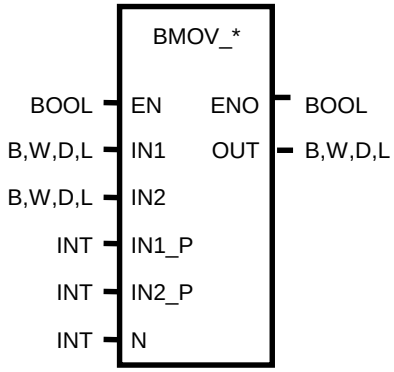
8.3 MK(MASTER-K) function libraries

Each MK function library is described.
MK function libraries mean the librarized command used in Master-K series.

BMOV_B,W,D,L

Copy or Move the part of bit string

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

Function	Description
	**Input** EN : Execute the function in case of 1 IN1 : String data having bit data to be combined IN2 : String data having bit data to be combined IN1_P : Start bit position on IN1 set data IN2_P : Start bit position on IN2 set data N : Bit number to be combined **Output** ENO : Output 1 in case of no error OUT : Combined bit string data output

■ Function

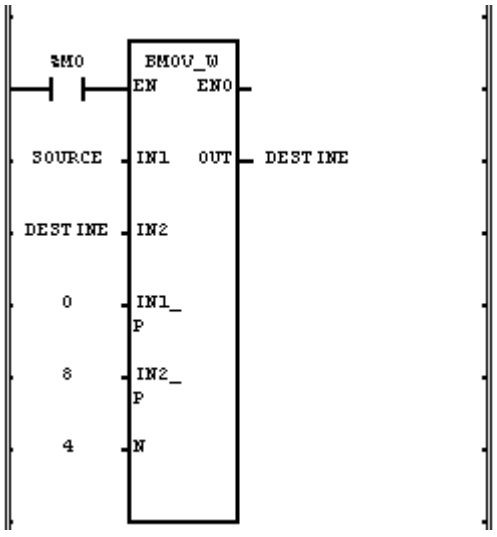
If EN is 1, take N bits from the bit position assigned by IN1_P to left direction among IN1 bit string and replace it to IN2 bit string, replaced position is assigned by IN2_P and replacement direction is from right to left.
If IN1=1111 0000 1111 0000 and IN2=0000 1010 1010 1111 and IN1_P=4 and IN2_P=8 and N=4, output data OUT=0000111110101111. The input can access to B(BYTE), W(WORD), D(DWORD) and L(LWORD) type data and L(LWORD) applies to GM1,2 only.
One of 'BMOV_B', 'BMOV _W', 'BMOV _D', 'BMOV _L' function can be selected according to input data.

■ Error

If IN1_P and IN2_P exceed the data range or N is negative or N bit of IN1_P and IN2_P exceeds the data range, _ERR and _LER flags are set.

■ Program example

LD



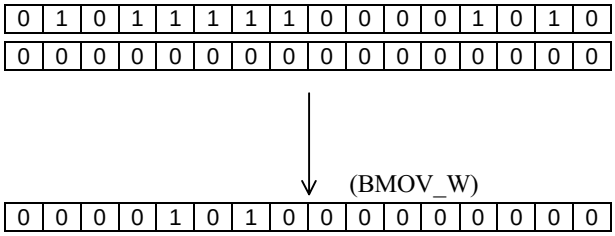
IL

```
LD          %M0
JMPN        LSB
LD          SOURCE
BMOV_W  IN1:=   CURRENT RESULT
        IN2:=   DESTINE
        IN_1P:= 0
        IN_2P:= 8
        N:=     4
ST          DESTINE
LSB
```

(1) If the execution condition(%M0) is On, BMOV_W function is executed.
(2) As input variable SOURCE=2#0101 1111 0000 1010 and DESTINE=2#0000 0000 0000 0000 and IN1_P=0 and IN2_P=8 and N=4, the operation result is 2#0000 1010 0000 0000 and DESTINE=2#0000 1010 0000 0000 since output is set to DESTINE.

Input (IN1) : SOURCE(WORD) = 16#5F0A
(IN2) : DESTINE(WORD) = 16#0000
(IN1_P) = 0
(IN2_P) = 8
(N) = 4

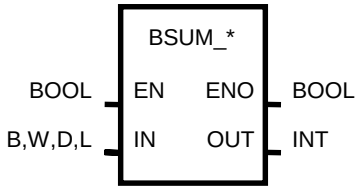
Output(OUT) : DESTINE(WORD) = 16#0A00



BSUM_B,W,D,L

Output ON bit number by digit

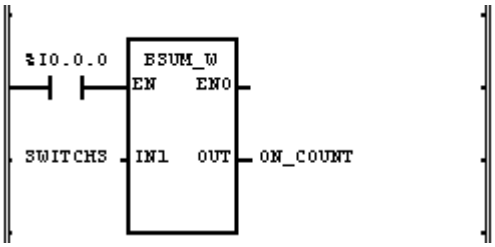
Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

Function	Description
	**Input** EN : Execute the function in case of 1 IN : Input data to detect ON bit **Output** ENO : Output EN value itself OUT : Result data sums of ON bit number

■ Function

If EN is 1, count Bit number of 1 among IN bit string and output the result to OUT.
The input can access to B(BYTE), W(WORD), D(DWORD) and L(LWORD) type data and L(LWORD) applies to GM1,2 only.
One of 'BSUM_B', 'BSUM_W', 'BSUM_D' and 'BSUM_L' function can be selected according to input data.

■ Program example

LD	IL
	LD %I0.0.0 JMPN AAA LD SWITCHS BSUM_W ST ON_COUNT AAA:

(1) If the execution condition(%I0.0.0) is On, BSUM_W function is executed.
(2) If input variable SWITCHS(WORD type)=2#0000 0100 0010 1000, output ON bit number, i.e. 3, and store integer value '3' to ON_COUNT(INT type).

Input(IN1) : SWITCHS(WORD) = 16#428

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

BSUM_W

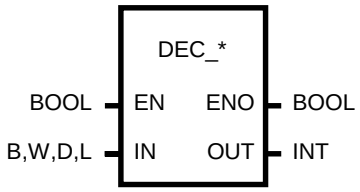
Output(OUT) : ON_COUNT(INT) = 16#3

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

DEC_B,W,D,L

Decrease IN data by 1

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

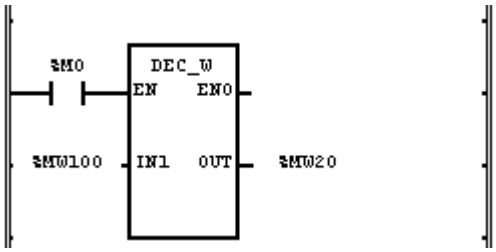
Function	Description
	**Input** EN : Execute the function in case of 1 IN : Input data to be decreased **Output** ENO : Output EN value itself OUT : Result data decreased

■ Function

If EN is 1, decrease IN data by 1 and output the result to OUT.
Though the underflow occurs, the error is not generated and the result will be 16#FFFF if the IN is 16#0000.
The input can access to B(BYTE), W(WORD), D(DWORD) and L(LWORD) type data and L(LWORD) applies to GM1,2 only.
One of 'DEC_B', 'DEC_W', 'DEC_D' and 'DEC_L' function can be selected according to input data.

■ Program example

LD



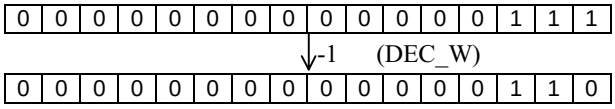
IL

```
LD      %M0
JMPN    KKK
LD      %MW100
DEC_W
ST      %MW20
KKK:
```

(1) If the execution condition(%M0) is On, DEC_W function is executed.
(2) If input variable %MW100=16#0007(2#0000 0000 0000 0111), %MW20=16#0006(2#0000 0000 0000 0110 after operation.

Input(IN1) : %MW100(WORD) = 16#0007

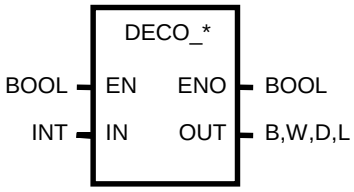
Output(OUT) : %MW20(WORD) = 16#0006



DECO_B,W,D,L

Set assigned bit position to ON

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

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

■ Function

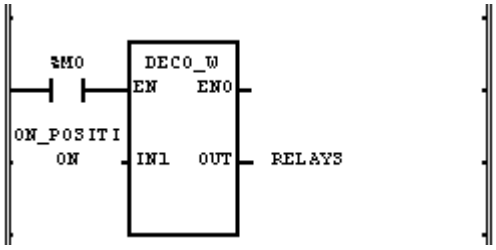
If EN is 1, outputs the bit string data of OUT type with assigned bit position set to 1, the bit position is assigned by IN value.
The input can access to B(BYTE), W(WORD), D(DWORD) and L(LWORD) type data and L(LWORD) applies to GM1,2 only.
One of 'DECO_B', 'DECO_W', 'DECO_D' and 'DECO_L' function can be selected according to input data.

■ Error

If input data is negative or bit location assign data exceed the bit limit of output type(over 16 in case of DECO_W), OUT will be 0 and _ERR and _LER flags are set.

■ Program example

LD



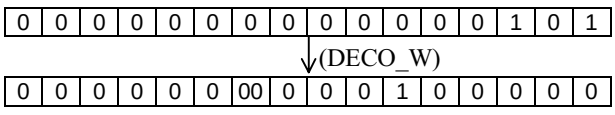
IL

```
LD      %M0
JMPN    AAA
LD      ON_POSITION
DECO_W
ST      RELAYS
AAA:
```

(1) If the execution condition(%M0) is On, DECO_W function is executed.
(2) If input variable ON_POSITION(INT type) = 5, RELAYS(WORD type) = 2# 0000 0000 0010 0000 since only No. 5 bit of output is on.

Input(IN1) : ON_POSITION(INT) = 5(16#5)

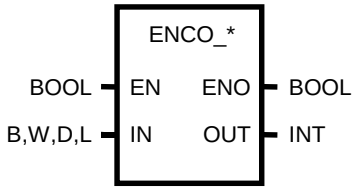
Output(OUT) : RELAYS(WORD) = 16#20



ENCO_B,W,D,L

Set assigned bit position to ON

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

Function	Description
	**Input** EN : Execute the function in case of 1 IN : Input data to be incoded **Output** ENO : Output 1 in case of no error OUT : Result data decoded

■ Function

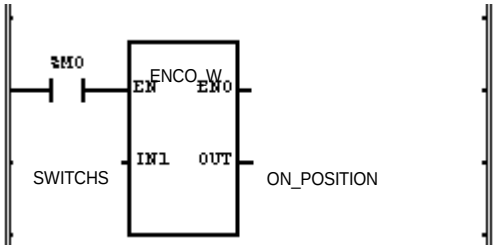
If EN is 1, output to OUT the highest bit position of 1 among IN bit string.
The input can access to B(BYTE), W(WORD), D(DWORD) and L(LWORD) type data and L(LWORD) applies to GM1, 2 only.
One of 'ENCO_B', 'ENCO_W', 'ENCO_D' and 'ENCO_L' function can be selected according to input data.

■ Error

If No bit is 1 among input data, Out turns 1, _ERR, _LER flag become three.

■ Program example

LD



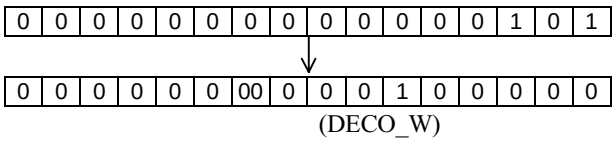
IL

```
LD      %M0
JMPN    AAA
LD      SWITCHS
ENCO_W
ST      ON_POSITION
AAA:
```

(1) If the execution condition(%M0) is On, ENCO_W function is executed.
(2) If SWITCHS(WORD type) = 2#000 1000 0010, output the position of 2 bits under On, in other words, '11' that is in upper position between '11' and '1', and store integer value '3' to ON_POSITION(INT type)

Input(IN1) : SWITCHS(WORD) = 16#802

Output(OUT) : ON_POSITION(INT) = 11(16#B)

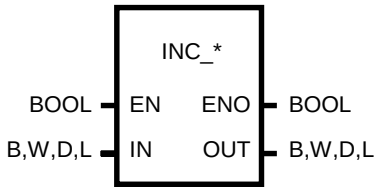


(DECO_W)

INC_B,W,D,L

Increase IN data by 1

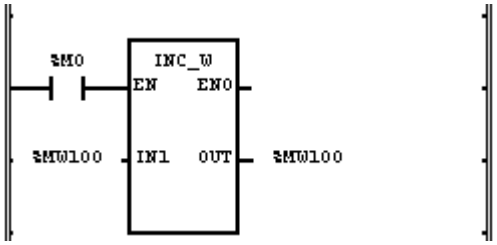
Product	GM1	GM2	GM3	GM4	GM5
Applicable	●	●	●	●	●

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

■ Function

If EN is 1, increase IN data by 1 and output the result to OUT.
There is no error when overflow occurs, the result will be 16#000 in case of 16#FFFF.
One of 'INC_B', 'INC _W', 'INC_D' and 'INC_L' function can be selected according to input data.
Data types are B(BYTE), W(WORD), D(DWORD), L(LWORD), L(LWORD), is only in GM1, 2.

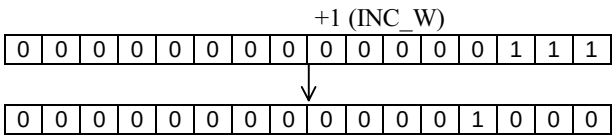
■ Program example

LD	IL
	LD %M0 JMPN BBB LD %MW100 INC_W ST %MW100 BBB:

(1) If the execution condition(%M0) is On, INC_W function is executed.
(2) If input variable %MW100=16#0007(2#0000 0000 0000 0111), %MW100=16#0008(2#0000 0000 0000 1000) after operation.

Input(IN1) : %MW100(WORD)=16#0007

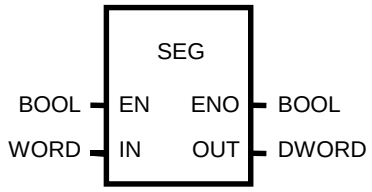
Output(OUT) : %MW100(WORD)=16#0008



SEG

Convert BCD or HEX value to 7 segment display code

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

Function	Description
	**Input** EN : Execute the function in case of 1 IN : Input data to be converted to 7 segment code **Output** ENO : Output EN value itself OUT : Result data converted to 7 segment code

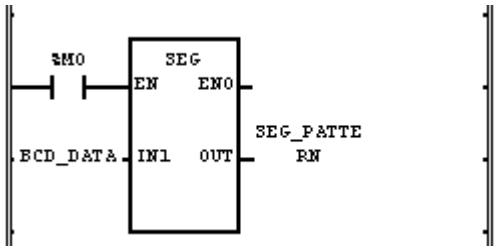
■ Function

If EN is 1, convert BCD or HEX(hexadecimal) digit to the code for 7 segment display as below table and output the result to OUT. The value from 0000 to 9999 can be displayed to four 7 segment in case of BCD input and the value from 0000 to FFFF can be displayed to four 7 segment in case of HEX input.

Display example

1) 4 position BCD -> 4 position 7 segment code: Use 'SEG' function
2) 4 position HEX -> 4 position 7 segment code: Use 'SEG' function
3) Integer -> 7 segment code of 4 position BCD type: Use 'SEG' function after 'INT_TO_BCD' function
4) Integer -> 7 segment code of 4 position HEX type: Use 'SEG' function after 'INT_TO_WORD' function
5) When to convert over 4 position digit,
   A) In case of BCD or HEX, split the digit by 4 position and apply 'SEG' to each 4 position fragment.
   B) Integer to 8 position BCD 7 segment code:
      Divide the integer by 10,000 and convert the quotient and remainder to BCD through 'INT_TO_BCD' function. After this, convert each BCD to lower 4 and upper 4 position 7 segment code.

■ Program example

LD	IL
	LD %M0 JMPN BBB LD BCD_DATA SEG ST SEG_PATTERN BBB:

(1) If the execution condition(%M0) is On, SEC_W function is executed.
(2) If input variable BCD_DATA(WORD type) = 16#1234, output '2#00000110_01011011_01001111_01100110' of '1234' display in 7 segment display and store it in SEG_PATTERN(DWORD type).

Input(IN1) : BCD_DATA(WORD) = 16#1234

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

(SEG)

Output(OUT) : SEG_PATTERN(DWORD) = 16#065B4F66

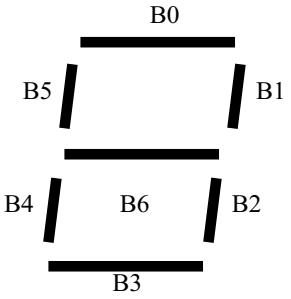
Upper

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

Lower

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

7 segment configuration



7 segment code conversion table

Input (BCD)	Input (Hexadecimal)	Integer value	Output B7	B6	B5	B4	B3	B2	B1	B0	Display data
0	0	0	0	0	1	1	1	1	1	1	***0***
1	1	1	0	0	0	0	0	1	1	0	***1***
2	2	2	0	1	0	1	1	0	1	1	***2***
3	3	3	0	1	0	0	1	1	1	1	***3***
4	4	4	0	1	1	0	0	1	1	0	***4***
5	5	5	0	1	1	0	1	1	0	1	***5***
6	6	6	0	1	1	1	1	1	0	1	***6***
7	7	7	0	0	1	0	0	1	1	1	***7***
8	8	8	0	1	1	1	1	1	1	1	***8***
9	9	9	0	1	1	0	1	1	1	1	***9***
	A	10	0	1	1	1	0	1	1	1	***A***
	B	11	0	1	1	1	1	1	0	0	***B***
	C	12	0	0	1	1	1	0	0	1	***C***
	D	13	0	1	0	1	1	1	1	0	***D***
	E	14	0	1	1	1	1	0	0	1	***E***
	F	15	0	1	1	1	0	0	0	1	***F***