =  POSITION
ST              OUT_TEXT
ABC:
```

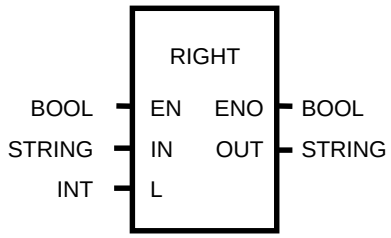
(1) If the execution condition(%M0) is On, REPLACE(character string replace) function is executed.
(2) If input variable to be replaced IN_TEXT1 is `ABCDEF` and replacing input variable IN_TEXT2 is `X` and input variable length to be replaced LENGTH is 3 and location input variable to be replaced POSITION is 2, `BCD` of IN_TEXT is replaced to `X` of IN_TEXT2 and OUT_TEXT outputs `AXET`.

Input (IN1) : IN_TEXT1(STRING) = `ABCDEF`
(IN2) : IN_TEXT2(STRING) = `X`
(L) : LENGTH(INT) = 3
(P) : POSITION(INT) = 2
↓
Output (OUT) : OUT_TEXT(STRING) = `AXET`

RIGHT

Right 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 **Output** ENO : Output 1 in case of no error OUT : Character string output

■ **Function**

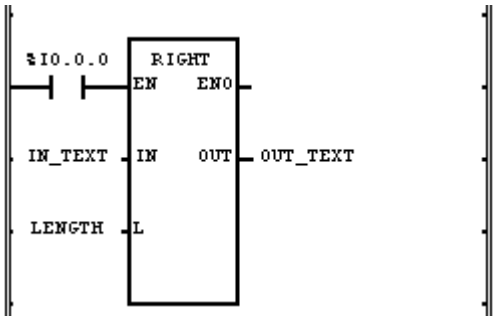
Output character of length L from right of IN to character string OUT.

■ **Error**

If L < 0, _ERR and _LER flags are set.

■ **Program example**

LD



IL

```
LD              %I0.0.0
JMPN            AAA
LD              IN_TEXT
RIGHT   IN :=   CURRENT RESULT
        L : =   LENGTH
ST              OUT_TEXT
AAA:
```

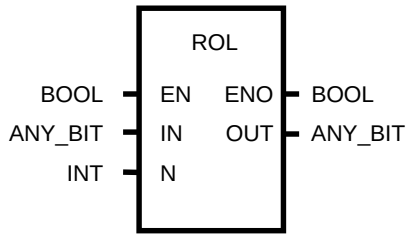
(1) If the execution condition(%M0) is On, RIGHT(right of character string) function is executed.
(2) The input variable IN_TEXT=`ABCDEFG` and strength length LENGTH=3, output variable OUT_TEXT=`EFG`.

Input (IN1) : IN_TEXT(STRING) = `ABCDEFG'
(L) : LENGTH(INT) = 3
↓ (RIGHT)
Output(OUT) : OUT_TEXT(STRING) = `EFG`

ROL

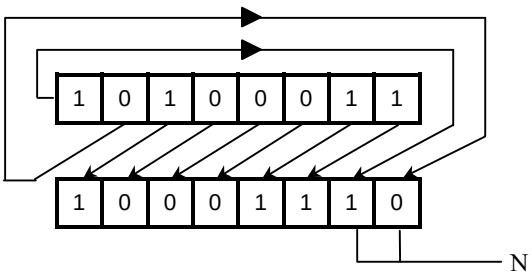
Rotate Left

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

Function	Description
	**Input** EN : Execute the function in case of 1 IN : Value to be rotated N : Bit number to be rotated **Output** ENO : Output EN value itself OUT : Rotated value

■ Function

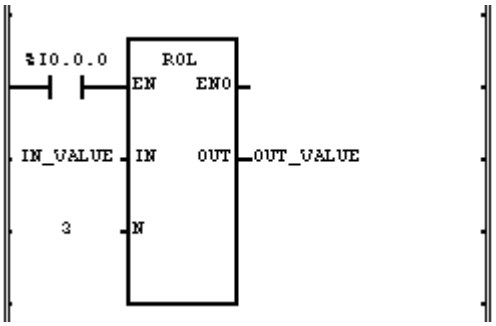
Rotate input IN as many as N bit number to left direction.



■ Program example

Program that rotates input data(1100_1100_1100_1100:16#CCCC) to left direction as 3 bit.

LD

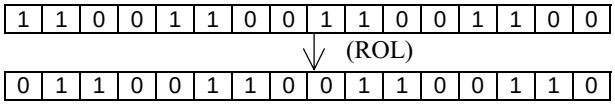


IL

```
LD          %I0.0.0
JMPN        PPP
LD          IN_VALUE
ROL   IN :=  CURRENT RESULT
      N :=  3
ST          OUT_VALUE
PPP:
```

(1) Set the data to be rotated to input value IN_VALUE.
(2) Set bit 3 of rotating bit number to the input(N).
(3) Set the output variable, which outputs the rotated data value, to OUT_VALUE.
(4) If the execution condition %I0.0.0 is On, ROL(rotate left) function is execution so that rotates data bit of input variable to left 3bit and output to OUT_VALVE.

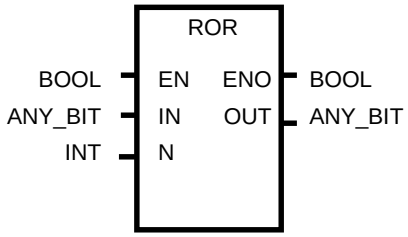
Input (IN1) : IN_VALUE(WORD) = 16#CCCC
(N) : 3
Output(OUT) :OUT_VALUE(WORD) = 16#6666



ROR

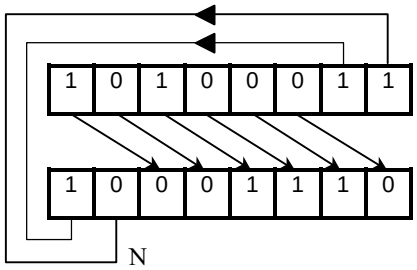
Rotate Right

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

Function	Description
	**Input** EN : Execute the function in case of 1 IN : Value to be rotated N : Bit number to be rotated **Output** ENO : Output EN value itself OUT : Rotated value

■ Function

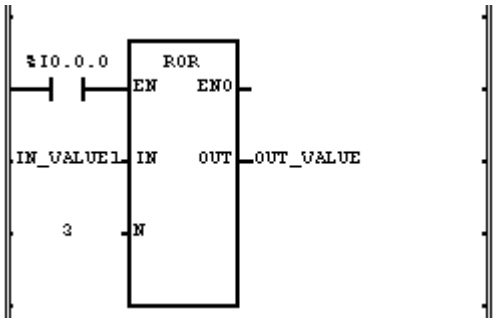
Rotate input IN as many as N bit number to right direction.



■ Program example

Program that rotates input data(1110001100110001:16#E331) to right direction as 3 bit when input %I.0.0.0 is on.

LD

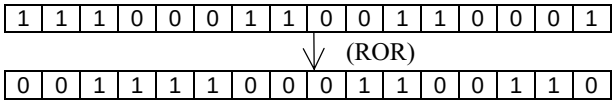


IL

```
LD          %I0.0.0
JMPN        PO
LD          IN_VALUE1
ROR   IN1:= CURRENT RESULT
      N:=   3
ST          OUT_VALUE
PO
```

(1) Set the data to be rotated to input value IN_VALUE1.
(2) Set bit 3 of rotating right bit number to the input(N).
(4) If the execution condition %I0.0.0 is On, ROR(rotate right) function is execution so that rotates data bit of input variable to right 3bit and output to OUT_VALVE.

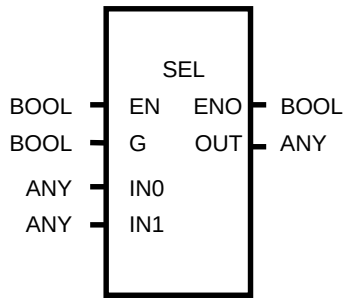
Input (IN1) : IN_VALUE1(WORD) = 16#E331
(N) : 3
Output(OUT) :OUT_VALUE(WORD)=16#3C66



SEL

Selection

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

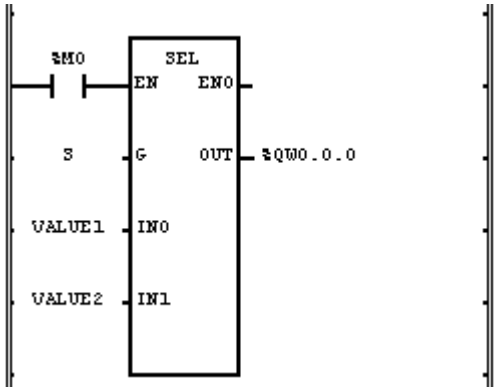
Function	Description
	**Input** EN : Execute the function in case of 1 G : Selection IN0 : Value to be selected IN1 : Value to be selected **Output** ENO : Output EN value itself OUT : Selected value IN1, IN2 and OUT shall be same type.

■ Function

As OUT outputs IN0 when G is 0, OUT outputs IN1 when G is 1.

■ Program example

LD



IL

```
LD          %M0

JMPN        PPP

LD          S
SEL   G :=  CURRENT RESULT
      IN1:= VALUE1
      IN2:= VALUE2
ST          %QW0.0.0
PPP:
```

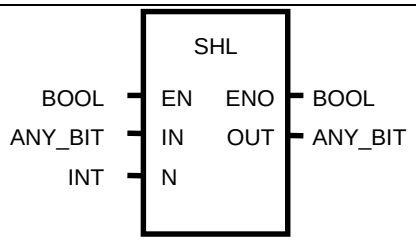
(1) If the execution condition(%M0) is On, SEL(selection) function is executed.
(2) If SEL function is executed, %QW0.0.0 = 16#FF0 when S = 1 and VALUE1 = 16#1110 and VALUE2 = 16#FF00.

Input (G) : S = 1
(IN0) : VALUE1(WORD) = 16#1110
(IN1) : VALUE2(WORD) = 16#FF00
↓ (SEL)
Output(OUT) : %QW0.0.0(WORD) = 16#FF00

SHL

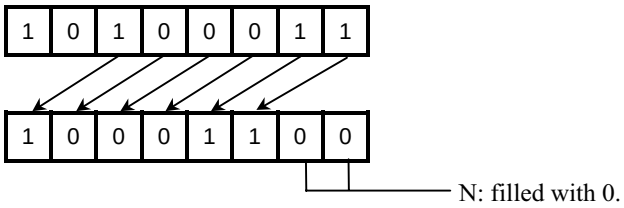
Shift Left

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

Function	Description
	**Input** EN : Execute the function in case of 1 IN : Bit array to be shifted N : Bit number to be moved **Output** ENO : Output EN value itself OUT : Shifted value

■ Function

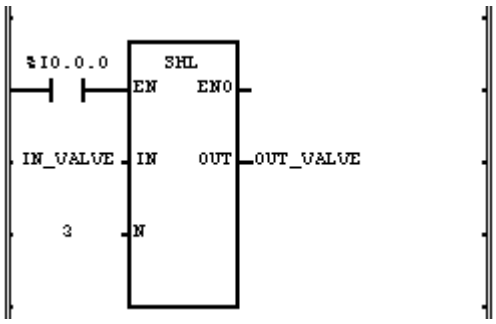
Shift input IN as many as N bit number to 2>2> left direction.
Fill N bit at the right of input IN with 0.



■ Program example

Program that shifts left input data(1100_1100_1100_1100:16#CCCC) to 3 bit when input %I.0.0.0 is on.

LD

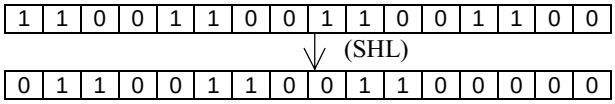


IL

```
LD      %I0.0.0
JMPN    ABC
LD      IN_VALUE
SHL IN:=  CURRENT RESULT
    N :=  3
ST      OUT_VALUE
ABC:
```

(1) Set the input variable to IN_VALUE(11001110:16#CE).
(2) Set Bit number 3 to the input(N).
(3) If the execution condition(%I0.0.0) is On, SHL(shift left) function is executed so that shifts left the input variable data bit to 3 bit and output the result to OUT_VALUE.

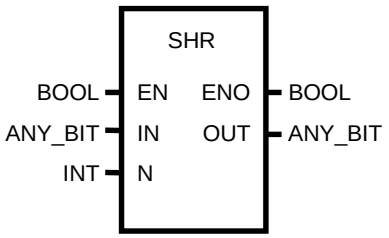
Input(IN1) : IN_VALUE(WORD) = 16#CCCC
(N) : 3
Output(OUT) :OUT_VALUE(WORD) = 16#6660



SHR

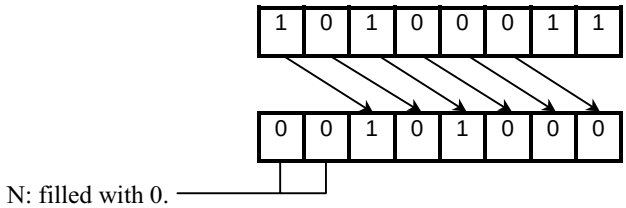
Shift right

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

Function	Description
	**Input** EN : Execute the function in case of 1 IN : Bit array to be shifted N : Bit number to be moved **Output** ENO : Output EN value itself OUT : Shifted value

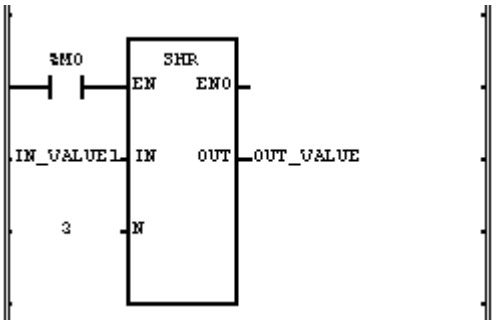
■ Function

Shift input IN as many as N bit number to 2>2> right direction.
Fill N bit at the right of input IN with 0.



■ Program example

LD

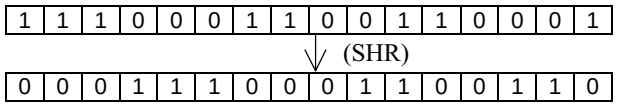


IL

```
LD          %M0
JMPN        AAA
LD          IN_VALUE
SHR   IN :=  CURRENT RESULT
      N :=  SHIFT_NUM
ST          OUT_VALUE
AAA:
```

(1) If the execution condition(%M0) is On, SHR(shift right) function is executed.
(2) Shift right the input variable data bit to 3 bit and output it to output variable OUT_VALUE.

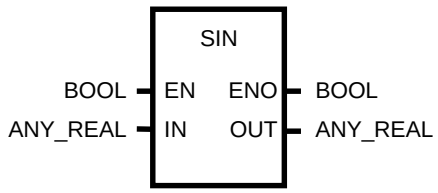
Input (IN1): IN_VALUE(WORD) = 16#E331
(N) : 3
Output(OUT) : OUT_VALUE(WORD) = 16#1C66



SIN

Sine operation

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●			

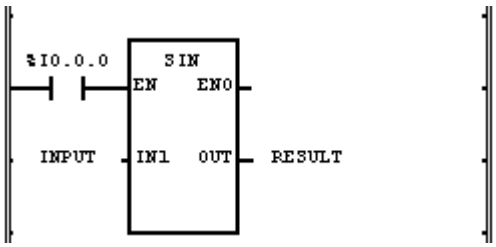
Function	Description
	**Input** EN : Execute the function in case of 1 IN : Radian value of sine operation **Output** ENO : Output EN value itself OUT : Sine operation result IN and OUT shall be same type.

■ Function

Output sine value of IN to OUT.
OUT = SIN (IN)

■ Program example

LD



IL

```
LD      %I0.0.0
JMPN    PPP
LD      INPUT
SIN
ST      RESULT
PPP:
```

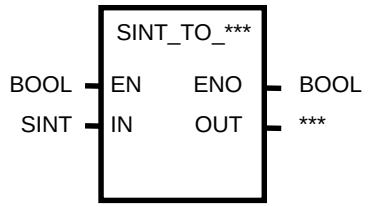
(1) If the execution condition(%I0.0.0) is On, SIN(Sine operation) function is executed.
(2) When input variable INPUT is 1.0471 .... (□/3 rad = 60°), output variable RESULT outputs 0.8660 ....($\sqrt{3}$ /2).
SIN (□/3) = $\sqrt{3}$ /2 = 0.8660

Input(IN1) : INPUT(REAL) = 1.0471
↓(SIN)
Output(OUT) : RESULT(REAL) =8.65976572E-01

SINT_TO_***

SINT type conversion

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

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

■ Function

Convert IN to OUT data type.

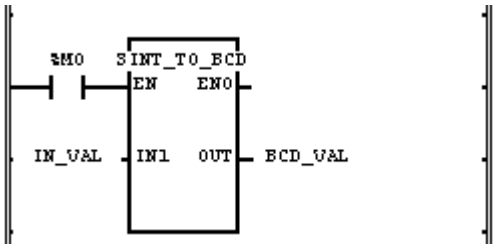
FUNCTION	Output type	Description
SINT_TO_INT	INT	Convert to INT type normally.
SINT_TO_DINT	DINT	Convert to DINT type normally.
SINT_TO_LINT	LINT	Convert to LINT type normally.
SINT_TO_USINT	USINT	If input is 0 ~ 127, convert it normally. Otherwise, the error occurs.
SINT_TO_UINT	UINT	If input is 0 ~ 127, convert it normally. Otherwise, the error occurs.
SINT_TO_UDINT	UDINT	If input is 0 ~ 127, convert it normally. Otherwise, the error occurs.
SINT_TO_ULINT	ULINT	If input is 0 ~ 127, convert it normally. Otherwise, the error occurs.
SINT_TO_BOOL	BOOL	Convert lower 1 bit to BOOL type.
SINT_TO_BYTE	BYTE	Convert SINT to BYTE type without conversion of internal bit array.
SINT_TO_WORD	WORD	Convert upper bit to WORD type filled with 0.
SINT_TO_DWORD	DWORD	Convert upper bit to DWORD type filled with 0.
SINT_TO_LWORD	LWORD	Convert upper bit to LWORD type filled with 0.
SINT_TO_BCD	BYTE	If input is 0 ~ 99, convert it normally. Otherwise, the error occurs.
SINT_TO_REAL	REAL	Convert SINT to REAL type normally.
SINT_TO_LREAL	LREAL	Convert SINT to LREAL type normally.

■ Error

If conversion error occurs, _ERR and _LER flags are set.

Note If error occurs, outputs bits from lower bit of IN as many as output type bit without conversion of internal bit array.

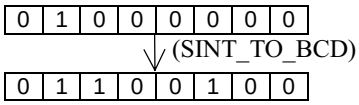
■ **Program example**

LD	IL
	LD %M0 JMPN AAA LD IN_VAL SINT_TO_BCD ST BCD_VAL AAA:

(1) If the execution condition(%M0) is On, SINT_TO_BCD function is executed.
(2) The input variable IN_VAL(SINT type) = 64(2#0100_0000), OUT_VAL(BCD type) = 16#64(2#0110_0100).

Input(IN1) : IN_VAL(SINT) = 64(16#40)

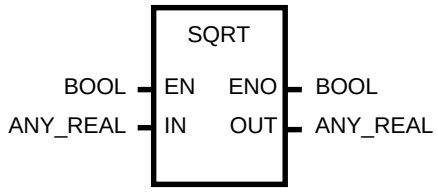
Output(OUT) : OUT_VAL(BCD) = 16#64(16#64)



SQRT

Square root operation

Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●			

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

■ Function

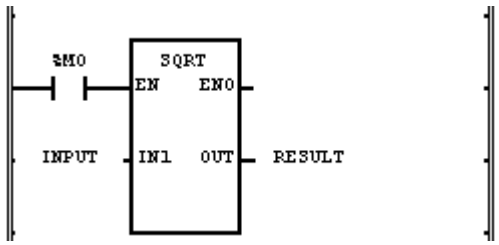
Output the square root of IN to OUT.

OUT = $\sqrt{IN}$

■ Error

If IN is negative number, _ERR and _LER flags are set.

■ Program example

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

(1) If the execution condition(%M0) is On, SQRT(square root operation) function is executed.

(2) The input variable INPUT is 9.0, output variable RESULT will be 3.0.

$\sqrt{9.0} = 3.0$

Input(IN1) : INPUT(REAL) = 9.0

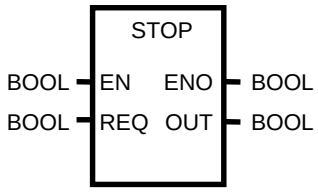
↓ (SQRT)

Output(OUT) : RESULT(REAL) = 3.0

STOP

STOP by program

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

Function	Description
	**Input** EN : Execute the function in case of 1 RE : STOP request **Output** ENO : Output EN value itself OUT : Output 1 when STOP is executed

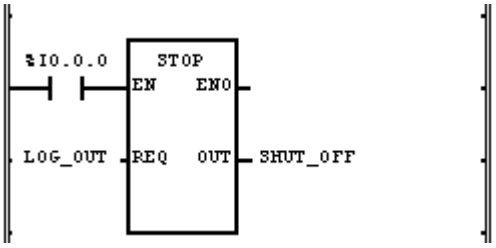
■ Function

☐ If EN is 1 and REQ has 1, stop the operation and go to STOP mode.

☐ If 'STOP' function is executed, scan program will be stopped after completing its last function.

☐ The operation will be run again by supplying the power or changing the mode to STOP and from STOP to RUN.

■ Program example

LD	IL
	LD %I0.0.0 JMPN PT LD LOG_OUT STOP ST SHUT_OFF PT:

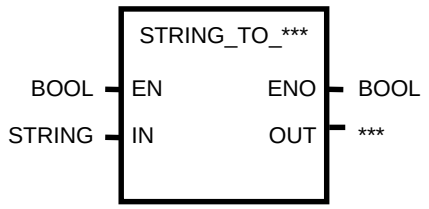
(1) If the execution condition(%I0.0.0) is ON and LOG_OUT is 1, go to STOP mode after completing the running scan program.

(2) Switch off PLC power after executing 'STOP' function.

STRING_TO_***

STRING type conversion

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

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

■ **Function**

Convert IN to OUT data type.

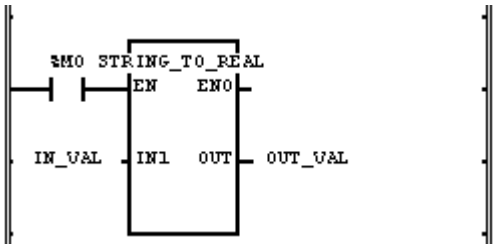
FUNCTION	Output type	Description
STRING _TO_SINT	SINT	Convert STRING to SINT type.
STRING _TO_INT	INT	Convert STRING to INT type.
STRING _TO_DINT	DINT	Convert STRING to DINT type.
STRING _TO_LINT	LINT	Convert STRING to LINT type.
STRING _TO_USINT	USINT	Convert STRING to USINT type.
STRING _TO_UINT	UINT	Convert STRING to UINT type.
STRING _TO_UDINT	UDINT	Convert STRING to UDINT type.
STRING _TO_ULINT	ULINT	Convert STRING to ULINT type.
STRING _TO_BOOL	BOOL	Convert STRING to BOOL type.
STRING _TO_BYTE	BYTE	Convert STRING to BYTE type.
STRING _TO_WORD	WORD	Convert STRING to WORD type.
STRING _TO_DWORD	DWORD	Convert STRING to DWORD type.
STRING _TO_LWORD	LWORD	Convert STRING to LWORD type.
STRING _TO_REAL	REAL	Convert STRING to REAL type.
STRING _TO_LREAL	LREAL	Convert STRING to LREAL type.
STRING _TO_DT	DT	Convert STRING to DT type.
STRING _TO_DATE	DATE	Convert STRING to DATE type.
STRING _TO_TOD	TOD	Convert STRING to TOD type.
STRING _TO_TIME	TIME	Convert STRING to TIME type.

■ **Error**

If input character pattern does not match to output data type, _ERR and _LER flags are set.

■ **Program example**

LD



IL

```
LD              %M0
JMPN            ZZ
LD              IN_VAL
STRING_TO_REAL
ST              OUT_VAL
ZZ:
```

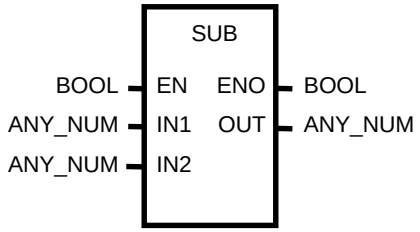
(1) If the input condition(%M0) is On, STRING_TO_REAL function is executed.
(2) If input variable IN_VAL(STRING type) = '-1.34E12', output variable OUT_VAL(REAL= -1.34E12.

Input(IN1) : IN_VAL(STRING) = '-1.34E12'
↓(STRING_TO_REAL)
Output(OUT) : OUT_VAL(REAL) = -1.34E12

SUB

Subtract

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

Function	Description
	**Input** EN : Execute the function in case of 1 IN1 : Minuend IN2 : Subtrahend **Output** ENO : Output 1 in case of no error OUT : Maximum value of input value IN1, IN2, ..., OUT shall be same type.

■ Function

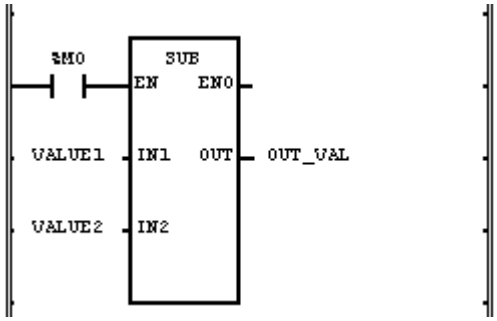
Subtract IN2 from IN1 and output the result to OUT.
OUT = IN1–IN2

■ Error

If the output exceeds the range of respective data type, _ERR and _LER flags are set.

■ Program example

LD



IL

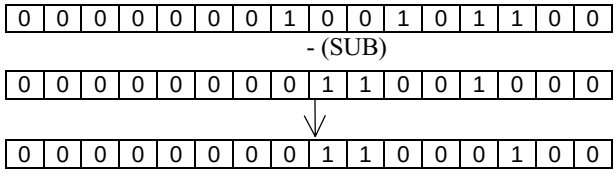
```
LD      %M0
JMPN    AAA
LD      VALUE1
SUB     IN1:=  CURRENT RESULT
        IN2:=  VALUE2
ST      OUT_VAL
AAA:
```

(1) If the execution condition(%M0) is On, SUB(subtract) function is executed.
(2) The input variable VALUE1 = 300 and VALUE2 = 200, the output variable OUT_VAL outputs (300-200=100).

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

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

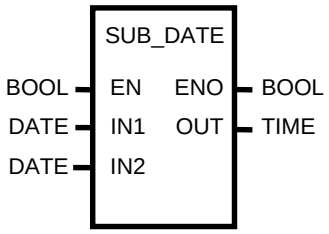
Output (OUT) : OUT_VAL(INT) = 100(16#0064)



SUB_DATE

Subtract date

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

Function	Description
	**Input** EN : Execute the function in case of 1 IN1 : Reference date IN2 : Date to be subtracted **Output** ENO : Output 1 in case of no error OUT : The difference of two dates as time

■ Function

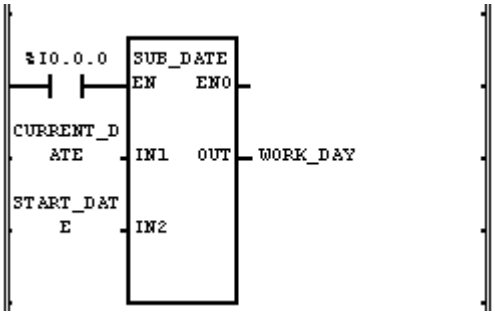
Subtract IN2(specific date) from IN1(reference date) and output the difference to OUT.

■ Error

If the output exceeds the range of TIME data type, _ERR and _LER flags are set.
If the difference exceeds the range of TIME data type T#49D17H2M47S295MS or the result is negative, the error occurs.

■ Program example

LD



IL

```
LD          %I0.0.0
JMPN        PPP
LD          CURRENT_DATE
SUB_DATE    IN1:=  CURRENT RESULT
            IN2:=  START_DATE
ST          WORK_DAY
PPP:
```

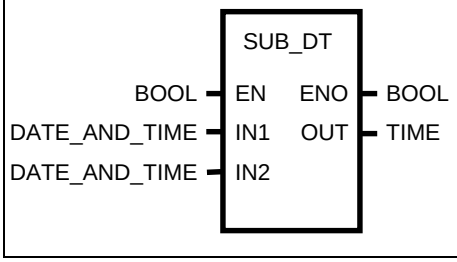
(1) If the execution condition(%I0.0.0) is On, SUB_DATE(subtract date) function is executed.
(2) If CURRENT_DATE is D#1995-12-15 and START_DATE is D#1995-11-1, WORK_DAY outputs T#44D.

Input(IN1) : CURRENT_DATE(DATE) = D#1995-12-15
(SUB_DATE)
(IN2) : START_DATE(DATE) = D#1995-11-1
↓
Output (OUT) : WORK_DAY(TIME) = T#44D

SUB_DT

Subtract DATE_AND_TIME

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

Function	Description
	**Input** EN : Execute the function in case of 1 IN1 : Reference DATE_AND_TIME IN2 : DATE_AND_TIME to be subtracted **Output** ENO : Output 1 in case of no error OUT : Subtracted time

■ Function

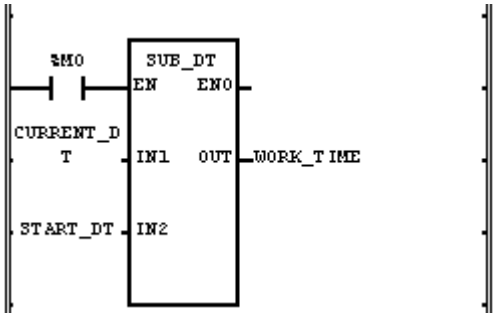
Subtract IN2(specific date and time) from IN1(reference date and time) and output the difference to OUT.

■ Error

If the output exceeds the range of TIME data type, _ERR and _LER flags are set.
If the result is negative, the error occurs.

■ Program example

LD



IL

```
LD          %M0
JMPN        PPP
LD          CURRENT_DT
SUB_DT  IN1:=  CURRENT RESULT
        IN2:=  START_DT
ST          WORK_TIME
PPP:
```

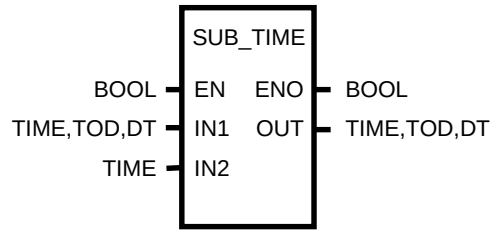
(1) If the execution condition(%M0) is On, SUB_DT(subtract TIME and DATE) function is executed.
(2) The input variable CURRENT_DT is DT#1995-12-15-14:30:00 and work start date and time START_DT is DT#1995-12-13-12:00:00, WORK_TIME outputs T#2D2H30M.

Input(IN1) : CURRENT_DT(DT) = DT#1995-12-15-14:30:00
(SUB_DATE)
(IN2) : START_DT(DT) = DT#1995-12-13-12:00:00
↓
Output (OUT) : WORK_TIME(TIME) = T#2D2H30M

SUB_TIME

Subtract time

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

Function	Description
	Input EN : Execute the function in case of 1 IN1 : Reference TIME, TOD, DT IN2 : TIME to be subtracted Output ENO : Output 1 in case of no error OUT : Subtracted TOD or TIME OUT type defends on input IN1 type. Therefore, if IN1 type is TIME, output OUT type is also TIME.

■ Function

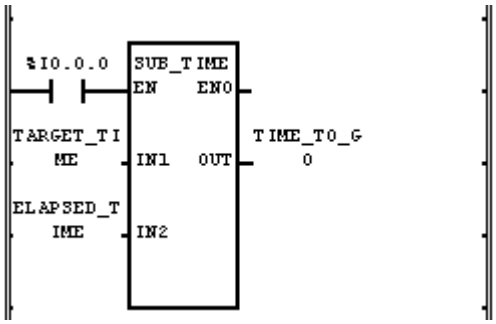
☐ If IN1 is TIME, subtract time from time and output the difference.
☐ If IN1 is TIME_OF_DAY, subtract the time from reference TOD and output TOD.
☐ If IN1 is DATE_AND_TIME, subtract the time from reference DT and time and output DT.

■ Error

If the output exceeds the range of respective data type, _ERR and _LER flags are set.
If the result is negative, the error occurs.

■ Program example

LD



IL

```
LD          %I0.0.0
JMPN        AAA
LD          TARGET_TIME
SUB_TIME    IN1:=  CURRENT RESULT
            IN2:=  ELAPSED_TIME
ST          TIME_TO_GO
AAA:
```

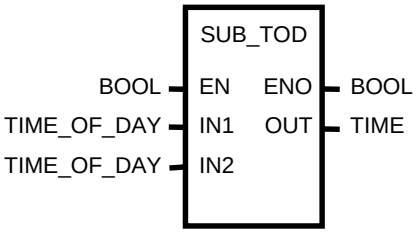
(1) If the execution condition(%I0.0.0) is On, SUB_TIME(subtract TIME) function is executed.
(2) If the input variable TARGET_TIME is T#2H30M and elapsed time ELAPSED_TIME is T#1H10M30S300MS, the output variable work time TIME_TO_G outputs T#1H19M29S700MS.

Input(IN1) : TARGET_TIME(TIME) = T#2H30M
(SUB_TIME)
(IN2) : ELAPSED_TIME(TIME) = T#1H10M30S300MS
↓
Output (OUT) : TIME_TO_GO(TIME) = T#1H19M29S700MS

SUB_TOD

Subtract TOD from TOD (Time of Day)

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

Function	Description
	**Input** EN : Execute the function in case of 1 IN1 : Reference TOD IN2 : TOD to be subtracted **Output** ENO : Output 1 in case of no error OUT : Subtracted TIME

■ **Function**

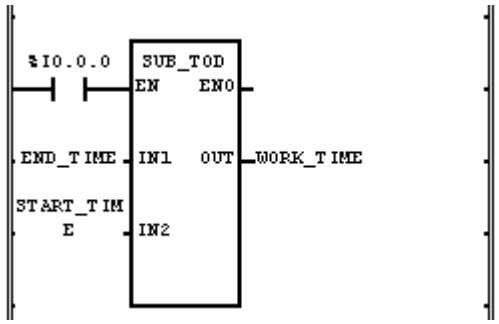
Subtract IN2(specific TOD) from IN1(reference TOD) and output the difference to OUT.

■ **Error**

If the result is negative, the error occurs.

■ **Program example**

LD



IL

```
LD          %I0.0.0
JMPN        AAA
LD          END_TIME
SUB_TOD  IN1:= CURRENT RESULT
         IN2:= START_TIME
ST          WORK_TIME
AAA:
```

(1) If the execution condition(%I0.0.0) is On, SUB_TOD(subtract TOD from TOD) function is executed.

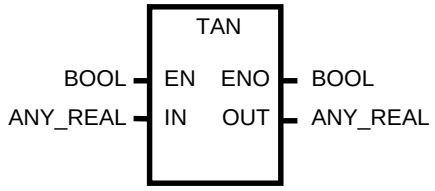
(2) IF the input variable END_TIME is TOD#14:20:30.5 and work start time START_TIME is TOD#12:00:00, output variable work time WORK_TIME outputs T#2H20M30S500MS.

Input(IN1) : END_TIME(TOD) = TOD#14:20:30.5
(SUB_TOD)
(IN2) : START_TIME(TOD) = TOD#12:00:00
↓
Output (OUT) : WORK_TIME(TIME) = T#2H20M30S500MS

TAN

Tangent operation

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

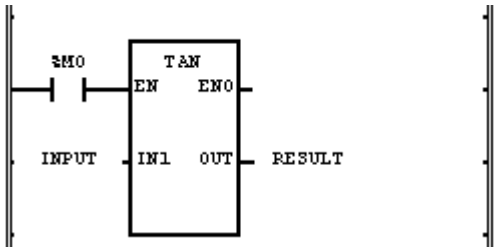
Function	Description
	**Input** EN : Execute the function in case of 1 IN : Tangent angle value(in radians) input **Output** ENO : Output EN value itself OUT : Tangent operation result IN and OUT shall be same type.

■ Function

Output Tangent value of IN to OUT.
OUT = TAN(IN)

■ Program example

LD



IL

```
LD      %M0
JMPN    BBB
LD      INPUT
TAN
ST      RESULT
BBB:
```

(1) If the execution condition(%M0) is On, TAN(Tangent operation) function is executed.
(2) IF the input variable INPUT is 0.7853 .... (□/4 rad = 45°, output variable RESULT will be 1.0000.

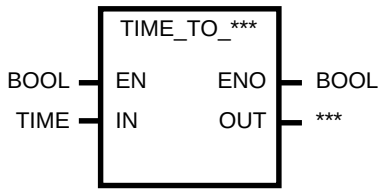
TAN(□/4) = 1

Input(IN1) : INPUT(REAL) = 0.7853
↓ (TAN)
Output(IN2) : RESULT(REAL) = 9.99803722E-01

TIME_TO_***

TIME type conversion

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

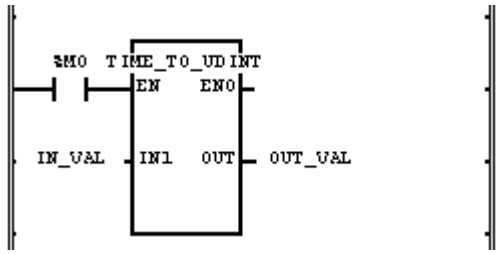
Function	Description
	**Input** EN : Execute the function in case of 1 IN : TIME data to be converted **Output** ENO : Output EN value itself OUT : Type converted data

■ Function

Convert IN to OUT data type.

FUNCTION	Output type	Description
TIME_TO_UDINT	UDINT	Convert TIME to UDINT type. Convert the data type without the conversion of internal bit data type.
TIME_TO_DWORD	DWORD	Convert TIME to DWORD type. Convert the data type without the conversion of internal bit data type.
TIME_TO_STRING	STRING	Convert TIME to STRING type.

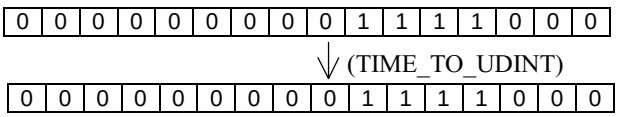
■ Program example

LD	IL
	LD %M0 JMPN AA LD IN_VAL TIME_TO_UDINT ST OUT_VAL AA :

(1) If the execution condition(%M0) is On, TIME_TO_UDINT function is executed.
(2) IF the input variable IN_VAL(TIME type) = T#120MS, output variable OUT_VAL(UDINT type) will be 120.

Input(IN1) : IN_VAL(TIME) = T#120MS(16#78)

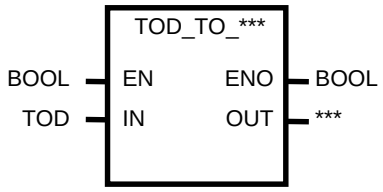
Output(OUT) : OUT_VAL(UDINT) = 120(16#78)



TOD_TO_***

TOD type conversion

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

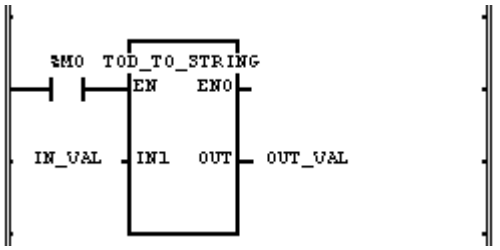
Function	Description
	**Input** EN : Execute the function in case of 1 IN : TOD to be converted **Output** ENO : Output EN value itself OUT : Type converted data

■ Function

Convert IN to OUT data type.

FUNCTION	Output type	Description
TOD_TO_UDINT	UDINT	Convert TOD to IDINT type. Convert data type only without conversion of internal bit array.
TOD_TO_DWORD	DWORD	Convert TOD to DWORD type. Convert data type only without conversion of internal bit array.
TOD_TO_STRING	STRING	Convert TOD to STRING type.

■ Program example

LD	IL
	LD % M0 JMPN AA LD IN_VAL DATE_TO_STRING ST OUT_VAL AA :

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

(2) IF the input variable IN_VAL(TOD type) is TOD#12:00:00, output variable OUT_VAL(STRING type) will be 'TOD#12:00:00'.

Input(IN1) : IN_VAL(TOD) = TOD#12:00:00

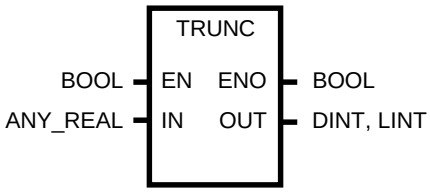
↓ (TOD_TO_STRING)

Output(IN2) : OUT_VAL(STRING) = 'TOD#12:00:00'

TRUNC

Integer conversion with round off (Truncate)

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

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

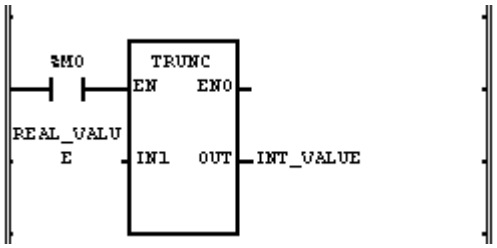
■ Function

FUNCTION	Input File	Output type	Description
TRUNC	REAL LREAL	DINT LINT	Round off the floating point and output the integer to OUT.

■ Error

If the converted value is greater than maximum value of out data type _ERR and _LER flags are set and result will be zero.

■ Program example

LD	IL
	LD % M0 JMPN XYZ LD REAL_VALUE TRUNC ST INT_VALUE XYZ :

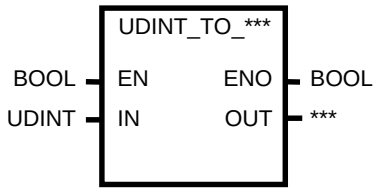
(1) If the execution condition(%M0) is On, TRUNC(Integer conversion with round off) function is executed.
(2) IF the input variable REAL_VALUE(REAL type) is 1.6, INT_VALUE(INT type) will be 1.
If REAL_VALUE(REAL type) is -1.6, INT_VALUE(INT type) will be -1.

Input(IN1) : REAL_VALUE(REAL) = 1.6
↓ (TRUNC)
Output(OUT) : INT_VALUE(INT) = 1

UDINT_TO_***

UDINT type conversion

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

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

■ Function

Convert IN to OUT data type.

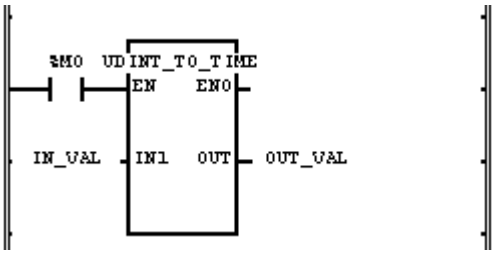
FUNCTION	Output type	Description
UDINT_TO_SINT	SINT	If input is 0 ~ 127, convert it normally. Otherwise, the error occurs.
UDINT_TO_INT	INT	If input is 0 ~ 32767, convert it normally. Otherwise, the error occurs.
UDINT_TO_DINT	DINT	If input is 0 ~ 2,147,483,64, convert it normally. Otherwise, the error occurs.
UDINT_TO_LINT	LINT	Convert UDINT to LINT type normally.
UDINT_TO_USINT	USINT	If input is 0 ~ 255, convert it normally. Otherwise, the error occurs.
UDINT_TO_UINT	UINT	If input is 0 ~ 65535, convert it normally. Otherwise, the error occurs.
UDINT_TO_ULINT	ULINT	Convert UDINT to ULINT type normally.
UDINT_TO_BOOL	BOOL	Convert lower 1 bit to BOOL type.
UDINT_TO_BYTE	BYTE	Convert lower 8 bit to BYTE type.
UDINT_TO_WORD	WORD	Convert lower 16 bit to WORD type.
UDINT_TO_DWORD	DWORD	Convert UDINT to DWORD type without conversion of internal bit array.
UDINT_TO_LWORD	LWORD	Convert upper bit to LWORD type filled with 0.
UDINT_TO_BCD	DWORD	If input is 0 ~ 99,999,999, convert it normally. Otherwise, the error occurs.
UDINT_TO_REAL	REAL	Convert UDINT to REAL type. The tolerance may be generated during converting by the precision.
UDINT_TO_LREAL	LREAL	Convert UDINT to LREAL type. The tolerance may be generated during converting by the precision.
UDINT_TO_TOD	TOD	Convert UDINT to TOD type without conversion of internal bit array.
UDINT_TO_TIME	TIME	Convert UDINT to TIME type without conversion of internal bit array.

■ Error

If the error occurs, _ERR and _LER flags are set.

Note If the error occurs, outputs bits from lower bit of IN as many as the bit of output type without the conversion of internal bit array.

■ **Program example**

LD	IL
	LD %M0 JMPN ZZ LD IN_VAL UDINT_TO_TIME ST OUT_VAL ZZ :

(1) If the input condition(%M0) is On, UDINT_TO_TIME function is executed.
(2) IF the input variable IN_VAL(UDINT type) is 123, output variable OUT_VAL(TIME type) will be T#123MS.

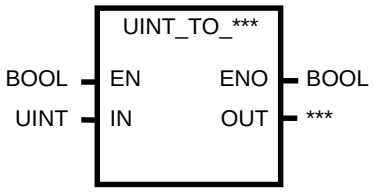
Input(IN1) : IN_VAL(UDINT) = 123

Output(OUT) : OUT_VAL(TIME) =T#123MS

UINT_TO_***

UINT type conversion

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

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

■ Function

Convert IN to OUT data type.

FUNCTION	Output type	Description
UINT_TO_SINT	SINT	If input is 0 ~ 127, convert it normally. Otherwise, the error occurs.
UINT_TO_INT	INT	If input is 0 ~ 32,767, convert it normally. Otherwise, the error occurs.
UINT_TO_DINT	DINT	Convert UINT to UDINT type normally.
UINT_TO_LINT	LINT	Convert UINT to ULINT type normally.
UINT_TO_USINT	USINT	If input is 0 ~ 255, convert it normally. Otherwise, the error occurs.
UINT_TO_UDINT	UDINT	Convert UINT to UDINT type normally.
UINT_TO_ULINT	ULINT	Convert UINT to ULINT type normally.
UINT_TO_BOOL	BOOL	Convert lower 1 bit to BOOL type.
UINT_TO_BYTE	BYTE	Convert lower 8 bit to BOOL type.
UINT_TO_WORD	WORD	Convert UINT to WORD type without conversion of internal bit array.
UINT_TO_DWORD	DWORD	Convert upper bit to WORD type filled with 0.
UINT_TO_LWORD	LWORD	Convert upper bit to LWORD type filled with 0.
UINT_TO_BCD	BCD	If input is 0 ~ 99,999,999, convert it normally. Otherwise, the error occurs.
UINT_TO_REAL	REAL	Convert UINT to REAL type.
UINT_TO_LREAL	LREAL	Convert UINT to LREAL type.
UNIT_TO_DATE	DATE	Convert UINT to DATE type without conversion of internal bit array.

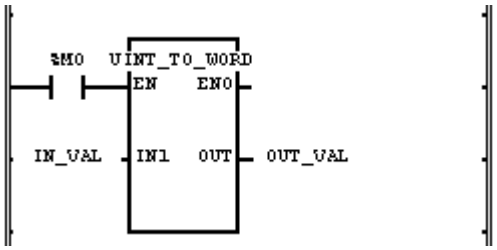
■ Error

If the error occurs, _ERR and _LER flags are set.

Note If the error occurs, outputs bits from lower bit of IN as many as the bit of output type without the conversion of internal bit array.

■ **Program example**

LD



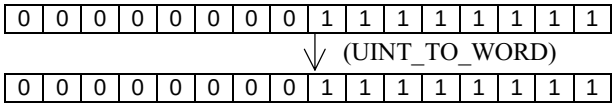
IL

```
LD          %M0
JMPN        PO
LD          IN_VAL
UINT_TO_WORD
ST          OUT_VAL
PO :
```

(1) If the execution condition(%M0) is On, UINT_TO_WORD function is executed.
(2) IF the input variable IN_VAL(UINT type) is 255(2#0000_0000_1111_1111), OUT_VAL(WORD type) is 2#0000_0000_1111_1111.

Input(IN1) : IN_VAL(UINT) = 255

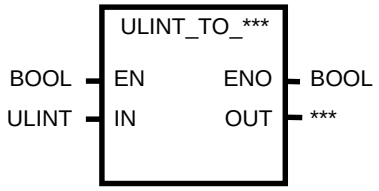
Output(OUT) : OUT_VAL(WORD) = 16#FF



ULINT_TO_***

ULINT type conversion

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

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

■ Function

Convert IN to OUT data type.

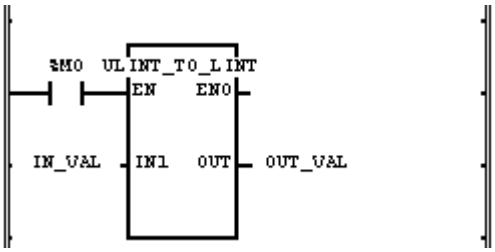
FUNCTION	Output type	Description
ULINT_TO_SINT	SINT	If input is 0 ~ 127, convert it normally. Otherwise, the error occurs.
ULINT_TO_INT	INT	If input is 0 ~ 32,767, convert it normally. Otherwise, the error occurs.
ULINT_TO_DINT	DINT	If input is 0 ~ 2^{31}-1, convert it normally. Otherwise, the error occurs.
ULINT_TO_LINT	LINT	If input is 0 ~ 2^{63}-1, convert it normally. Otherwise, the error occurs.
ULINT_TO_USINT	USINT	If input is 0 ~ 255, convert it normally. Otherwise, the error occurs.
ULINT_TO_UINT	UINT	If input is 0 ~ 65,535, convert it normally. Otherwise, the error occurs.
ULINT_TO_UDINT	UDINT	If input is 0 ~ 2^{32}-1, convert it normally. Otherwise, the error occurs.
ULINT_TO_BOOL	BOOL	Convert lower 1 bit to BOOL type.
ULINT_TO_BYTE	BYTE	Convert lower 8 bit to BYTE type.
ULINT_TO_WORD	WORD	Convert lower 1 bit to WORD type.
ULINT_TO_DWORD	DWORD	Convert lower 32 bit to DWORD type.
ULINT_TO_LWORD	LWORD	Convert ULINT to LWORD type without conversion of internal bit array.
ULINT_TO_BCD	BCD	If input is 0 ~ 9,999,999,999,999,999, convert it normally. Otherwise, the error occurs.
ULINT_TO_REAL	REAL	Convert ULINT to REAL type. The tolerance may be generated during converting by the precision.
ULINT_TO_LREAL	LREAL	Convert ULINT to LREAL type. The tolerance may be generated during converting by the precision.

■ Error

If the error occurs, _ERR and _LER flags are set.

Note If the error occurs, outputs bits from lower bit of IN as many as the bit of output type without the conversion of internal bit array.

■ **Program example**

LD	IL
	LD %M0 JMPN PP LD IN_VAL ULINT_TO_LINT ST OUT_VAL PP :

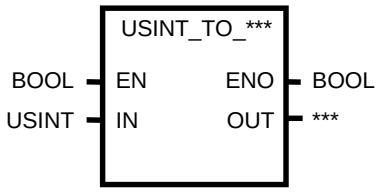
(1) If the execution condition(%M0) is On, ULINT_TO_LINT function is executed.
(2) IF the input variable IN_VAL(ULINT type) is 123,567,899, output variable OUT_VAL(LINT type) will be 123,567,899.

Input(IN1) : IN_VAL(ULINT) = 123,567,899
↓(ULINT_TO_LINT)
Output(OUT) : OUT_VAL(LINT) = 123,567,899

USINT_TO_***

USINT type conversion

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

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

■ Function

Convert IN to OUT data type.

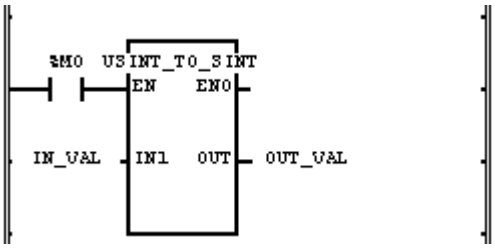
FUNCTION	Output type	Description
USINT_TO_SINT	SINT	If input is 0 ~ 127, convert it normally. Otherwise, the error occurs.
USINT_TO_INT	INT	Convert input to INT type.
USINT_TO_DINT	DINT	Convert USINT to DINT type normally.
USINT_TO_LINT	LINT	Convert input to LINT type.
USINT_TO_UINT	UINT	Convert input to UINT type.
USINT_TO_UDINT	UDINT	Convert input to UDINT type.
USINT_TO_ULINT	ULINT	Convert input to ULINT type.
USINT_TO_BOOL	BOOL	Convert lower 1 bit to BOOL type.
USINT_TO_BYTE	BYTE	Convert USINT to BYTE type without conversion of internal bit array.
USINT_TO_WORD	WORD	Convert upper bit to WORD type filled with 0.
USINT_TO_DWORD	DWORD	Convert upper bit to DWORD type filled with 0.
USINT_TO_LWORD	LWORD	Convert upper bit to LWORD type filled with 0.
USINT_TO_BCD	BCD	If input is 0 ~ 99, convert it normally. Otherwise, the error occurs.
USINT_TO_REAL	REAL	Convert USINT to REAL type.
USINT_TO_LREAL	LREAL	Convert USINT to LREAL type.

■ Error

If the conversion error occurs, _ERR and _LER flags are set.

Note If the error occurs, outputs bits from lower bit of IN as many as the bit of output type without the conversion of internal bit array.

■ **Program example**

LD	IL
	LD %M0 JMPN LL LD IN_VAL USINT_TO_SINT ST OUT_VAL LL :

(1) If the execution condition(%M0) is On, ULINT_TO_SINT function is executed.
(2) IF the input variable IN_VAL(USINT type) is 123, output variable OUT_VAL(SINT type) will be 123.

Input(IN1) : IN_VAL(USINT) = 123(16#7B)

0	1	1	1	1	0	1	1

↓ (ULINT_TO_SINT)

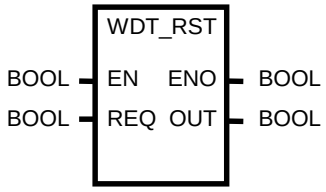
Output(OUT) : OUT_VAL(SINT) = 123(16#7B)

0	1	1	1	1	0	1	0

WDT_RST

Watch_Dog timer reset

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

Function	Description
	**Input** EN : Execute the function in case of 1 IN : Watch_Dog timer reset request **Output** ENO : Output EN value itself OUT : Output 1 after reset Watch_Dog timer

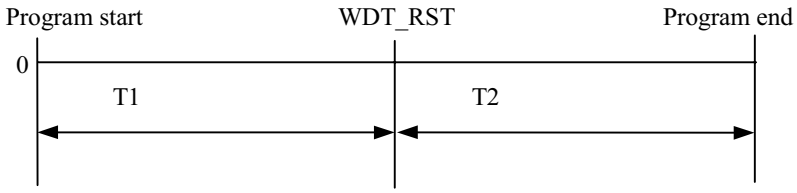
■ Function

☐ Reset Watch-Dog Timer in the program.

☐ The program uses WDT_RST, if the scan time may exceeds the defined Watch-Dog Time.

☐ When scan time exceeds Watch Dog Timer frequently, please change watch dog time set value of basic parameter section of GMWIN the programming software.

☐ Both T1 from 0 Line to WDT-RST function and T2 from WDT-RST function to end of program shall not exceed SCAN Watch_Dog time setting value.

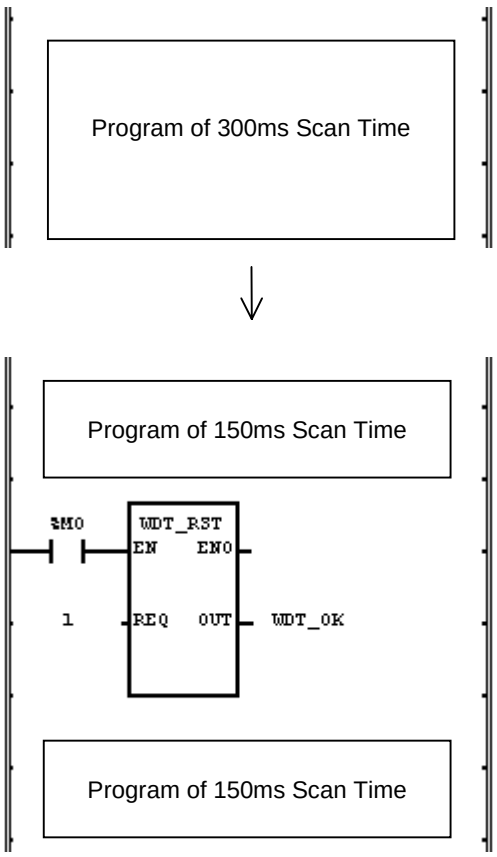


-WDT_RST function can be used several times at 1 scan.

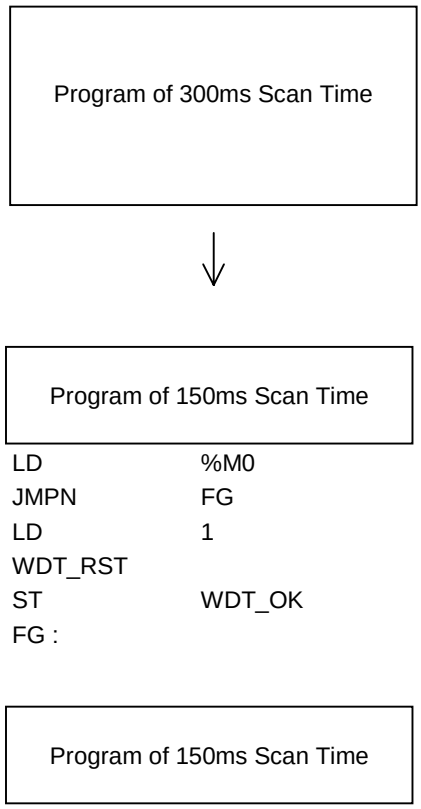
■ **Program example**

Program that the program running time is 300ms according to the execution condition of 200ms Scan Watch_Dog time.

LD



IL



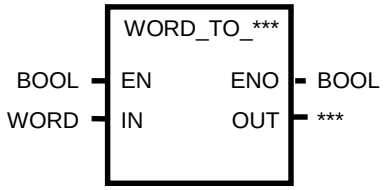
(1) If the execution condition(%M0) is On, WDT-RST(Watch Dog timer initialization) function is executed.

(2) IF WDT-RST function is executed, the program of 300ms Scan time can be run with no interference when watch dog time set value is 200ms.

WORD_TO_***

WORD type conversion

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

Function	Description
	**Input** EN : Execute the function in case of 1 IN : Bit array(16Bit) to be converted **Output** ENO : Output EN value itself OUT : Type converted data

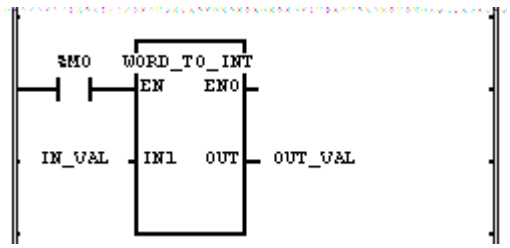
■ Function

Convert IN to OUT data type.

FUNCTION	Output type	Description
WORD _TO_SINT	SINT	Convert lower 8 bit to SINT type.
WORD _TO_INT	INT	Convert WORD to INT type without conversion of internal bit array.
WORD _TO_DINT	DINT	Convert upper bit to DINT type filled with 0.
WORD _TO_LINT	LINT	Convert upper bit to LINT type filled with 0.
WORD _TO_USINT	USINT	Convert lower 8 bit to SINT type.
WORD _TO_UINT	UINT	Convert WORD to INT type without conversion of internal bit array.
WORD _TO_UDINT	UDINT	Convert upper bit to DINT type filled with 0.
WORD _TO_ULINT	ULINT	Convert upper bit to LINT type filled with 0.
WORD _TO_BOOL	BOOL	Convert lower 1 bit to BOOL type.
WORD _TO_BYTE	BYTE	Convert lower 8 bit to SINT type.
WORD _TO_DWORD	DWORD	Convert upper bit to DWORD type filled with 0.
WORD _TO_LWORD	LWORD	Convert upper bit to LWORD type filled with 0.
WORD _TO_DATE	DATE	Convert WORD to DATE type without conversion of internal bit array.
WORD _TO_STRING	STRING	Convert WORD to STRING type.

■ Program example

LD



IL

```
LD          %M0
JMPN        P0
LD          IN_VAL
WORD_TO_INT
ST          OUT_VAL
PO:
```

(1) If the execution condition(%M0) is On, WORD-TO-INT function is executed.

(2) IF the input variable IN_VAL(WORD type) is 2#0001_0001_0001_0001, output variable OUT_VAL(INT type) will be 4096 + 256 + 16 + 1 = 4,369.

Input(IN1) : IN_VAL(WORD) = 16#1111

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

(WORD_TO_INT)

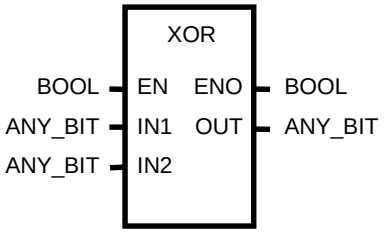
0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

Output(OUT) : OUT_VAL(INT) = 4,369(16#1111)

XOR

Logical XOR

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

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

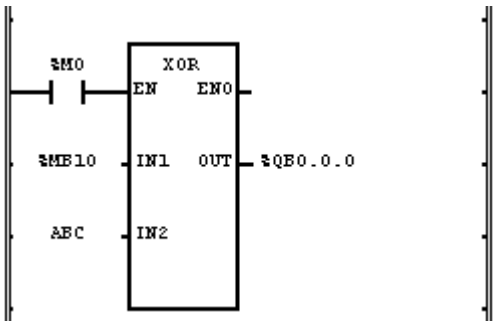
■ Function

Execute XOR of IN1 and IN2 and output the result to OUT.

IN1 1111 0000
XOR
IN2 1010 1010
OUT 0101 1010

■ Program example

LD



IL

```
LD          %M0
JMPN        ZZ
LD          %MB10
XOR   IN1:=  CURRENT RESULT
      IN2:=  ABC
ST          %QB0.0.0
ZZ :
```

(1) If the execution condition(%M0) is On, XOR(exclusive logical sum) function is executed.
(2) IF the input variable %MB10 is 11001100 and ABC is 11110000, XOR result of output variable %QB0.0.0 will be 00111100.

Input(IN1) : %MB10(BYTE) = 16#CC

(IN2) : ABC(BYTE) = 16#F0

Output (OUT) : %QB0.0.0(BYTE) = 16#3C

1	1	0	0	1	1	0	0

(XOR)

1	1	1	1	0	0	0	0

↓

0	0	1	1	1	1	0	0