

Instruction Manual of Samkoon PLC

V1.0.0

SHENZHEN SAMKOON TECHNOLOGY CORPORATION LTD.

Bit Logic Instructions

Notes


1. Instructions in this chapter do bit logic operations to the corresponding bit-register.
2. The bit-registers have several types, details refer to [Table 1.1.1](#).

Register type	Description	Numbering system	Range
X	Input bit-register of PLC, some of them correspond to actual input contacts, and the rest work like inner bit-register.	Octal	■ FGs_16MR/FGs_16MT 0~77 ■ Else 0~177
Y	Output bit-register of PLC, some of them correspond to actual output contacts, and the rest work like inner bit-register.	Octal	■ FGs_16MR/FGs_16MT 0~77 ■ Else 0~177
M	Inner bit-register of PLC.	Decimal	0~8223
T	Flag bit of correspond timer, set when timer reaches the target number, can be used to reset the timer.	Decimal	0~255
C	Flag bit of correspond counter, set when counter reach the target number, can be used to reset the counter.	Decimal	0~255
S	System state bit-register, used in program control.	Decimal	0~999

Table 1.1. 1

LD/AND/OR

Instruction introduction

1. This instruction is NO (normally open) contact operation start instruction, it sets specified bit register as input contact. When value of the bit register is 1, contact closes; When value of the bit register is 0, contact opens.
2. This instruction shows as  in the ladder diagram.
3. In instruction table, input contacts can also be connected through **AND** (NO contact series connection instruction) and **OR** (NO contact parallel connection instruction) (refer to [example](#)).

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(s)	Bit register used as NO input contact	X/Y/M/T/C/S	-	bool

Table 1.2. 1

Example

Instruction table:

Network 000

```
LD      X000 //add normal open coil X000, coil will be closed when X000 is ON
OR      X002 //add normal open coil X002 to make parallel connection with X000
AND     X001 //add normal open coil X001 to make series connection with X000
OUT     Y000 //add Y000 as output contact
```

Ladder diagram:



Figure 1.2. 1

LDIM/ANDIM/ORIM

Instruction introduction

1. This instruction is immediate NO (normally open) contact operation start instruction, it sets specified bit register as input contact. When value of the bit register is 1, contact closes immediately; When value of the bit register is 0, contact opens immediately, and the refresh of contact doesn't rely on scan period of PLC.
2. This instruction shows as —| |— in the ladder diagram.
3. In instruction table, input contacts can also be connected through **ANDIM** (immediate NO contact series connection instruction) and **ORIM** (immediate NO contact parallel connection instruction) (refer to [example](#)).

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(s)	Bit register used as immediate NO input contact	X/Y/M/T/C/S	-	bool

Table 1.3. 1

Example

Instruction table:

Network 000

```

LDIM    X000 //add immediate normally open coil X000
ORIM    X002 //add immediate normally open coil X002 to make parallel connection with X000
ANDIM   X001 //add immediate normally open coil X001 to make series connection with X000
OUT     Y000 //add Y000 as output contact
    
```

Ladder diagram:

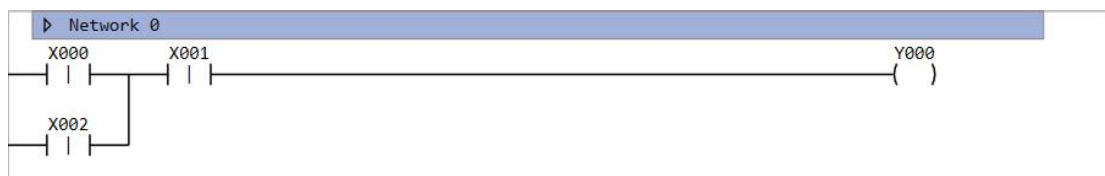



Figure 1.3. 1

LDI/ANDI/ORI

Instruction introduction

1. This instruction is NC (normally close) contact operation start instruction, it sets specified bit register as input contact. When value of the bit register is 1, contact opens; When value of the bit register is 0, contact closes.
2. This instruction shows as  in the ladder diagram.
3. In instruction table, input contacts can also be connected through **ANDI** (NC contact series connection instruction) and **ORI** (NC contact parallel connection instruction) (refer to [example](#)).

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(s)	Bit register used as NC input contact	X/Y/M/T/C/S	-	bool

Table 1.4. 1

Example

Instruction table:

Network 000

```
LDI    X000 //add normally close coil X000
ORI    X002 //add normally close coil X002 to make parallel connection with X000
ANDI   X001 //add normally close coil X001 to make series connection with X000
OUT    Y000 //add Y000 as output contact
```

Ladder diagram:

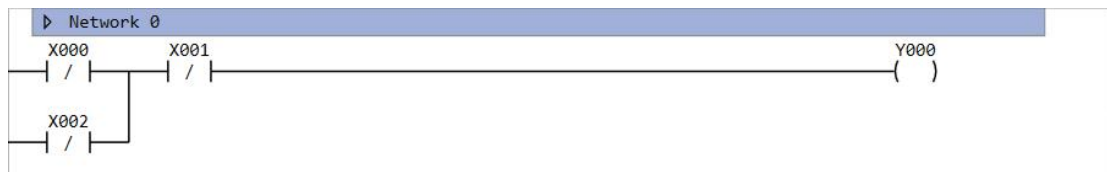



Figure 1.4. 1

LDIIM/ANDIIM/ORIIM

Instruction introduction

1. This instruction is immediate NC (normally close) contact operation start instruction, it sets specified bit register as input contact. When value of the bit register is 1, contact opens immediately; When value of the bit register is 0, contact closes immediately, and the refresh of contact doesn't rely on scan period of PLC.
2. This instruction shows as  in the ladder diagram.
3. In instruction table, input contacts can also be connected through **ANDIIM** (immediate NC contact series connection instruction) and **ORIIM** (immediate NC contact parallel connection instruction) (refer to [example](#)).

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(s)	Bit register used as immediate NC input contact	X/Y/M/T/C/S	-	bool

Table 1.5. 1

Example

Instruction table:

Network 000

```

LDIIM    X000 //add immediate normally close coil X000
ORIIM    X002 //add immediate normally close coil X002 to make parallel connection with X000
ANDIIM   X001 //add immediate normally close coil X001 to make series connection with X000
OUT      Y000 //add Y000 as output contact
    
```

Ladder diagram:



Figure 1.5. 1

LDP/ANDP/ORP

Instruction introduction

1. This instruction is rising edge pulse operation start instruction, it sets specified bit register as input contact. Contact closes for a scan period only at rising edge (OFF to ON) of the bit register.
2. This instruction shows as $\text{—}|\uparrow\text{—}$ in the ladder diagram.
3. In instruction table, input contacts can also be connected through **ANDP** (rising edge pulse series connection instruction) and **ORP** (rising edge pulse parallel connection instruction) (refer to [example](#)).

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(s)	Bit register used as rising edge pulse input contact	X/Y/M/T/C/S	-	bool

Table 1.6. 1

Example

Instruction table:

Network 000

```
LD      X000 //add normal open coil X000, coil will be closed when X000 is ON
OR      X002 //add normal open coil X002 to make parallel connection with X000
ANDP   X001 //add rising edge pulse coil X001 to make series connection with X000
SET     Y000      K1 //set Y000
```

Ladder diagram:

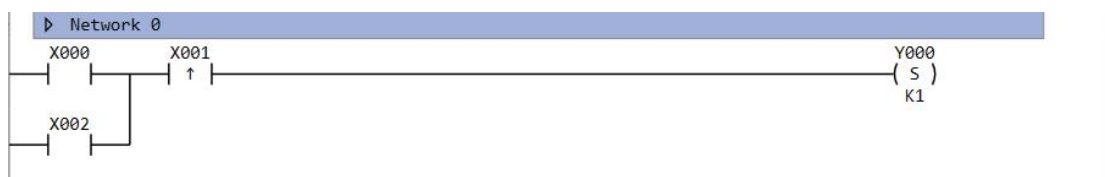


Figure 1.6. 1

LDF/ANDF/ORF

Instruction introduction

1. This instruction is falling edge pulse operation start instruction, it sets specified bit register as input contact. Contact closes for a scan period only at falling edge (ON to OFF) of the bit register.
2. This instruction shows as $\text{—}|\downarrow\text{—}$ in the ladder diagram.
3. In instruction table, input contacts can also be connected through **ANDF** (falling edge pulse series connection instruction) and **ORF** (falling edge pulse parallel connection instruction) (refer to [example](#)).

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(s)	Bit register used as falling edge pulse input contact	X/Y/M/T/C/S	-	bool

Table 1.7. 1

Example

Instruction table:

Network 000

```

LD      X000 //add normal open coil X000, coil will be closed when X000 is ON
OR      X002 //add normal open coil X002 to make parallel connection with X000
ANDP    X001 //add falling edge pulse coil X001 to make series connection with X000
SET     Y000      K1 //set Y000
  
```

Ladder diagram:

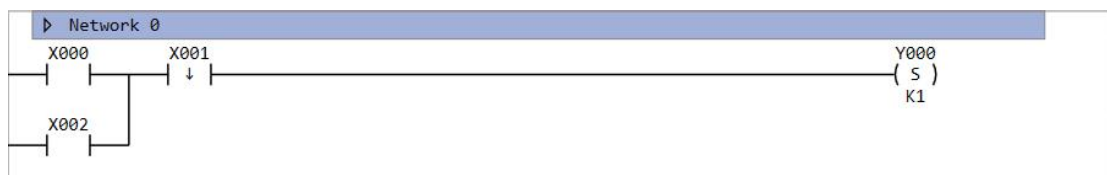


Figure 1.7. 1

MEP

Instruction introduction

1. This instruction turns ON (conductive state) for a scan period at the rising edge (OFF to ON) of the ahead operation result. This instruction turns OFF (non-conductive state) in instances other than the rising edge of the ahead operation result.
2. This instruction shows as $\text{---}\uparrow\text{---}$ in the ladder diagram.

Example

Instruction table:

Network 000

```
LD      X000 //add normal open coil X000, coil will be closed when X000 is ON
OR      X002 //add normal open coil X002 to make parallel connection with X000
MEP //turn ON at rising edge of ahead operation result
SET     Y000      K1 //set Y000
```

Ladder diagram:



Figure 1.8. 1

MEF

Instruction introduction

1. This instruction turns ON (conductive state) for a scan period at the falling edge (ON to OFF) of the ahead operation result. This instruction turns OFF (non-conductive state) in instances other than the falling edge of the ahead operation result.
2. This instruction shows as $\text{---}\downarrow\text{---}$ in the ladder diagram.

Example

Instruction table:

Network 000

```
LD      X000 //add normal open coil X000, coil will be closed when X000 is ON
OR      X002 //add normal open coil X002 to make parallel connection with X000
MEF //turn ON at falling edge of ahead operation result
SET     Y000      K1 //set Y000
```

Ladder diagram:



Figure 1.9. 1

INV

Instruction introduction

1. This instruction inverts the operation result before it.
2. This instruction shows as $\text{—}/\text{—}$ in the ladder diagram.

Example

Instruction table:

Network 000

LD M000 //add normal open coil M000

INV //invert the operation result

OUT Y001 //turn ON when M000 is OFF, turn OFF when M000 is ON

Ladder diagram:

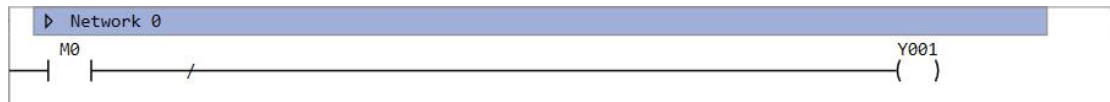


Figure 1.10. 1

OUT

Instruction introduction

1. This instruction outputs the operation result up to this instruction to the specified register.
2. This instruction shows as —()— in the ladder diagram.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(s)	Bit register used as output contact	Y/M/S	-	bool

Table 1.11. 1

Example

Instruction table:

Network 000

LD X000 //add normal open coil X000

OUT Y000 //turn ON when X000 is ON, turn OFF when X000 is OFF

Ladder diagram:



Figure 1.11. 1

Attention

Do not add two or more **OUT** instructions of same bit-register in the ladder diagram, otherwise the output may be prohibited to output. [Figure 1.11.2](#) shows the incorrect example.

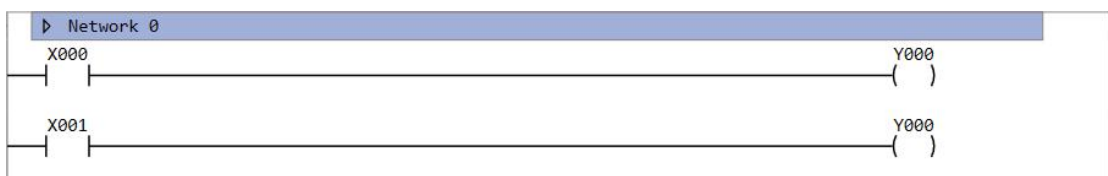


Figure 1.11. 2

OUTIM

Instruction introduction

1. This instruction immediately outputs the operation result up to this instruction to the specified register, regardless of scan period of PLC.
2. This instruction shows as $\text{---}(\text{I})\text{---}$ in the ladder diagram.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(s)	Bit register used as immediate output contact	Y/M/S	-	bool

Table 1.12. 1

Example

Instruction table:

Network 000

LD X000 //add normal open coil X000

OUTIM Y000 //turn ON immediately when X000 is ON, turn OFF immediately when X000 is OFF

Ladder diagram:



Figure 1.12. 1

SET

Instruction introduction

1. This instruction sets (to 1) bits of specified bit-register and the following ones.
2. This instruction shows as $-(s)-$ in the ladder diagram.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(s)	Head of bit-registers to set	Y/M/S	-	bool
(L)	Numbers of set operation	K/H	depends on case	32-bit unsigned integer

Table 1.13. 1

Example

Instruction table:

Network 000

```
LD      X000 //add normal open coil X000
SET     Y000      K16 //set bits from Y000 to Y015
```

Ladder diagram:

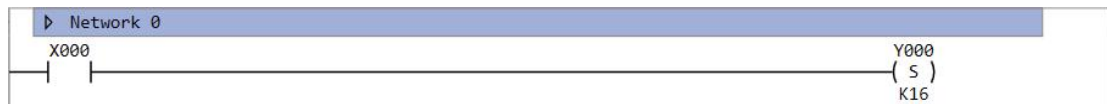


Figure 1.13 1

SETIM

Instruction introduction

1. This instruction immediately set (to 1) bits of specified bit-register and the following ones, regardless of scan period of PLC.
2. This instruction shows as $\text{---(s)}\text{---}$ in the ladder diagram.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(s)	Head of bit-registers to immediate set	Y/M/S	-	bool
(L)	Numbers of set operation	K/H	depends on case	32-bit unsigned integer

Table 1.14. 1

Example

Instruction table:

Network 000

```
LD      X000 //add normal open coil X000
SETIM   Y000      K16 //set bits from Y000 to Y015 immediately
```

Ladder diagram:



Figure 1.14. 1

RST

Instruction introduction

1. This instruction resets (to 0) bits of specified bit-register and the following ones.
2. This instruction shows as ---(R)--- in the ladder diagram.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(S)	Head of bit-registers to reset	Y/M/S	-	bool
(L)	Numbers of reset operation	K/H	depends on case	32-bit unsigned integer

Table 1.15. 1

Example

Instruction table:

Network 000

```
LD      X000 //add normal open coil X000
RST     Y000      K16 //reset bits from Y000 to Y015
```

Ladder diagram:

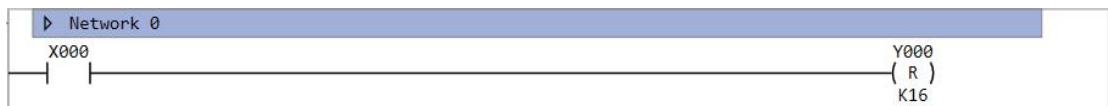


Figure 1.15. 1

RSTIM

Instruction introduction

1. This instruction immediately reset (to 0) bits of specified bit-register and the following ones, regardless of scan period of PLC.
2. This instruction shows as ---(RI)--- in the ladder diagram.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(s)	Head of bit-registers to immediate reset	Y/M/S	-	bool
(L)	Numbers of reset operation	K/H	depends on case	32-bit unsigned integer

Table 1.16. 1

Example

Instruction table:

Network 000

LD X000 //add normal open coil X000

RSTIM Y000 K16 //reset bits from Y000 to Y015 immediately

Ladder diagram:



Figure 1.16. 1

ALT

Instruction introduction

When this instruction is enabled, the specified bit-register will reverse (ON <-> OFF) at each scan period of PLC. When this instruction is disabled, the reversing will stop.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enable or disable instruction		-	bool
(OUT)	Bit register to reserve	Y/M/S	-	bool

Table 1.17. 1

Example

Instruction table:

Network 000

LD X000 //add normal open coil X000

ALT Y000 //Y0 will reverse at each period time of PLC when X000 is ON

Ladder diagram:

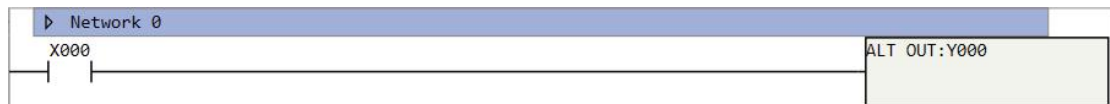


Figure 1.17. 1

ALTP

Instruction introduction

The specified bit-register will reverse (ON <-> OFF) once at rising edge of the input bit.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enable or disable instruction		-	bool
(OUT)	Bit register to reserve	Y/M/S	-	bool

Table 1.18. 1

Example

Instruction table:

Network 000

LD X000 //add normal open coil X000

ALTP Y000 //Y0 will reverse once at rising edge of X000.

Ladder diagram:

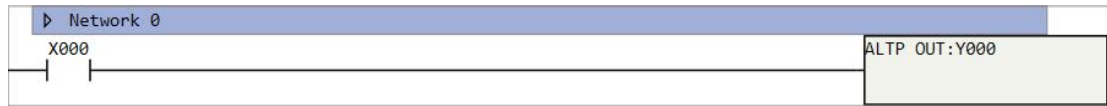


Figure 1.18. 1

Compare Instructions

Notes

1. Instructions in this chapter do compare operations to values of data with different types.
2. The data has 3 types: WORD, DWORD and FLOAT. Details refer to [Table 2.1.1](#).

Data type	Description	Range
WORD	16-bit signed integer	-32768~32767
DWORD	32-bit signed integer	-2147483648~2147483647
FLOAT	float	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$

Table 2.1. 1

3. There are several types of registers or inputs that can be used for comparison, refer to [Table 2.1.2](#).

Register/input type	Description	Numbering system	Range
K	Decimal constant, can be an integer or a decimal.	-	-
H	Hexadecimal code, used to fill the bytes of register.	-	-
D	Inner data-register of PLC.	Decimal	0~8233
TV	Data-register of timer that counts the time pulse.	Decimal	0~255
CV	Data-register of counter that counts the rising-edge.	Decimal	0~255
AI	Data-register of analog input.	Decimal	0~31
AO	Data-register of analog output.	Decimal	0~31
V/Z	Index register that add offset on address. (Usage refer to example of LDW=/LDD=/LDF=)	Decimal	0~7

Table 2.1. 2

LDW=/LDD=/LDF=

Instruction introduction

1. These instructions are EQ (equal to) comparison contact instructions that compare two values of WORD/DWORD/FLOAT type.
2. When the first value is equal to the second, the contact closes, otherwise the contact opens.
3. This instruction shows as $\text{---}^w \text{---} / \text{---}^d \text{---} / \text{---}^f \text{---}$ in the ladder diagram.
4. In instruction table, input contacts can also be connected through **AW=/AD=/AF=** (EQ comparison contact series connection instruction) and **ORW=/ORD=/ORF=** (EQ comparison contact parallel connection instruction) (refer to [example](#)).

Setting data

➤ LDW=

Inputs/outputs	Description	Operand	Range	Data type
(s1)	First value to compare	K/H/D/CV/TV/AI/AO/V/Z	-32768~32767	16-bit signed integer
(s2)	Second value to compare	K/H/D/CV/TV/AI/AO/V/Z	-32768~32767	16-bit signed integer

Table 2.2. 1

➤ LDD=

Inputs/outputs	Description	Operand	Range	Data type
(s1)	First value to compare	K/H/D/CV32	-2147483648~2147483647	32-bit signed integer
(s2)	Second value to compare	K/H/D/CV32	-2147483648~2147483647	32-bit signed integer

Table 2.2. 2

➤ LDF=

Inputs/outputs	Description	Operand	Range	Data type
(s1)	First value to compare	K/D/CV32	$\pm 1.17549\text{e-}38\text{F} \sim \pm 3.40282\text{e}+38\text{F}$	float
(s2)	Second value to compare	K/D/CV32	$\pm 1.17549\text{e-}38\text{F} \sim \pm 3.40282\text{e}+38\text{F}$	float

Table 2.2. 3

Example

Instruction table:

Network 000

```

LDW=   D0      D1 //make EQ comparison between D0 and D1
ORW=   D4      H0123 //make EQ comparison between D4 and H0123, parallel connect the result
AW=    D2      D3 //make EQ comparison between D2 and D3, series connect the result
LDD=   D6      D8 //make EQ comparison between D6(DWORD) and D8(DWORD)
ORD=   D14     K888 //make EQ comparison between D14(DWORD) and K888, parallel connect the result
AD=    D10V0   D12Z0 //make EQ comparison between D10V0 (D0 with address offset of V0) and D12Z0 (D12 with address offset of Z0), series connect the result
    
```

LDF= D18 D20 //make EQ comparison between D18(FLOAT) and D20(FLOAT)
 ORF= D26 K1.234567 //make EQ comparison between D26(FLOAT) and K1.234567, series connect
 the result
 AF= D22 D24 //make EQ comparison between D22(FLOAT) and D24(FLOAT), parallel connect the
 result
 ORB //parallel connect
 ORB //parallel connect
 OUT Y000 //output result to Y000

Ladder diagram:



Figure 2.2. 1

LDW<>/LDD<>/LDF<>

Instruction introduction

1. These instructions are NE (not equal to) comparison contact instructions that compare two values of WORD/DWORD/FLOAT type.
2. When the first value is unequal to the second, the contact closes, otherwise the contact opens.
3. This instruction shows as $\neg|w<>| \neg|d<>| \neg|f<>|$ in the ladder diagram.
4. In instruction table, input contacts can also be connected through **AW<>/AD<>/AF<>** (unequal comparison contact series connection instruction) and **ORW<>/ORD<>/ORF<>** (unequal comparison contact parallel connection instruction) (refer to [example](#)).

Setting data

➤ LDW<>

Inputs/outputs	Description	Operand	Range	Data type
(s1)	First value to compare	K/H/D/CV/TV/AI/AO/V/Z	-32768~32767	16-bit signed integer
(s2)	Second value to compare	K/H/D/CV/TV/AI/AO/V/Z	-32768~32767	16-bit signed integer

Table 2.3. 1

➤ LDD<>

Inputs/outputs	Description	Operand	Range	Data type
(s1)	First value to compare	K/H/D/CV32	-2147483648~2147483647	32-bit signed integer
(s2)	Second value to compare	K/H/D/CV32	-2147483648~2147483647	32-bit signed integer

Table 2.3. 2

➤ LDF<>

Inputs/outputs	Description	Operand	Range	Data type
(s1)	First value to compare	K/D/CV32	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(s2)	Second value to compare	K/D/CV32	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float

Table 2.3. 3

Example

Instruction table:

Network 000

LDW<>	D0	D1 //make NE comparison between D0 and D1
ORW<>	D4	D5 //make NE comparison between D4 and D5, parallel connect the result
AW<>	D2	D3 //make NE comparison between D2 and D3, series connect the result
LDD<>	D6	D8 //make NE comparison between D6(DWORD) and D8(DWORD)
ORD<>	D14	D16 //make NE comparison between D14(DWORD) and D16(DWORD), parallel connect the result
AD<>	D10	D12 //make NE comparison between D10 and D12, series connect the result

LDF<> D18 D20 //make NE comparison between D18(FLOAT) and D20(FLOAT)
 ORF<> D26 D28 //make NE comparison between D26(FLOAT) and D28(FLOAT), series connect
 the result
 AF<> D22 D24 //make NE comparison between D22(FLOAT) and D24(FLOAT), parallel connect the
 result
 ORB //parallel connect
 ORB //parallel connect
 OUT Y000 //output result to Y000

Ladder diagram:



Figure 2.3. 1

LDW>=/LDD>=/LDF>=

Instruction introduction

1. These instructions are GE (greater than or equal to) comparison contact instructions that compare two values of WORD/DWORD/FLOAT type.
2. When the first value is greater than or equal to the second, the contact closes, otherwise the contact opens.
3. This instruction shows as $\text{---|w>=|---|d>=|---|f>=|---}$ in the ladder diagram.
4. In instruction table, input contacts can also be connected through **AW>=/AD>=/AF>=** (GE comparison contact series connection instruction) and **ORW>=/ORD>=/ORF>=** (GE comparison contact parallel connection instruction) (refer to [example](#)).

Setting data

➤ LDW>=

Inputs/outputs	Description	Operand	Range	Data type
(s1)	First value to compare	K/H/D/CV/TV/AI/AO/V/Z	-32768~32767	16-bit signed integer
(s2)	Second value to compare	K/H/D/CV/TV/AI/AO/V/Z	-32768~32767	16-bit signed integer

Table 2.4. 1

➤ LDD>=

Inputs/outputs	Description	Operand	Range	Data type
(s1)	First value to compare	K/H/D/CV32	-2147483648~2147483647	32-bit signed integer
(s2)	Second value to compare	K/H/D/CV32	-2147483648~2147483647	32-bit signed integer

Table 2.4. 2

➤ LDF>=

Inputs/outputs	Description	Operand	Range	Data type
(s1)	First value to compare	K/D/CV32	±1.17549e-38F~±3.40282e+38F	float
(s2)	Second value to compare	K/D/CV32	±1.17549e-38F~±3.40282e+38F	float

Table 2.4. 3

Example

Instruction table:

Network 000

```

LDW>=  D0      D1 //make GE comparison between D0 and D1
ORW>=  D4      D5 //make GE comparison between D4 and D5, parallel connect the result
AW>=   D2      D3 //make GE comparison between D2 and D3, series connect the result
LDD>=  D6      D8 //make GE comparison between D6(DWORD) and D8(DWORD)
ORD>=  D14     D16 //make GE comparison between D14(DWORD) and D16(DWORD), parallel connect
                    the result
    
```

AD>= D10 D12 //make GE comparison between D10 and D12, series connect the result
 LDF>= D18 D20 //make GE comparison between D18(FLOAT) and D20(FLOAT)
 ORF>= D26 D28 //make GE comparison between D26(FLOAT) and D28(FLOAT), series connect
 the result
 AF>= D22 D24 //make GE comparison between D22(FLOAT) and D24(FLOAT), parallel connect the
 result
 ORB //parallel connect
 ORB //parallel connect
 OUT Y000 //output result to Y000

Ladder diagram:



Figure 2.4. 1

LDW<=/LDD<=/LDF<=

Instruction introduction

1. These instructions are LE (less than or equal to) comparison contact instructions that compare two values of WORD/DWORD/FLOAT type.
2. When the first value is less than or equal to the second, the contact closes, otherwise the contact opens.
3. This instruction shows as $\text{---|w<=|---|d<=|---|f<=|---}$ in the ladder diagram.
4. In instruction table, input contacts can also be connected through **AW<=/AD<=/AF<=** (LE comparison contact series connection instruction) and **ORW<=/ORD<=/ORF<=** (LE comparison contact parallel connection instruction) (refer to [example](#)).

Setting data

➤ LDW<=

Inputs/outputs	Description	Operand	Range	Data type
(s1)	First value to compare	K/H/D/CV/TV/AI/AO/V/Z	-32768~32767	16-bit signed integer
(s2)	Second value to compare	K/H/D/CV/TV/AI/AO/V/Z	-32768~32767	16-bit signed integer

Table 2.5. 1

➤ LDD<=

Inputs/outputs	Description	Operand	Range	Data type
(s1)	First value to compare	K/H/D/CV32	-2147483648~2147483647	32-bit signed integer
(s2)	Second value to compare	K/H/D/CV32	-2147483648~2147483647	32-bit signed integer

Table 2.5. 2

➤ LDF<=

Inputs/outputs	Description	Operand	Range	Data type
(s1)	First value to compare	K/D/CV32	$\pm 1.17549\text{e}-38\text{F} \sim \pm 3.40282\text{e}+38\text{F}$	float
(s2)	Second value to compare	K/D/CV32	$\pm 1.17549\text{e}-38\text{F} \sim \pm 3.40282\text{e}+38\text{F}$	float

Table 2.5. 3

Example

Instruction table:

Network 000

```

LDW<=  D0      D1 //make LE comparison between D0 and D1
ORW<=  D4      D5 //make LE comparison between D4 and D5, parallel connect the result
AW<=   D2      D3 //make LE comparison between D2 and D3, series connect the result
LDD<=  D6      D8 //make LE comparison between D6(DWORD) and D8(DWORD)
ORD<=  D14     D16 //make LE comparison between D14(DWORD) and D16(DWORD), parallel connect
           the result
AD<=   D10     D12 //make LE comparison between D10 and D12, series connect the result
LDF<=  D18     D20 //make LE comparison between D18(FLOAT) and D20(FLOAT)
    
```

ORF<= D26 D28 //make LE comparison between D26(FLOAT) and D28(FLOAT), series connect
 the result
 AF<= D22 D24 //make LE comparison between D22(FLOAT) and D24(FLOAT), parallel connect the
 result
 ORB //parallel connect
 ORB //parallel connect
 OUT Y000 //output result to Y000

Ladder diagram:



Figure 2.5. 1

LDW>/LDD>/LDF>

Instruction introduction

1. These instructions are GT (greater than) comparison contact instructions that compare two values of WORD/DWORD/FLOAT type.
2. When the first value is greater than the second, the contact closes, otherwise the contact opens.
3. This instruction shows as $\text{---}^w \text{---} / \text{---}^D \text{---} / \text{---}^F \text{---}$ in the ladder diagram.
4. In instruction table, input contacts can also be connected through **AW>/AD>/AF>** (GT comparison contact series connection instruction) and **ORW>/ORD>/ORF>** (GT comparison contact parallel connection instruction) (refer to [example](#)).

Setting data

➤ LDW>

Inputs/outputs	Description	Operand	Range	Data type
(s1)	First value to compare	K/H/D/CV/TV/AI/AO/V/Z	-32768~32767	16-bit signed integer
(s2)	Second value to compare	K/H/D/CV/TV/AI/AO/V/Z	-32768~32767	16-bit signed integer

Table 2.6. 1

➤ LDD>

Inputs/outputs	Description	Operand	Range	Data type
(s1)	First value to compare	K/H/D/CV32	-2147483648~2147483647	32-bit signed integer
(s2)	Second value to compare	K/H/D/CV32	-2147483648~2147483647	32-bit signed integer

Table 2.6. 2

➤ LDF>

Inputs/outputs	Description	Operand	Range	Data type
(s1)	First value to compare	K/D/CV32	$\pm 1.17549\text{e}-38\text{F} \sim \pm 3.40282\text{e}+38\text{F}$	float
(s2)	Second value to compare	K/D/CV32	$\pm 1.17549\text{e}-38\text{F} \sim \pm 3.40282\text{e}+38\text{F}$	float

Table 2.6. 3

Example

Instruction table:

Network 000

```

LDW>   D0      D1 //make GT comparison between D0 and D1
ORW>   D4      D5 //make GT comparison between D4 and D5, parallel connect the result
AW>    D2      D3 //make GT comparison between D2 and D3, series connect the result
LDD>   D6      D8 //make GT comparison between D6(DWORD) and D8(DWORD)
ORD>   D14     D16 //make GT comparison between D14(DWORD) and D16(DWORD), parallel connect
           the result
    
```

```

AD>      D10      D12 //make GT comparison between D10 and D12, series connect the result
LDF>      D18      D20 //make GT comparison between D18(FLOAT) and D20(FLOAT)
ORF>      D26      D28 //make GT comparison between D26(FLOAT) and D28(FLOAT), series connect
           the result
AF>      D22      D24 //make GT comparison between D22(FLOAT) and D24(FLOAT), parallel connect the
           result

ORB //parallel connect
ORB //parallel connect
OUT      Y000 //output result to Y000

```

Ladder diagram:



Figure 2.6. 1

LDW</LDD</LDF<

Instruction introduction

1. These instructions are LT (less than) comparison contact instructions that compare two values of WORD/DWORD/FLOAT type.
2. When the first value is less than the second, the contact closes, otherwise the contact opens.
3. This instruction shows as $\text{---}^{\text{w}}\text{---}/\text{---}^{\text{D}}\text{---}/\text{---}^{\text{F}}\text{---}$ in the ladder diagram.
4. In instruction table, input contacts can also be connected through **AW</AD</AF<** (LT comparison contact series connection instruction) and **ORW</ORD</ORF<** (LT comparison contact parallel connection instruction) (refer to [example](#)).

Setting data

➤ LDW<

Inputs/outputs	Description	Operand	Range	Data type
(s1)	First value to compare	K/H/D/CV/TV/AI/AO/V/Z	-32768~32767	16-bit signed integer
(s2)	Second value to compare	K/H/D/CV/TV/AI/AO/V/Z	-32768~32767	16-bit signed integer

Table 2.7. 1

➤ LDD<

Inputs/outputs	Description	Operand	Range	Data type
(s1)	First value to compare	K/H/D/CV32	-2147483648~2147483647	32-bit signed integer
(s2)	Second value to compare	K/H/D/CV32	-2147483648~2147483647	32-bit signed integer

Table 2.7. 2

➤ LDF<

Inputs/outputs	Description	Operand	Range	Data type
(s1)	First value to compare	K/D/CV32	$\pm 1.17549\text{e}-38\text{F} \sim \pm 3.40282\text{e}+38\text{F}$	float
(s2)	Second value to compare	K/D/CV32	$\pm 1.17549\text{e}-38\text{F} \sim \pm 3.40282\text{e}+38\text{F}$	float

Table 2.7. 3

Example

Instruction table:

Network 000

```
LDW<   D0      D1 //make LT comparison between D0 and D1
ORW<   D4      D5 //make LT comparison between D4 and D5, parallel connect the result
AW<    D2      D3 //make LT comparison between D2 and D3, series connect the result
LDD<   D6      D8 //make LT comparison between D6(DWORD) and D8(DWORD)
ORD<   D14     D16 //make LT comparison between D14(DWORD) and D16(DWORD), parallel connect
           the result
```

AD< D10 D12 //make LT comparison between D10 and D12, series connect the result
 LDF< D18 D20 //make LT comparison between D18(FLOAT) and D20(FLOAT)
 ORF< D26 D28 //make LT comparison between D26(FLOAT) and D28(FLOAT), series connect the result
 the result
 AF< D22 D24 //make LT comparison between D22(FLOAT) and D24(FLOAT), parallel connect the result
 ORB //parallel connect
 ORB //parallel connect
 OUT Y000 //output result to Y000

Ladder diagram:



Figure 2.7. 1

CMP/CMPD/CMPF

Instruction introduction

1. These instructions output comparison result of two input values to 3 continuous bit-registers. **CMP** is for WORD type value, **CMPD** is for DWORD type value, **CMPF** is for FLOAT type value.
2. When the first input value is greater than the second, the first of output bit-registers is set to 1; When the first input value is equal to the second, the second of output bit-registers is set to 1; When the first input value is less than the second, the third of output bit-registers is set to 1.

Setting data

➤ **CMP**

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN1)	First value to compare	K/H/D/CV/TV/AI/AO/V/Z	-32768~32767	16-bit signed integer
(IN2)	Second value to compare	K/H/D/CV/TV/AI/AO/V/Z	-32768~32767	16-bit signed integer
(OUT)	First of the output bit-registers	Y/M/S	-	bool

Table 2.8. 1

➤ **CMPD**

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN1)	First value to compare	K/H/D	-2147483648~2147483647	32-bit signed integer
(IN2)	Second value to compare	K/H/D	-2147483648~2147483647	32-bit signed integer
(OUT)	First of the output bit-registers	Y/M/S	-	bool

Table 2.8. 2

➤ **CMPF**

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool

(IN1)	First value to compare	K/D	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(IN2)	Second value to compare	K/D	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(OUT)	First of the output bit-registers	Y/M/S	-	bool

Table 2.8. 3

Example

Instruction table:

Network 000

```

LD      M0
CMP     D0      D1      M100 //compare D0 and D1
POP
LD      M100 //if D0>D1, set M100 to ON
OUT     M1000
POP
LD      M101 //if D0=D1, set M101 to ON
OUT     M1001
POP
LD      M102 //if D0<D1, set M102 to ON
OUT     M1002
POP

```

Ladder diagram:

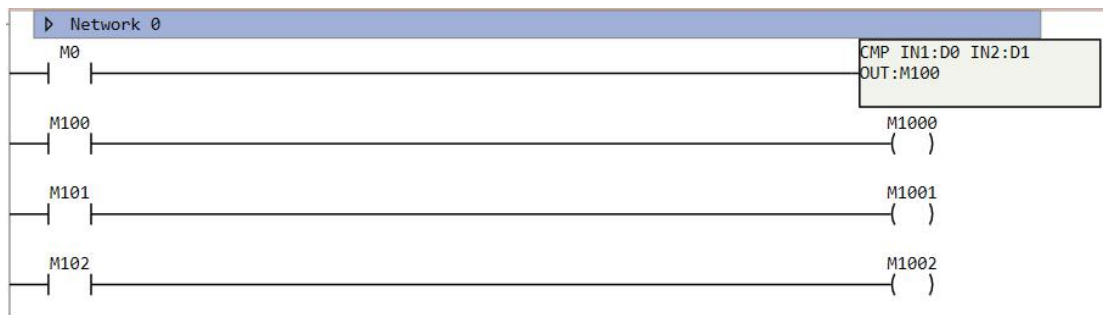


Figure 2.8. 1

ZCP/ZCPD/ZCPF

Instruction introduction

1. These instructions calculate where comparison value located by comparing it with an interval consists of 2 limit values, and output the result to 3 continuous bit-registers. **ZCP** is for WORD type value, **ZCPD** is for DWORD type value, **ZCPF** is for FLOAT type value.
2. When comparison value is less than the lower-limit value, first of output bit-registers is set to 1; When comparison value is between the lower-limit value and upper-limit value (no greater than lower-limit value and no less than upper-limit value), second of output bit-registers is set to 1; When comparison value is greater than the upper-limit value, third of output bit-registers is set to 1.

Setting data

➤ ZCP

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN1)	lower-limit value of interval	K/H/D/CV/TV/AI/AO/V/Z	-32768~32767	16-bit signed integer
(IN2)	upper-limit value of interval	K/H/D/CV/TV/AI/AO/V/Z	-32768~32767	16-bit signed integer
(IN3)	comparison value to be compared with the interval	K/H/D/CV/TV/AI/AO/V/Z	-32768~32767	16-bit signed integer
(OUT)	First of the output bit-registers	Y/M/S	-	bool

Table 2.9. 1

➤ CMPD

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN1)	lower-limit value of interval	K/H/D	-2147483648~2147483647	32-bit signed integer
(IN2)	upper-limit value of interval	K/H/D	-2147483648~2147483647	32-bit signed integer
(IN3)	comparison value to be compared with the interval	K/H/D	-2147483648~2147483647	32-bit signed integer
(OUT)	First of the output bit-registers	Y/M/S	-	bool

Table 2.9. 2

➤ CMPF

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables		0/1	bool

	instruction.			
(IN1)	lower-limit value of interval	K/D	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(IN2)	upper-limit value of interval	K/D	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(IN3)	comparison value to be compared with the interval	K/D	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(OUT)	First of the output bit-registers	Y/M/S	-	bool

Table 2.9. 3

Example

Instruction table:

Network 000

```

LD      M0
ZCP     K100      K200      D0      Y000 //When D0<K100, Y000 is ON; When K100<=D0<=K200,
Y001 is ON; When D0>K200, Y002 is ON

```

Ladder diagram:

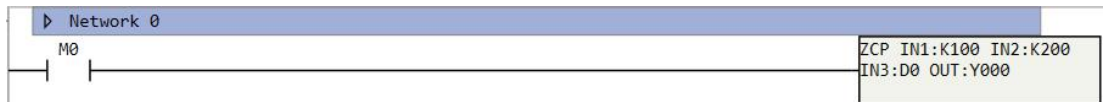


Figure 2.9. 1

Conversion Instructions

Notes

1. BIN code is binary code, [Figure 3.1.1](#) shows how it is encoded.

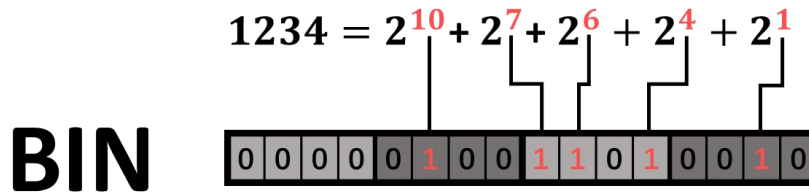


Figure 3.1. 1

2. BCD code is binary-coded decimal code, it converts each decimal digit of the number into 4-bit binary code, [Figure 3.1.2](#) shows how it is encoded.

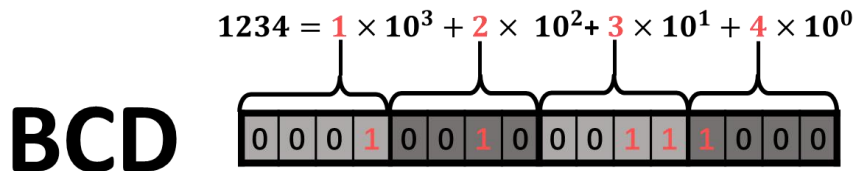


Figure 3.1. 2

3. Conversion error may occur, and there are system special function registers that record these errors:

M8168 record if operation makes overflow, when DWORD data is too big to be converted to WORD, this bit-register will be set to ON.

M8169 record if BIN to BCD conversion is valid, if the BCD code is over 0x9999, this bit-register will be set to ON.

WTOD

Instruction introduction

This instruction converts WORD data to DWORD data, and stores converted data in specified D-registers.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	WORD data to be converted.	K/H/D/CV/TV/AI/AO/V/Z	-32768~32767	16-bit signed integer
(OUT)	Output D-register to store DWORD converted data	D	-	32-bit signed integer

Table 3.2. 1

Example

Instruction table:

Network 000

LD X000

WTOD D0 D1 //Convert WORD data in D0 to DWORD and store result in D1D2

Ladder diagram:



Figure 3.2. 1

DTOW

Instruction introduction

1. This instruction converts DWORD data to WORD data, and stores converted data in specified D-registers.
2. If the DWORD data is too big to convert, the overflow flag bit-register **M8169** will be set to 1, and the output D-registers won't show the data correctly.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	DWORD data to be converted.	K/H/D	-2147483648~2147483647	32-bit signed integer
(OUT)	D-register to store converted WORD data	D	-	32-bit signed integer

Table 3.3. 1

Example

Instruction table:

Network 000

LD X000

WTD D0 D2 //Convert DWORD data in D0D1 to WORD and store result in D2

Ladder diagram:

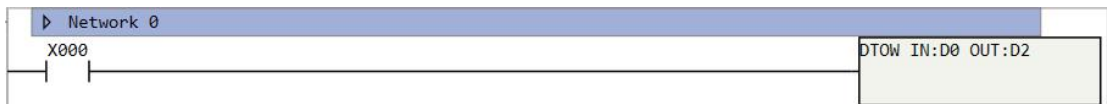


Figure 3.3. 1

DTOF

Instruction introduction

This instruction converts DWORD data to FLOAT data, and stores converted data in specified D-registers.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	DWORD data to be converted.	K/H/D	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(OUT)	D-register to store converted FLOAT data	D	-	float

Table 3.4. 1

Example

Instruction table:

Network 000

LD X000

DTOF D0 D2 //Convert DWORD data in D0D1 to FLOAT and store result in D2D3

Ladder diagram:

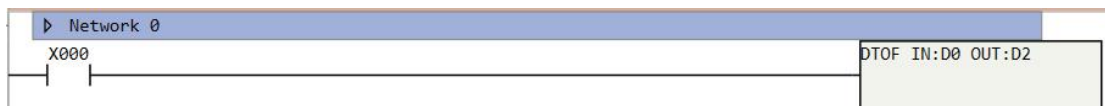


Figure 3.4. 1

BIN/BIND

Instruction introduction

1. These instructions convert BCD data to BIN data, and stores converted data in specified D-registers (details of BCD and BIN refer to [notes](#)). **BIN** instruction is for 16-bit data, **BIND** instruction is for 32-bit data.
2. When the conversion is invalid, system special function bit-register **M8168** will be set to ON.

Setting data

➤ BIN

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	16-bit BCD data to be converted.	K/H/D/CV/TV/AI/AO/V/Z	0x0000~0x9999	16-bit BCD code
(OUT)	D-register to store converted BIN data	D/CV/TV/AO/V/Z	-	16-bit BIN code

Table 3.5. 1

➤ BIND

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	32-bit BCD data to be converted.	K/H/D/C V	0x0000000~0x9999999	32-bit BCD code
(OUT)	D-register to store converted BIN data	D/CV	-	32-bit BIN code

Table 3.5. 2

Example

Instruction table:

Network 000

```
LD      X000
BIN     H0234      D0 //Convert BCD code 234 to BIN code and store result in D0
```

Ladder diagram:



Figure 3.5. 1

BCD/BCDD

Instruction introduction

1. These instructions convert BIN data to BCD data, and stores converted data in specified D-registers (details of BCD and BIN refer to [notes](#)). **BCD** instruction is for 16-bit data, **BCDD** instruction is for 32-bit data.
2. When the conversion is invalid, system special function bit-register **M8168** will be set to ON.

Setting data

➤ BCD

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	16-bit BIN data to be converted.	K/H/D/CV/TV/AI/AO/V/Z	0x0000~0xFFFF	16-bit BIN code
(OUT)	D-register to store converted BCD data	D/CV/TV/AO/V/Z	-	16-bit BCD code

Table 3.6. 1

➤ BCDD

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	32-bit BIN data to be converted.	K/H/D/CV	0x00000000~0xFFFFFFFF	32-bit BCD code
(OUT)	D-register to store converted BCD data	D/CV	-	32-bit BCD code

Table 3.6. 2

Example

Instruction table:

Network 000

LD X000

BCD D0 D1 //Convert BIN code in D0 to BCD code and store result in D1

Ladder diagram:

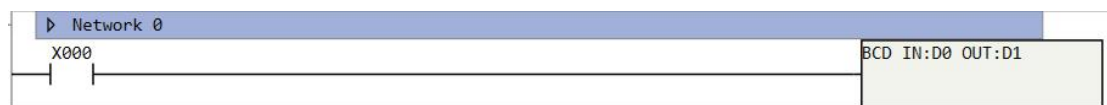


Figure 3.6. 1

ROUND

Instruction introduction

This instruction rounds FLOAT data to DWORD data, and stores converted data in specified D-registers. If the decimal part of the FLOAT date is less than 0.5, FLOAT data is rounded off; otherwise it is rounded up.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	FLOAT data to be rounded.	K/D	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(OUT)	D-register to store rounded DWORD data.	D	-	32-bit signed integer

Table 3.7. 1

Example

Instruction table:

Network 000

LD X000

ROUND K4.56 D0 //round up K4.56 to K5 and store result in D0D1

Ladder diagram:

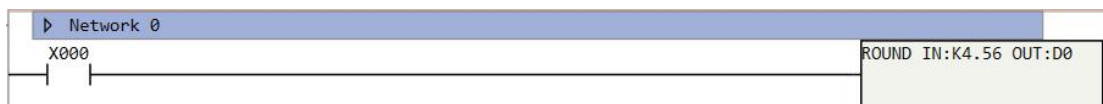


Figure 3.7. 1

TRUNC

Instruction introduction

This instruction truncates decimal part of FLOAT data, convert it to DWORD data, and stores converted data in specified D-registers.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	FLOAT data to be truncated.	K/D	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(OUT)	D-register to store converted DWORD data.	D	-	32-bit signed integer

Table 3.8. 1

Example

Instruction table:

Network 000

LD X000

TRUNC K4.56 D0 //truncate K4.56 to K4 and store result in D0D1

Ladder diagram:



Figure 3.8. 1

Logic Operation Instructions

Notes

Instructions in this chapter do bitwise logic operations on WORD (16-bit) or DWORD (32-bit) data. The logic operations include invert operation (**INV**), and operation (**AND**), or operation (**OR**), exclusive-or operation (**XOR**), and complementary operation (**NEG**). **INV** and **NEG** operate on a single code, details refer to [Table 4.1.1](#); **AND**, **OR**, and **XOR** operate on two codes, details refer to [Table 4.1.2](#).

Operator	Description	Example	
		Input	Output
INV	Invert each bit of the code.	0xFFFF	0x0000
NEG	Invert each bit of the code and add the result with 1 to get complement which is negative form of the source code.	0xFFFF	0x0001

Table 4.1. 1

Operator	Description	Example		
		Input1	Input2	Output
AND	Do add operation on each corresponding bit of two codes and output result. (If at least one of two bits is 0, operation result is 0, otherwise result is 1)	0	0	0
		0	1	0
		1	0	0
		1	1	1
OR	Do or operation on each corresponding bit of two codes and output result. (If at least one of two bits is 1, operation result is 1, otherwise result is 0.)	0	0	0
		0	1	1
		1	0	1
XOR	Do exclusive-or operation on each corresponding bit of two codes and output result. (If two bits are different, operation result is 1, otherwise result is 0)	1	1	1
		0	0	0
		0	1	1
		1	0	1
		1	1	0

Table 4.1. 2

INVW/INVD

Instruction introduction

These instructions do invert operation on input code, and store the result in specified register. **INVW** is for WORD data, **INVD** is for DWORD data. Details of invert operation refer to [notes](#) of this chapter.

Setting data

➤ INVW

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	Code to do invert operation.	K/H/D/CV/TV/AI/AO/V/Z	0x0000~0xFFFF	16-bit code
(OUT)	D-register to store output code	D/CV/TV/AO/V/Z	-	16-bit code

Table 4.2. 1

➤ INVD

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	Code to do invert operation.	K/H/D	0x00000000~0xFFFFFFFF	32-bit code
(OUT)	D-register to store output code	D	-	32-bit code

Table 4.2. 2

Example

Instruction table:

Network 000

LD X000

INVW D0 D1 //do invert operation on code in D0 and store result in D1

Ladder diagram:



Figure 4.2. 1

ANDW/ANDD

Instruction introduction

These instructions do and operation on input codes, and store the result in specified register. **ANDW** is for WORD data, **ANDD** is for DWORD data. Details of and operation refer to [notes](#) of this chapter.

Setting data

➤ ANDW

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN1)	First code to do and operation.	K/H/D/CV/TV/AI/AO/V/Z	0x0000~0xFFFF	16-bit code
(IN2)	Second code to do and operation	K/H/D/CV/TV/AI/AO/V/Z	0x0000~0xFFFF	16-bit code
(OUT)	D-register to store output code	D/CV/TV/AO/V/Z	-	16-bit code

Table 4.3. 1

➤ ANDD

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN1)	First code to do and operation.	K/H/D	0x00000000~0xFFFFFFFF	32-bit code
(IN2)	Second code to do and operation	K/H/D	0x00000000~0xFFFFFFFF	32-bit code
(OUT)	D-register to store output code	D	-	32-bit code

Table 4.3. 2

Example

Instruction table:

Network 000

LD X000

ANDW D0 D1 D2 //do and operation on codes in D0 and D2, and store result in D1

Ladder diagram:

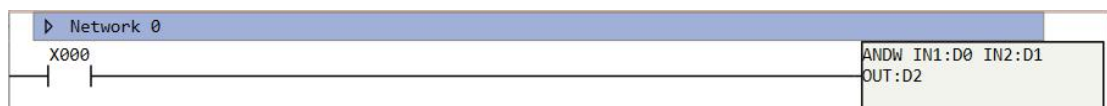


Figure 4.3. 1

ORW/ORD

Instruction introduction

These instructions do or operation on input codes, and store the result in specified register. **ORW** is for WORD data, **ORD** is for DWORD data. Details of or operation refer to [notes](#) of this chapter.

Setting data

➤ ORW

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN1)	First code to do or operation.	K/H/D/CV/TV/AI/AO/V/Z	0x0000~0xFFFF	16-bit code
(IN2)	Second code to do or operation	K/H/D/CV/TV/AI/AO/V/Z	0x0000~0xFFFF	16-bit code
(OUT)	D-register to store output code	D/CV/TV/AO/V/Z	-	16-bit code

Table 4.4. 1

➤ ORD

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN1)	First code to do or operation.	K/H/D	0x00000000~0xFFFFFFFF	32-bit code
(IN2)	Second code to do or operation	K/H/D	0x00000000~0xFFFFFFFF	32-bit code
(OUT)	D-register to store output code	D	-	32-bit code

Table 4.4. 2

Example

Instruction table:

Network 000

LD X000

ANDW D0 D1 D2 //do and operation on codes in D0 and D1, and store result in D2

Ladder diagram:



Figure 4.4. 1

XORW/XORD

Instruction introduction

These instructions do exclusive-or operation on input codes, and store the result in specified register. **XORW** is for WORD data, **XORD** is for DWORD data. Details of exclusive-or operation refer to [notes](#) of this chapter.

Setting data

➤ XORW

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN1)	First code to do exclusive-or operation.	K/H/D/CV/TV/AI/AO/V/Z	0x0000~0xFFFF	16-bit code
(IN2)	Second code to do exclusive-or operation	K/H/D/CV/TV/AI/AO/V/Z	0x0000~0xFFFF	16-bit code
(OUT)	D-register to store output code	D/CV/TV/AO/V/Z	-	16-bit code

Table 4.5. 1

➤ XORD

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN1)	First code to do exclusive-or operation.	K/H/D	0x00000000~0xFFFFFFFF	32-bit code
(IN2)	Second code to do exclusive-or operation	K/H/D	0x00000000~0xFFFFFFFF	32-bit code
(OUT)	D-register to store output code	D	-	32-bit code

Table 4.5. 2

Example

Instruction table:

Network 000

LD X000

XORW D0 D1 D2 //do and operation on codes in D0 and D1, and store result in D2

Ladder diagram:

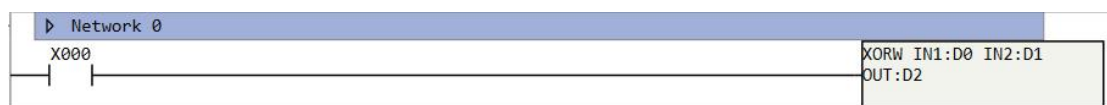


Figure 4.5. 1

NEGW/NEGD

Instruction introduction

These instructions do complementary operation on input codes, and store the result in specified register. **NEGW** is for WORD data, **NEGD** is for DWORD data. Details of complementary operation refer to [notes](#) of this chapter.

Setting data

➤ NEGW

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN1)	First code to do complementary operation.	K/H/D/CV/TV/AI/AO/V/Z	0x0000~0xFFFF	16-bit code
(IN2)	Second code to do complementary operation	K/H/D/CV/TV/AI/AO/V/Z	0x0000~0xFFFF	16-bit code
(OUT)	D-register to store output code	D/CV/TV/AO/V/Z	-	16-bit code

Table 4.6. 1

➤ NEGD

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN1)	First code to do complementary operation.	K/H/D	0x00000000~0xFFFFFFFF	32-bit code
(IN2)	Second code to do complementary operation	K/H/D	0x00000000~0xFFFFFFFF	32-bit code
(OUT)	D-register to store output code	D	-	32-bit code

Table 4.6. 2

Example

Instruction table:

Network 000

LD X000

NEGW D0 D1 //do complementary operation on code in D0 and store result in D1

Ladder diagram:



Figure 4.6. 1

Transfer Instructions

Notes

Instructions in this chapter transfer data with certain length to specified registers. The data length has different, details refer to [Table 5.1.1](#).

data length type	Description	Corresponding instructions
bit	The minimum unit of data	-
nibble	4 bits	SMOV
byte	8 bits	-
word	16 bits, length of WORD data	MOV, MVBLK, FMOV, XCH
double-word	32 bits, length of DWORD and FLOAT data.	MOVD/MOVF, MVDBLK, FMOVD, XCHD/XCHF

Table 5.1. 1

MOV/MOVD/MOVF

Instruction introduction

These instructions copy data from source registers and store it in destination registers. **MOV** is for WORD data, **MOVD** is for DWORD data, **MOVF** is for FLOAT data.

Setting data

➤ MOV

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(S)	Source register.	D/CV/TV/AI/AO/V/Z	-	16-bit signed integer
(D)	Destination register.	D/CV/TV/AO/V/Z	-	36-bit signed integer

Table 5.2. 1

➤ MOVD

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(S)	Source register.	K/H/D/CV	-	32-bit signed integer
(D)	Destination register.	D/CV	-	32-bit signed integer

Table 5.2. 2

➤ MOVF

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(S)	Source register.	K/D/AI/AO	-	float
(D)	Destination register.	D	-	float

Table 5.2. 3

Example

Instruction table:

Network 000

LD X000

MOV D0 D1 //copy WORD data from D0 and store in D1

Ladder diagram:

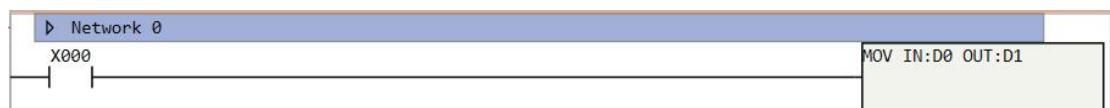


Figure 5.2. 1

MVBLK/MVDBLK

Instruction introduction

These instructions copy word/double-word data of specified length from source registers and store it in destination registers. **MVBLK** is for word, **MVDBLK** is for double-word.

Setting data

➤ MVBLK

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(S)	Source registers.	D/CV/TV/AI/AO/V/Z	-	16-bit code
(D)	Destination registers.	D/CV/TV/AO/V/Z	-	16-bit code
(N)	Numbers of word data to transfer	K/H/D	1~1024	16-bit unsigned integer

Table 5.3. 1

➤ MVDBLK

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(S)	Source registers.	D/CV/TV/AI/AO/V/Z	-	32-bit code
(D)	Destination registers.	D/CV/TV/AO/V/Z	-	32-bit code
(N)	Numbers of double-word data to transfer	K/H/D	1~1024	16-bit unsigned integer

Table 5.3. 2

Example

Instruction table:

Network 000

LD X000

MVBLK D0 D5 K10 //copy data of D0-D9 and store in D5-D14

Ladder diagram:

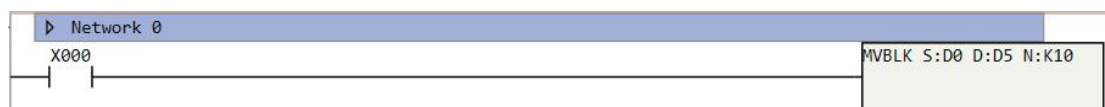


Figure 5.3. 1

FMOV/FMOVD

Instruction introduction

These instructions copy word/double-word data from source registers and store it in multiple continuous destination registers. **FMOV** is for word, **FMOVD** is for double-word.

Setting data

➤ MVBLK

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(S)	Source register.	D/CV/TV/AI/AO/V/Z	-	16-bit code
(D)	Destination registers.	D/CV/TV/AO/V/Z	-	16-bit code
(N)	Numbers of destination registers to transfer.	K/H/D	1~1024	16-bit unsigned integer

Table 5.4. 1

➤ MVDBLK

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(S)	Source register.	K/H/D	-	32-bit code
(D)	Destination register.	D	-	32-bit code
(N)	Numbers of double-word data to transfer.	K/H/D	1~8233	16-bit unsigned integer

Table 5.4. 2

Example

Instruction table:

Network 000

LD X000

FMOV K100 D5 K5 //copy K100 and store in D5-D9

Ladder diagram:

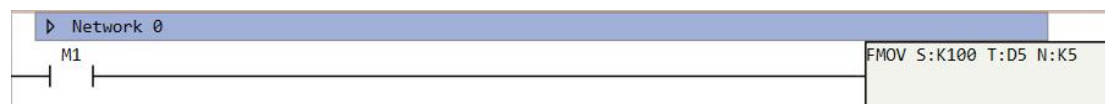


Figure 5.4. 1

SMOV

Instruction introduction

1. This instruction is digit move instruction, it distributes and composes data in units of nibble (4 bits).
2. This instruction converts binary code in source register (S) and destination register (D) to BCD code (0x0000 to 0x9999), (N2) length nibbles starting from the (N1)th nibble are transferred to the destination register (D) starting from the (N3)th nibble, converted into binary, and then stored to the destination register(D).

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(S)	Source register.	K/H/D/CV/TV/AI/AO/V/Z	-	16-bit BIN code
(N1)	Head nibble position to be moved	K/H	1~4	16-bit unsigned integer
(N2)	Number of nibbles to be moved	K/H	1~4	16-bit unsigned integer
(D)	Destination registers.	D/CV/TV/AO/V/Z	-	16-bit BIN code
(N3)	Head digit position of movement destination	K/H	1~4	16-bit unsigned integer

Table 5.5. 1

Example

Instruction table:

Network 000

```

LDP    M1
MOV    K4321    D1 //write 4321 in D1
MOV    K8888    D2 //write 8888 in D2
POP
LD     M1
SMOV   D1      K4      K2      D2      K3 //When M1 is ON, D2 is stored with 8438
  
```

Ladder diagram:

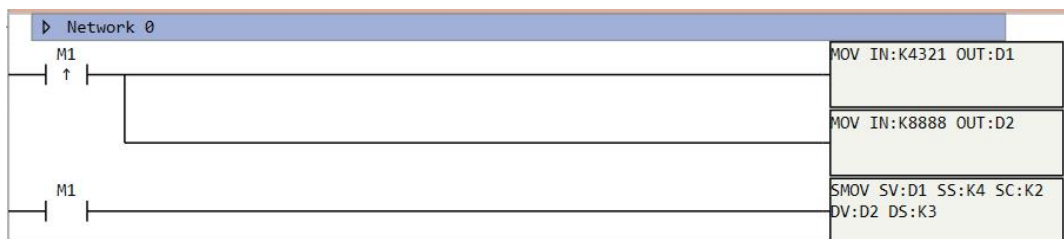


Figure 5.5. 1

XCH/XCHD/XCHF

Instruction introduction

These instructions exchange data of two registers. **XCH** is for WORD data, **XCHD** is for DWORD data, **XCHF** is for FLOAT data.

Setting data

➤ XCH

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(D1)	Register to be exchanged.	D/CV/TV/AO/V/Z	-	16-bit signed integer
(D2)	Register to be exchanged.	D/CV/TV/AO/V/Z	-	16-bit signed integer

Table 5.6. 1

➤ XCHD

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(D1)	Register to be exchanged.	D	-	32-bit signed integer
(D2)	Register to be exchanged.	D	-	32-bit signed integer

Table 5.6. 2

➤ XCHF

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(D1)	Register to be exchanged.	D	-	float
(D2)	Register to be exchanged.	D	-	float

Table 5.6. 3

Example

Instruction table:

Network 000

LDP M7

XCH D5 D6 //Exchange data of D5 and D6

Ladder diagram:

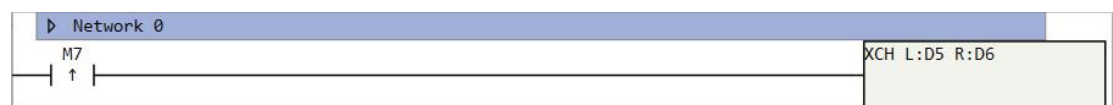


Figure 5.6. 1

Floating-point Calculation Instructions

Notes

1. Instructions in this chapter do floating-point calculations to float inputs. The floating-point calculations include: addition, subtraction, multiplication, division, radication, sine, cosine, tangent, logarithm, exponent, power, absolute value, arcsine, arccosine, arctangent, angle-to-radian conversion, and radian-to-angle conversion.
2. There are system special function bit-registers that record calculation errors.
M8169: When overflow or underflow occurs in calculation, this bit is ON.
M8170: When calculation result is minus or calculation input is illegal minus, this bit is ON.
M8171: When calculation result is 0, this bit is ON.
M8172: When divisor of division is 0, this bit is ON.

ADDF

Instruction introduction

1. This instruction adds one float type input (augend) with another (addend) and stores sum in specified register.
2. This instruction may set [M8170](#) and [M8171](#) to ON.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN1)	Augend of addition.	K/D	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(IN2)	Addend of addition.	K/D	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(OUT)	Output register stores the sum	D	-	float

Table 6.2. 1

Example

Instruction table:

Network 000

```
LD      X000
ADDF   D0      D2      D4 //D0D1+D2D3=D4D5
```

Ladder diagram:

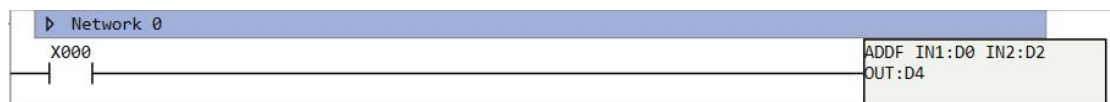


Figure 6.2. 1

SUBF

Instruction introduction

1. This instruction subtracts one float type input (minuend) with another (subtrahend) and stores difference in specified register.
2. This instruction may set [M8170](#) and [M8171](#) to ON.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN1)	Minuend of subtraction.	K/D	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(IN2)	Subtrahend of subtraction.	K/D	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(OUT)	Register stores the difference.	D	-	float

Table 6.3. 1

Example

Instruction table:

Network 000

```

LD      X000
SUBF   D0      D2      D4 //D0D1-D2D3=D4D5

```

Ladder diagram:



Figure 6.3. 1

MULF

Instruction introduction

1. This instruction multiplies one float type input (multiplicand) with another (multiplier) and stores product in specified register.
2. This instruction may set [M8170](#) and [M8171](#) to ON.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN1)	Multiplicand of multiplication.	K/D	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(IN2)	Multiplier of multiplication.	K/D	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(OUT)	Register stores the product.	D	-	float

Table 6.4. 1

Example

Instruction table:

Network 000

```

LD      X000
MULF   D0      D2      D4 //D0D1×D2D3=D4D5

```

Ladder diagram:



Figure 6.4. 1

DIVF

Instruction introduction

1. This instruction divides one float type input (dividend) with another (divisor) and stores quotient in specified register.
2. This instruction may set [M8170](#), [M8171](#) and [M8172](#) to ON.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN1)	Dividend of division.	K/D	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(IN2)	Divisor of division.	K/D	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(OUT)	Register stores the quotient.	D	-	float

Table 6.5. 1

Example

Instruction table:

Network 000

```

LD      X000
DIVF   D0      D2      D4 //D0D1÷D2D3=D4D5
  
```

Ladder diagram:

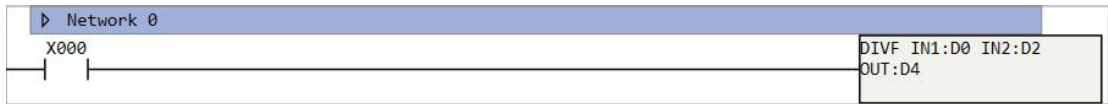


Figure 6.5. 1

SQRT

Instruction introduction

1. This instruction takes square root of the float type input (radicand) and stores root in specified register.
2. This instruction may set [M8170](#) and [M8171](#) to ON.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	Radicand of radication.	K/D	$\pm 1.17549\text{e-}38\text{F} \sim \pm 3.40282\text{e+}38\text{F}$	float
(OUT)	Register stores the square root.	D	-	float

Table 6.6. 1

Example

Instruction table:

Network 000

LD X000

SQRT K256 D0 //SQRT(256) = D0D1 = 16

Ladder diagram:



Figure 6.6. 1

SIN

Instruction introduction

1. This instruction calculates sine of the float type input (angle) and stores result in specified register. The unit of input is degree, user can also use [DEG](#) instruction to convert the radian input into angle input.
2. This instruction may set [M8170](#) and [M8171](#) to ON.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	Input angle.	K/D/AI/AO	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(OUT)	Register stores the sine result.	D	-	float

Table 6.7. 1

Example

Instruction table:

Network 000

```
LD      X000
SIN     K30      D0 //SIN(30°) = D0D1 = 0.5
```

Ladder diagram:



Figure 6.7. 1

COS

Instruction introduction

1. This instruction calculates cosine of the float type input (angle) and stores result in specified register. The unit of input is degree, user can also use [DEG](#) instruction to convert the radian input into angle input.
2. This instruction may set [M8170](#) and [M8171](#) to ON.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	Input angle.	K/D/AI/AO	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(OUT)	Register stores the cosine result.	D	-	float

Table 6.8. 1

Example

Instruction table:

Network 000

```
LD      X000
COS     K60      D0 //COS(60°) = D0D1 = 0.5
```

Ladder diagram:



Figure 6.8. 1

TAN

Instruction introduction

1. This instruction calculates tangent of the float type input (angle) and stores result in specified register. The unit of input is degree, user can also use [DEG](#) instruction to convert the radian input into angle input.
2. This instruction may set [M8169](#), [M8170](#) and [M8171](#) to ON.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	Input angle.	K/D/AI/AO	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(OUT)	Register stores the tangent result.	D	-	float

Table 6.9. 1

Example

Instruction table:

Network 000

LD X000

TAN K45 D0 //TAN(45°) = D0D1 = 1

Ladder diagram:

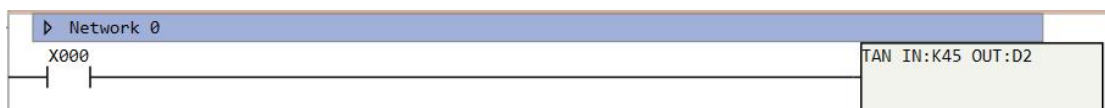


Figure 6.9. 1

LN

Instruction introduction

1. This instruction calculates natural logarithm (base is natural constant which is approximately equal to 2.71828) of the float type input and stores result in specified register.
2. This instruction may set [M8169](#), [M8170](#) and [M8171](#) to ON.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	Input antilogarithm.	K/D/AI/AO	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(OUT)	Register stores the result.	D	-	float

Table 6.10. 1

Example

Instruction table:

Network 000

```
LD      X000
LN      D0      D2 //LN(D0D1) = D2D3
```

Ladder diagram:



Figure 6.10. 1

EXP

Instruction introduction

1. This instruction calculates natural exponential (base is natural constant which is approximately equal to 2.71828) of the float type input and stores result in specified register.
2. This instruction may set [M8171](#) to ON.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	Input exponent.	K/D/AI/AO	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(OUT)	Register stores the result.	D	-	float

Table 6.11. 1

Example

Instruction table:

Network 000

```
LD      X000
EXP     K3      D2 //EXP(3) = D2D3 = e^3
```

Ladder diagram:

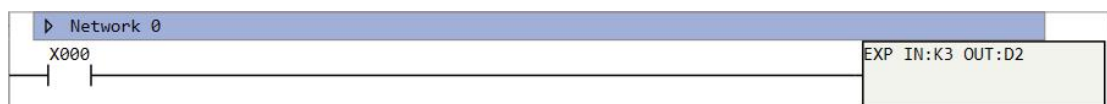


Figure 6.11. 1

LOG

Instruction introduction

1. This instruction calculates common logarithm (base is 10) of the float type input and stores result in specified register.
2. This instruction may set [M8169](#), [M8170](#) and [M8171](#) to ON.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	Input antilogarithm.	K/D/AI/AO	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(OUT)	Register stores the result.	D	-	float

Table 6.12. 1

Example

Instruction table:

Network 000

LD X000

LOG K1000 D4 //LOG(K1000) = D4D5 = 3

Ladder diagram:

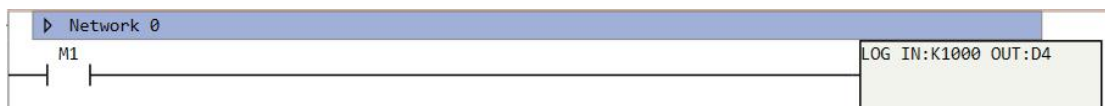


Figure 6.12. 1

POW

Instruction introduction

1. This instruction calculates one float type input (exponent) to the power of another one (base of power) and stores result in specified register.
2. This instruction may set [M8171](#) to ON.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(B)	Input base of power.	K/D	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(E)	Input exponent of power.	K/D	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(P)	Register stores the result.	D	-	float

Table 6.13. 1

Example

Instruction table:

Network 000

LD X000

POW K1000

D4 //LOG(K1000) = D4D5 = 3

Ladder diagram:

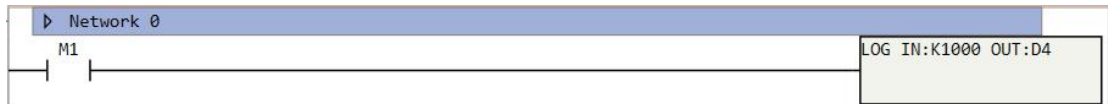


Figure 6.13. 1

ABSF

Instruction introduction

This instruction calculates absolute value of the float type input and stores result in specified register.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	Input to calculate absolute value.	K/D	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(OUT)	Register stores the result.	D	-	float

Table 6.14. 1

Example

Instruction table:

Network 000

LD M100

LOG D0 D10 //ABS(D0D1) = D10D11

Ladder diagram:

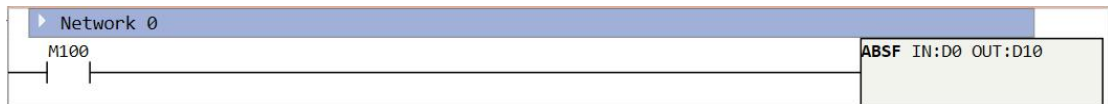


Figure 6.14. 1

RAD/DEG

Instruction introduction

1. **RAD** instruction converts degree to radian.
2. **DEG** instruction converts radian to degree.

Setting data

➤ RAD

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	Input degree.	K/D/AI/AO	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(OUT)	Register stores the radian result	D	-	float

Table 6.15. 1

➤ MVDBLK

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	Input degree.	K/D/AI/AO	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(OUT)	Register stores the radian result	D	-	float

Table 6.15. 2

Example

Instruction table:

Network 000

```

LD      X000
RAD     D0      D2 //convert degree in D0D1 to radian in D2D3
POP
LD      X001
RAD     D0      D2 // convert radian in D0D1 to degree in D2D3
POP
    
```

Ladder diagram:

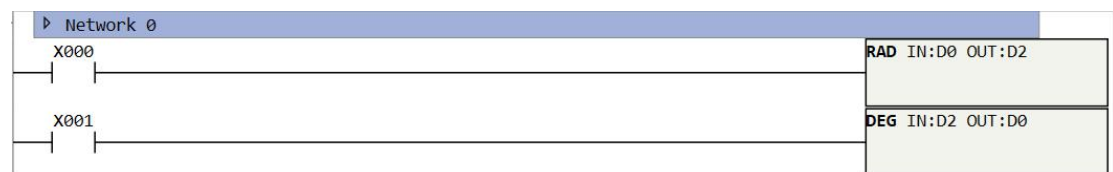


Figure 6.15. 1

Integer Calculation Instructions

Notes

1. Instructions in this chapter do integer calculations to integer inputs. The integer calculations include: addition, subtraction, multiplication, division, increment, decrement, factorial, and absolute value.
2. There are system special function bit-registers that record calculation errors.
 - M8169:** When overflow or underflow occurs in calculation, this bit-register is ON.
 - M8170:** When calculation result is minus or calculation input is illegal minus, this bit-register is ON.
 - M8171:** When calculation result is 0, this bit-register is ON.
 - M8172:** When divisor of division is 0, this bit-register is ON.

ADD/ADDD

Instruction introduction

1. These instructions add one integer type input (augend) with another (addend) and stores sum in specified register. **ADD** is for 16-bit signed integer, **ADDD** is for 32-bit signed integer.
2. This instruction may set [M8169](#), [M8170](#) and [M8171](#) to ON.

Setting data

➤ ADD

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN1)	Augend of addition.	K/H/D/TV/CV/AI/AO/V/Z	-32768~32767	16-bit signed integer
(IN2)	Addend of addition.	K/H/D/TV/CV/AI/AO/V/Z	-32768~32767	16-bit signed integer
(OUT)	Register stores the sum	D/TV/CV/AO/V/Z	-	16-bit signed integer

Table 7.2. 1

➤ ADDD

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN1)	Augend of addition.	K/H/D/CV	-2147483648~2147483647	32-bit signed integer
(IN2)	Addend of addition.	K/H/D/CV	-2147483648~2147483647	32-bit signed integer
(OUT)	Register stores the sum	D/CV	-	32-bit signed integer

Table 7.2. 2

Example

Instruction table:

Network 000

```
LD      X000
ADD     D0      D1      D2 //D0+D1=D2
```

Ladder diagram:

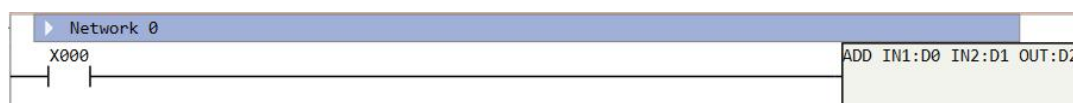


Figure 7.2. 1

SUB/SUBD

Instruction introduction

1. These instructions subtract one integer type input (minuend) with another (subtrahend) and stores difference in specified register. **SUB** is for 16-bit signed integer, **SUBD** is for 32-bit signed integer.
2. This instruction may set [M8169](#), [M8170](#) and [M8171](#) to ON.

Setting data

➤ SUB

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN1)	Minuend of subtraction.	K/H/D/TV/CV/AI/AO/V/Z	-32768~32767	16-bit signed integer
(IN2)	Subtrahend of subtraction.	K/H/D/TV/CV/AI/AO/V/Z	-32768~32767	16-bit signed integer
(OUT)	Register stores the difference.	D/TV/CV/AO/V/Z	-	16-bit signed integer

Table 7.3. 1

➤ SUBD

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN1)	Minuend of subtraction.	K/H/D/CV	-2147483648~2147483647	32-bit signed integer
(IN2)	Subtrahend of subtraction.	K/H/D/CV	-2147483648~2147483647	32-bit signed integer
(OUT)	Register stores the difference.	D/CV	-	32-bit signed integer

Table 7.3. 2

Example

Instruction table:

Network 000

```
LD      X000
SUB     D0      D1      D2 //D0-D1=D2
```

Ladder diagram:

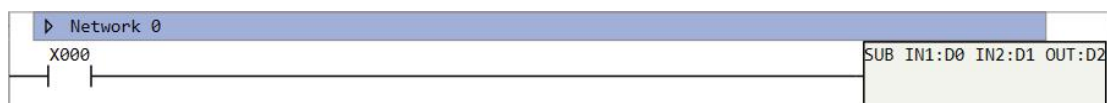


Figure 7.3. 1

MUL/MULW/MULD

Instruction introduction

1. These instructions multiply one integer type input (multiplicand) with another (multiplier) and stores product in specified register. **MUL** is for 16-bit signed integer inputs and 16-bit signed integer output; **MULW** is for 16-bit signed integer inputs and 32-bit signed integer output; **MULD** is for 32-bit integer inputs and 32-bit signed integer output.
2. This instruction may set [M8170](#) and [M8171](#) to ON.

Setting data

➤ MUL

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN1)	Multiplicand of multiplication.	K/H/D/TV/CV/AI/AO/V/Z	-32768~32767	16-bit signed integer
(IN2)	Multiplier of multiplication.	K/H/D/TV/CV/AI/AO/V/Z	-32768~32767	16-bit signed integer
(OUT)	Register stores the product.	D/CV	-	16-bit signed integer

Table 7.4. 1

➤ MULW

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN1)	Multiplicand of multiplication.	K/H/D/TV/CV/AI/AO/V/Z	-32768~32767	16-bit signed integer
(IN2)	Multiplier of multiplication.	K/H/D/TV/CV/AI/AO/V/Z	-32768~32767	16-bit signed integer
(OUT)	Register stores the product.	D/CV	-	32-bit signed integer

Table 7.4. 2

➤ MULD

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN1)	Multiplicand of multiplication.	K/H/D/CV	-2147483648~2147483647	32-bit signed integer
(IN2)	Multiplier of multiplication.	K/H/D/CV	-2147483648~2147483647	32-bit signed integer
(OUT)	Register stores the product.	D/CV	-	32-bit signed integer

Table 7.4. 3

Example

Instruction table:

Network 000

LD X000

MUL D0 D1 D2 //D0×D1=D2D3

Ladder diagram:

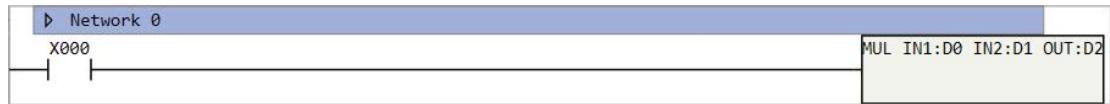


Figure 7.4. 1

DIV/DIVW/DIVD

Instruction introduction

1. These instructions divide one integer type input (dividend) with another (divisor) and stores quotient in specified register. **DIV** is for 16-bit signed integer inputs and 16-bit signed integer output; **DIVW** is for 16-bit signed integer inputs and 32-bit signed integer output; **DIVD** is for 32-bit integer inputs and 32-bit signed integer output.
2. This instruction may set [M8170](#) and [M8171](#) to ON.

Setting data

➤ DIV

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN1)	Dividend of division.	K/H/D/TV/CV/AI/AO/V/Z	-32768~32767	16-bit signed integer
(IN2)	Divisor of division.	K/H/D/TV/CV/AI/AO/V/Z	-32768~32767	16-bit signed integer
(OUT)	Register stores the quotient.	D/TV/CV/AO/V/Z		16-bit signed integer

Table 7.5. 1

➤ DIVW

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN1)	Dividend of division.	K/H/D/TV/CV/AI/AO/V/Z	-32768~32767	16-bit signed integer
(IN2)	Divisor of division.	K/H/D/TV/CV/AI/AO/V/Z	-32768~32767	16-bit signed integer
(OUT)e	Register stores the quotient.	D/CV	-	32-bit signed integer

Table 7.5. 2

➤ DIVD

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN1)	Dividend of division.	K/H/D/CV	-2147483648~2147483647	32-bit signed integer
(IN2)	Divisor of division.	K/H/D/CV	-2147483648~2147483647	32-bit signed integer
(OUT)	Register stores the quotient.	D/CV	-	32-bit signed integer

Table 7.5. 3

Example

Instruction table:

Network 000

```
LD      X000
DIV     D0      D1      D2 //D0=D1=D2D3
```

Ladder diagram:

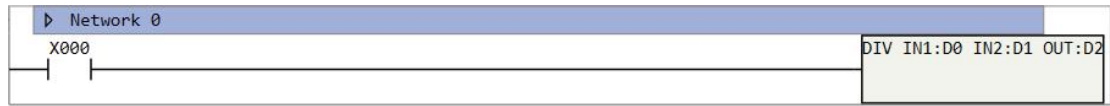


Figure 7.5. 1

INC/INCD

Instruction introduction

1. These instructions do increment operation on integer type input and stores result in specified register. **INC** is for 16-bit signed integer, **INCD** is for 32-bit signed integer.
2. This instruction may set [M8169](#), [M8170](#) and [M8171](#) to ON.

Setting data

➤ INC

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	Input to increase.	K/H/D/TV/CV/AI/AO/V/Z	-32768~32767	16-bit signed integer
(OUT)	Register stores the result.	D/TV/CV/AO/V/Z	-	16-bit signed integer

Table 7.6. 1

➤ INCD

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	Input to increase.	K/H/D/CV	-2147483648~2147483647	32-bit signed integer
(OUT)	Register stores the result.	D/CV	-	32-bit signed integer

Table 7.6. 2

Example

Instruction table:

Network 000

```
LD      X000
INC     D0      D1 //D0 = D0 + 1
```

Ladder diagram:



Figure 7.6. 1

DEC/DECD

Instruction introduction

1. These instructions do decrement operation on integer type input and stores result in specified register. **INC** is for 16-bit signed integer, **INCD** is for 32-bit signed integer.
2. This instruction may set [M8169](#), [M8170](#) and [M8171](#) to ON.

Setting data

➤ INC

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	Input to decrease.	K/H/D/TV/CV/AI/AO/V/Z	-32768~32767	16-bit signed integer
(OUT)	Register stores the result.	D/TV/CV/AO/V/Z	-	16-bit signed integer

Table 7.7. 1

➤ INCD

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	Input to decrease.	K/H/D/CV	-2147483648~2147483647	32-bit signed integer
(OUT)	Register stores the result.	D/CV	-	32-bit signed integer

Table 7.7. 2

Example

Instruction table:

Network 000

```
LD      X000
DEC     D0      D1 //D0 = D0 - 1
```

Ladder diagram:

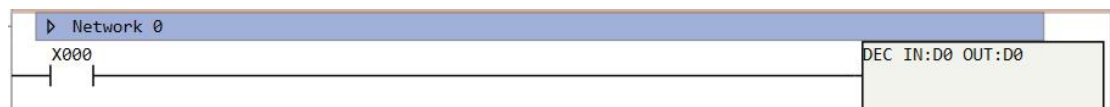


Figure 7.7. 1

FACT

Instruction introduction

1. This instruction calculates factorial of integer type input and stores result in specified register.
2. The range of input is 0 to 12.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	Input to decrease.	K/H/D/TV/CV/AI/AO/V/Z	0~12	16-bit signed integer
(OUT)	Register stores the result.	D	-	16-bit signed integer

Table 7.8. 1

Example

Instruction table:

Network 000

LD X000

FACT K4 D0 //D0D1 = 4! = 24

Ladder diagram:

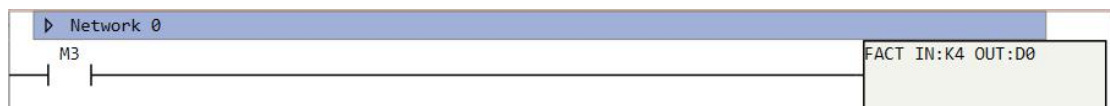


Figure 7.8. 1

ABS/ABSD

Instruction introduction

These instructions calculate absolute value of the integer type input and stores result in specified register. **ABS** is for 16-bit signed integer input, **ABSD** is for 32-bit signed integer input.

Setting data

➤ ABS

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	Input to calculate absolute value.	K/H/D/TV/CV/AI/AO/V/Z	-32768~32767	16-bit signed integer
(OUT)	Register stores the result.	D/TV/CV/AO/V/Z	-	16-bit signed integer

Table 7.7. 3

➤ ABSD

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	Input to calculate absolute value.	K/H/D/CV	-2147483648~2147483647	32-bit signed integer
(OUT)	Register stores the result.	D/CV		32-bit signed integer

Table 7.7. 4

Example

Instruction table:

Network 000

LD M100

ABS D0 D10 //ABS(D0D1) = D10D11

Ladder diagram:

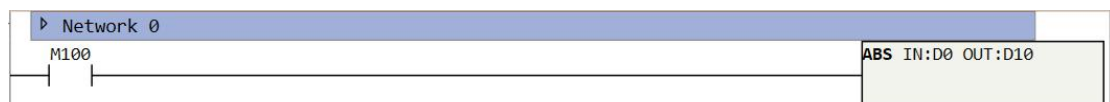


Figure 7.9. 1

Timer Instructions

Notes

1. Instructions in this chapter record time counting in corresponding timer and set flag-bit of timer in specific condition.
2. There are 3 kinds of timers, they differ from resolution. Resolution is the minimum time interval that count of timer increases. [Table 8.1.1](#) shows the details.

Resolution	Maximum timing	Timer number
100ms	6553.5s	T0~T199
10ms	655.35s	T200~T249
1ms	65.535s	T250~T255

Table 8.1. 1

For timers of different, their refresh ways are also different.

- 1ms resolution timer:
Timer refreshes once in each 1ms interval, it depends on system interruption, regardless of scan period or programmer processing of PLC. Timer may refresh more than once when scan period is longer than 1ms, value of timer count (TV register) and timer flag-bit (T register) may be inconsistent during one scan period.
 - 10ms resolution timer:
Timer refreshes in each scan period automatically, so that value of timer count (TV register) and timer flag-bit (T register) are consistent during one scan period.
 - 100ms resolution timer:
Timers refreshed only when timer instruction is processing, if instruction is not processed or processed more than once during one scan period, the time count will be inaccurate. This timer is only allowed to be used once in program.
3. Different type of timer instructions can't share one timer. Additionally, it is recommended to add only one timer instruction of one timer in a program.

TON

Instruction introduction

1. This instruction is on-delay timer instruction. When instruction is enabled, timer starts to time. The timer flag-bit will be set to ON when timer count meets target time; When instruction is disabled, timer count will be cleared and timer flag-bit will be set to OFF. Details refer to [Figure 8.2.1](#).

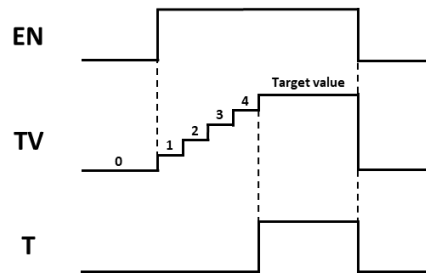


Figure 8.2. 1

2. User can use this instruction for single target time counting.
3. Reset flag-bit of timer can also clear corresponding timer count.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(T)	Selected timer.	TV		16-bit unsigned integer
(SV)	Target time of timer.	K/H/D	0~65535	16-bit unsigned integer

Table 8.2. 1

Example

Instruction table:

Network 000

```
LD X000
TON TV10 D0 //when timer count of TV10 meet target time D0, T10 is set to ON
```

Network 001

```
LD T10
OUT Y000
```

Ladder diagram:

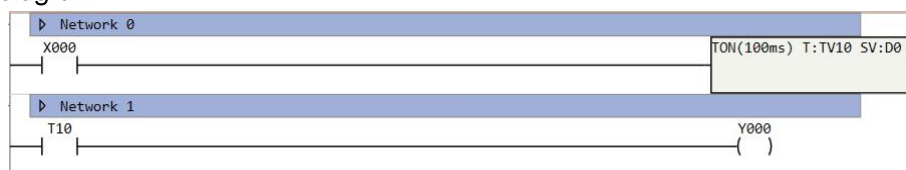


Figure 8.2. 2

TONR

Instruction introduction

1. This instruction is on-delay memory timer instruction. When instruction is enabled, timer starts (continues) to time. The timer flag-bit will be set to ON when timer count meets target time; When instruction is disabled, timer count and time flag-bit will be sustained. Details refer to [Figure 8.3.1](#).

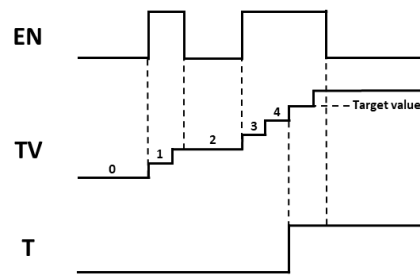


Figure 8.3. 1

2. User can use this instruction for multiple target time counting.
3. Reset flag-bit of timer can also clear corresponding timer count.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(T)	Selected timer.	TV		16-bit unsigned integer
(SV)	Target time of timer.	K/H/D	0~65535	16-bit unsigned integer

Table 8.3. 1

Example

Instruction table:

Network 000

```
LDI Y000
TONR TV251 K100 //when timer count of TV251 meet target time K100, T251 is set to ON
```

Network 000

```
LD T251
OUT Y000 //when Y000 is ON, timer instruction is disabled.
```

Ladder diagram:

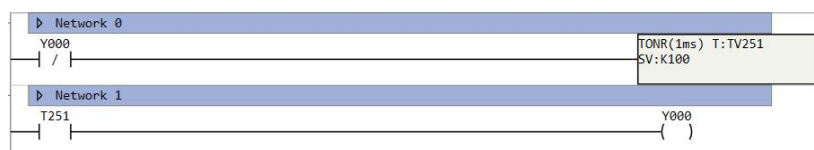


Figure 8.3. 2

TOF

Instruction introduction

1. This instruction is off-delay timer instruction. When instruction is disabled, timer starts (continues) to time. The timer flag-bit will be set to OFF when timer count meets target time; When instruction is enabled, timer count will be cleared and timer flag-bit will be set to ON. Details refer to [Figure 8.4.1](#).

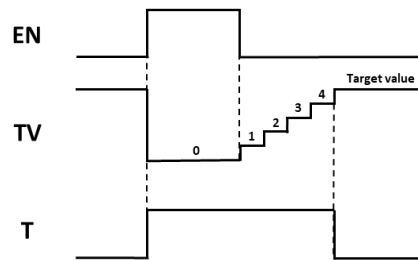


Figure 8.4. 1

2. User can use this instruction for single target time counting.
3. Reset flag-bit of timer can also clear corresponding timer count.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(T)	Selected timer.	TV		16-bit unsigned integer
(SV)	Target time of timer.	K/H/D	0~65535	16-bit unsigned integer

Table 8.4. 1

Example

Instruction table:

Network 000

```
LDI Y000
TONR TV251 K100 //when timer count of TV251 meet target time K100, T251 is set to ON
```

Network 000

```
LD T251
OUT Y000 //when Y000 is ON, timer instruction is disabled.
```

Ladder diagram:

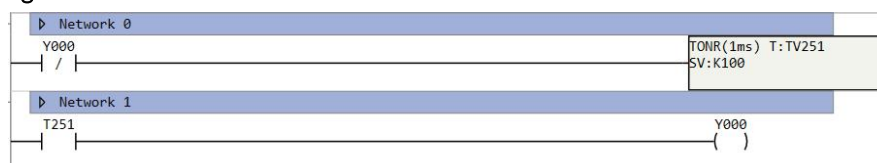


Figure 8.4. 2

Counter Instructions

Notes

1. Instructions in this chapter record signal-edge counting in corresponding counter and set flag-bit of counter when count meets target number.
2. There are 3 kinds of counters, they differ from their data bits and apply occasion. [Table 9.1.1](#) shows the details.

Counter type	Counting range	Timer number
16-bit	-32768~32767	C0~C199
32-bit	-2147483648~2147483647	C200~C249
32-bit (high-speed)	-2147483648~2147483647	C250~C255

Table 9.1. 1

High-speed counter is applied for high-speed pulse counting.

3. It is recommended to add only one type timer instruction of one timer in a program.

CTU

Instruction introduction

1. This instruction counts upward (count increases) at each rising-edge of input. When the count meet set target number, corresponding flag-bit will be set to ON.
2. Reset flag-bit of counter will also clear count of counters.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(IN)	Input that trigger the counting.		0/1	bool
(T)	Selected counter.	CV		■ C0~C199 16-bit signed integer ■ C200~C255 32-bit signed integer
(SV)	Target number of counting.	K/H/D	■ C0~C199 -32768~32767 ■ C200~C255 -2147483648~2147483647	■ C0~C199 16-bit signed integer ■ C200~C255 32-bit signed integer

Table 9.2. 1

Example

Instruction table:

Network 000

```
LD    X000
CTU   CV0
```

D0 //counter counts upward, when counter count of CV0 meet target number D0, C0 is set to ON

Network 000

```
LD    C0
OUT   Y000
```

Ladder diagram:

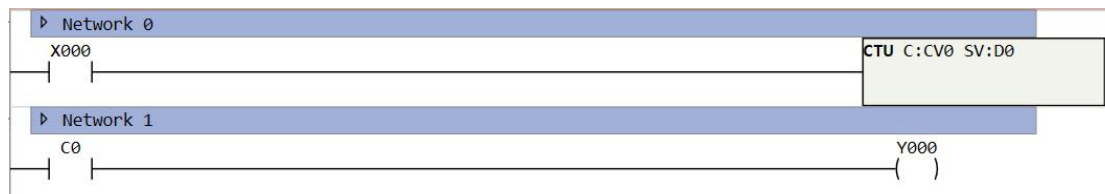


Figure 9.2. 1

CTD

Instruction introduction

1. This instruction counts downward (count decreases) at each rising-edge of input. When the count meet set target number, corresponding flag-bit will be set to ON.
2. Reset flag-bit of counter will also clear count of counters.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(IN)	Input that trigger the counting.		0/1	bool
(T)	Selected counter.	CV		■ C0~C199 16-bit signed integer ■ C200~C255 32-bit signed integer
(SV)	Target number of counting.	K/H/D	■ C0~C199 -32768~32767 ■ C200~C255 -2147483648~2147483647	■ C0~C199 16-bit signed integer ■ C200~C255 32-bit signed integer

Table 9.3. 1

Example

Instruction table:

Network 000

```
LD    X000
CTD   CV0
```

D0 //counter counts downward, when counter count of CV0 meet target number D0, C0 is set to ON

Network 000

```
LD    C0
OUT   Y000
```

Ladder diagram:

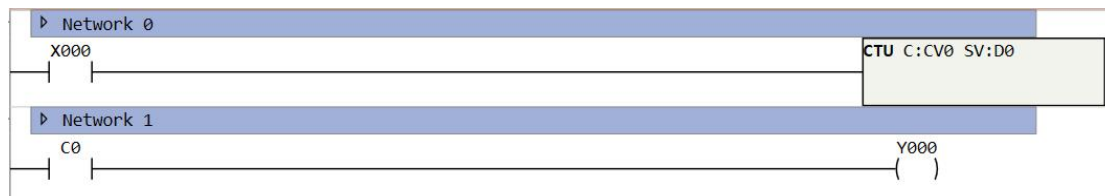


Figure 9.3. 1

CTUD

Instruction introduction

1. This instruction counts upward or downward at each rising-edge of input. The counting direction depends on comparison between target number and current counts, when target number is greater than current counts, counter counts upward; When target number is less than current counts, counter counts downward; When the count meet target number, corresponding flag-bit will be set to ON.
2. Reset flag-bit of counter will also clear count of counters.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(IN)	Input that trigger the counting.		0/1	bool
(T)	Selected counter.	CV		■ C0~C199 16-bit signed integer ■ C200~C255 32-bit signed integer
(SV)	Target number of counting.	K/H/D	■ C0~C199 -32768~32767 ■ C200~C255 -2147483648~2147483647	■ C0~C199 16-bit signed integer ■ C200~C255 32-bit signed integer

Table 9.4. 1

Example

Instruction table:

Network 000

```
LD      X000
CTUD   CV0      D0 //compare target number and current counts, and counter counts upward or
                        downward, when counter count of CV0 meet target number D0, C0 is set to ON
```

Network 000

```
LD      C0
OUT     Y000
```

Ladder diagram:

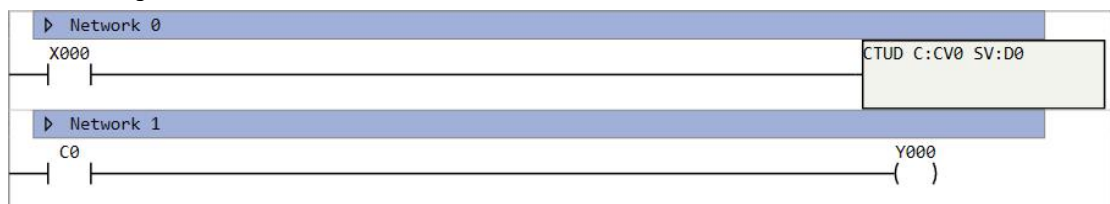


Figure 9.4. 1

Program Control Instructions

Notes

1. Instructions in this chapter control location of program execution. Usually, program execution in sequence, by using these instructions, program execution can realize function like loop, jump, call, and so on.
2. Program of PLC project can be divided into 3 types: main program, subroutine program, and function block. Main program is the entrance of execution, execution start from main program; Subroutine program is branch of execution, user can use program control instructions to call it and back to main program; Function block is a kind of user-defined instruction that user can edit by C programming language, user can also call it in program like other instructions.
3. S register is system state bit-register, it marks state of system and takes effects in state instructions (**STL/STLE/ST** and **IST**).

FOR~NEXT

Instruction introduction

1. **FOR** instruction must be used together with **NEXT** instruction in pairs. When **FOR** is enabled, program segments between **FOR** and **NEXT** is loop program that will be executed for specified loop number, then continue after **NEXT**. When **FOR** is disabled, the program between FOR and NEXT will be skipped.
2. Loop number can be modified when program is running. Once program reenters the loop, it will loop for modified loop number.
3. It is not recommended to add timer in the loop program.
4. The maximum number of nested loops is 8.
5. User can use [CJ](#) or [BREAK](#) instruction to jump out the cycle.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(CNT)	Loop number.	K/H/D	0~65535	16-bit unsigned integer

Table 10.2. 1

Example

Instruction table:

Network 000

```
LDP X000
FOR K10 //loop numbers is set with 10
```

Network 000

```
LD X001
INC D0 D0 //D0 increase in each cycle
```

Network 000

```
NEXT //end of loop
```

Ladder diagram:

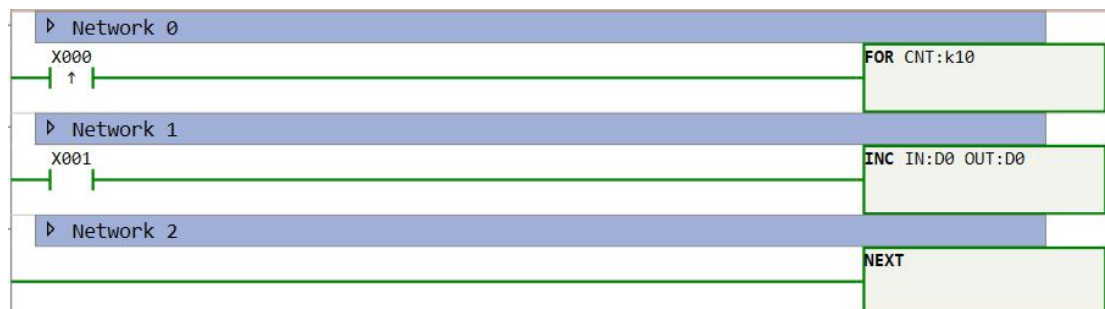


Figure 10.2. 1

CJ~LBL

Instruction introduction

1. **CJ** instruction must be used together with **LBL** instruction in pairs. When **CJ** is enabled, program will jump to specified **LBL** label; When **CJ** is disabled, program executes as usual sequences.
2. **CJ** instruction and its corresponding **LBL** label must be in same program (main program or subroutine program). It cannot jump from main program to subroutine program, also it cannot jump out from subroutine program.

Setting data

➤ CJ

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(LBL)	Number of label to jump to.	K/H	0~65535	16-bit unsigned integer

Table 10.3. 1

➤ LBL

Inputs/outputs	Description	Operand	Range	Data type
(L)	Number of label.	K/H	0~65535	16-bit unsigned integer

Table 10.3. 2

Example

Instruction table:

Network 000

```
LD X000
CJ K0 //if X000 is ON, jump to LBL K0
```

Network 000

```
LD X001
ADD D0 D1 D2 //if X000 is OFF, D2=D0+D1 when X001 is ON
```

Network 000

```
LBL K0 //set label number with K0
```

Ladder diagram:

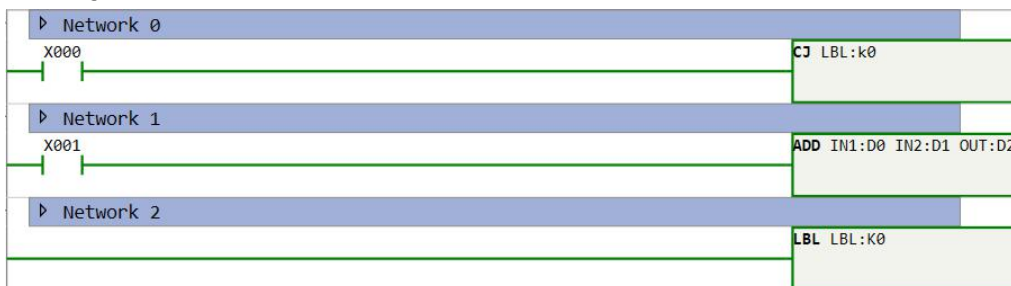


Figure 10.3. 1

CALL

Instruction introduction

1. When this instruction is enabled, program execution will jump to start of the set subroutine program; When this instruction is disabled, this instruction will be skipped and continue the following program.
2. User should create a subroutine program at first, and user can also add this instruction by dragging the subroutine program from project interface to ladder diagram interface.
3. The maximum number of nested calls is 8.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(P)	Subroutine program to call.	-	-	pointer

Table 10.4. 1

Example

Instruction table:

Network 000

LD X000

CALL Sub1 //call subroutine program "Sub1"

Ladder diagram:



Figure 10.4. 1

CALLM

Instruction introduction

1. When this instruction is enabled, program will call specified function block; When this instruction is disabled, this instruction will be skipped and continue the following program.
2. This instruction should be triggered by an edge signal.
3. Details of function block refer to "[Build a function block](#)" in appendix.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(F)	Function block to call.	-	-	pointer
(P1)	Parameter of function.	-	-	any
(P2)	Parameter of function.	-	-	any
(P3)	Parameter of function.	-	-	any
(P4)	Parameter of function	-	-	any

Table 10.5. 1

Example

1. Sort function

Function block:

```
// Initialize the data
void Data_Init(WORD data, WORD size)
{
    short i = 0;
    for ( i = 0; i < *size; i++)
        data[i] = (i ^ 0x4321); //do XOR operation on each data with 0x4321
}

// sort data from smallest to biggest
void Data_Sort(WORD data, WORD size)
{
    short i = 0;
    short j = 0;
    short temp = 0;
    for ( i = 0; i < *size; i++)
    {
        temp = data[i];
        for (j = i; j > 0 && data[j-1] > temp; j--)
            data[j] = data[j-1]; // keep bubbling forward
        data[j] = temp;
    }
}
```

Instruction table:

```

LDP      M0 //call function Data_Init once when M0 is ON, D[0]=0x4321, D[1]=0x4320, D[2]=0x4323,
          D[3]=0x4322, D[4]=0x4325, D[5]=0x4324, D[6]=0x4327
MOV      K7      D100 //set data length D100 with 7
CALLM    Data_Init  D0      D100 //call Data_Init, data=D0, size=D100
POP
LDP      M1 //call function Data_Sort when M1 is ON, D[0]=0x4320, D[1]=0x4321, D[2]=0x4322, D[3]=0x4323,
          D[4]=0x4324, D[5]=0x4325, D[6]=0x4327
CALLM    Data_Sort  D0      D100 //call Data_Sort, data=D0, size=D100
POP
LDW<    D1      D2
AW<     D2      D3
AW<     D3      D4
AW<     D4      D5
AW<     D5      D6
OUT      Y000 //if D1<D2<D3<D4<D5<D6, Y000 is ON
POP

```

Ladder diagram:

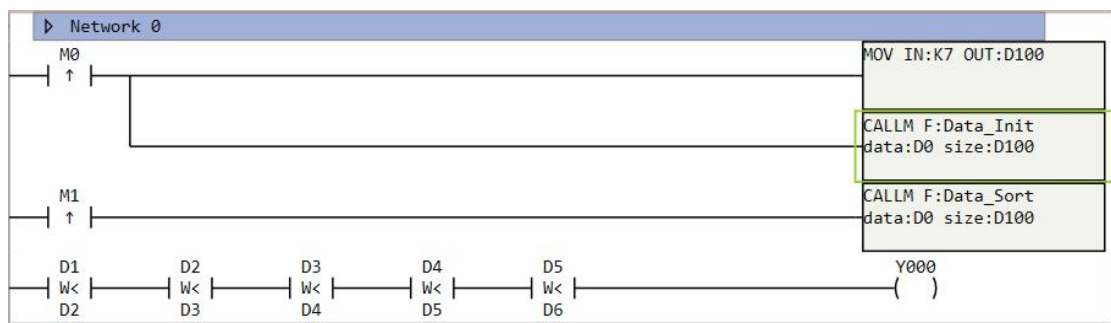


Figure 10.5. 1

2. Calculate standard deviation

Function block:

```

// calculate the average
void Average(FLOAT in, WORD size, FLOAT out)
{
    *out = 0.0;
    short i = 0;
    for (; i < *size; i++)
        *out += in[i];
    *out /= *size;
}

// calculate the standard deviation
[Array(in, size)]
void StandardDeviation(FLOAT in, WORD size, FLOAT out)
{
    float ave = 0.0;
    Average(in, size, &ave);
    short i = 0;
    *out = 0.0;
    for (; i < *size; i++)
        *out += (in[i] - ave) * (in[i] - ave);
    *out /= *size;
    *out = sqrt(*out);
}

```

Instruction table:

```

LDP      M0
MOV      K10     D100 //set length of data D100 with 10
CALLM    StandardDeviation  D0      D100      D200 // call function StandardDeviation, in=D0,
          size=D100, out=D200

```


Ladder diagram:

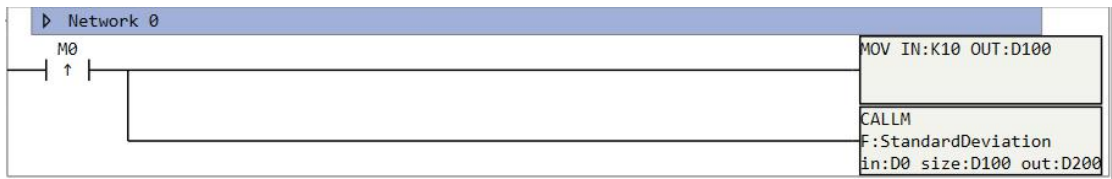


Figure 10.5. 2

STL/STLE/ST

Instruction introduction

1. These instructions are step ladder instructions, **STL** starts step ladder, **STLE** ends step ladder, **ST** switches on step ladder. Additionally, resetting corresponding S bit-register by using **RST** instruction can switch off step ladder.
2. When use **RST** to switch off a step ladder, all output (output contact, timer, counter, pulse output, e.g.) in the step ladder will be reset or clear.
3. **STL** must be used together **STLE**, and step ladder between **STL** and **STLE** can't be nested.
4. Setting corresponding S bit-register by using **SET** instruction can also switch on step ladder. The difference is, when use **SET**, the set S bit-register will be reset automatically in next scan period; When use **ST**, the set S-bit-register will hold ON.

Setting data

➤ STL

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(L)	Flag-bit of step ladder.	S	0/1	bool

Table 10.6. 1

➤ ST

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(S)	Target step ladder to switch on.	S	0/1	bool

Table 10.6. 2

Example

1. Selectable branches

In this example, user can select one or both of two step ladder branches to execute, [Figure 10.6.1](#) and following diagram show the detail.

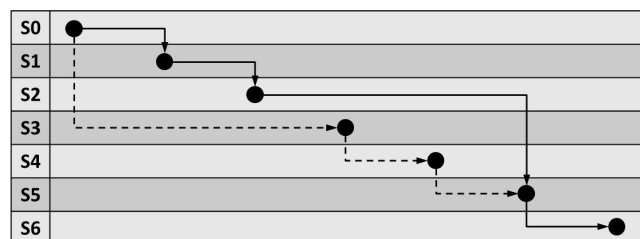


Figure 10.6. 1

Instruction table:

Network 000

```
LD      X000
SET     S0      K1 //go to S0 step ladder when X000 is ON
POP
```

Network 001

```
STL     S0 //start of S0 step ladder
LD      M8151
OUT     Y000
POP
LD      X000
SET     S1      K1 //go to S1 step ladder when X000 is ON
POP
LD      X003
SET     S3      K1 //go to S3 step ladder when X003 is ON
POP
STLE //end of S0 step ladder
```

Network 002

```
STL     S1 //start of S1 step ladder
LD      M8151
OUT     Y001
POP
LD      X001
SET     S2      K1 //go to S2 step ladder when X001 is ON
POP
STLE //end of S1 step ladder
```

Network 003

```
STL     S2 //start of S2 step ladder
LD      M8151
OUT     Y002
POP
LD      X002
SET     S5      K1 //go to S5 step ladder when X002 is ON
POP
STLE //end of S2 step ladder
```

Network 004

```
STL     S3 //start of S3 step ladder
LD      M8151
OUT     Y003
POP
LD      X004
SET     S4      K1 //go to S4 step ladder when X004 is ON
POP
STLE //end of S3 step ladder
```

Network 005

```
STL     S4 //start of S4 step ladder
LD      M8151
OUT     Y004
POP
LD      X005
SET     S5      K1 //go to S5 step ladder when X005 is ON
POP
STLE //end of S4 step ladder
```

Network 006

```
STL     S5 //start of S5 step ladder
LD      M8151
OUT     Y005
POP
LD      X006
SET     S6      K1 //go to S6 step ladder when X006 is ON
POP
STLE //end of S5 step ladder
```

Ladder diagram:

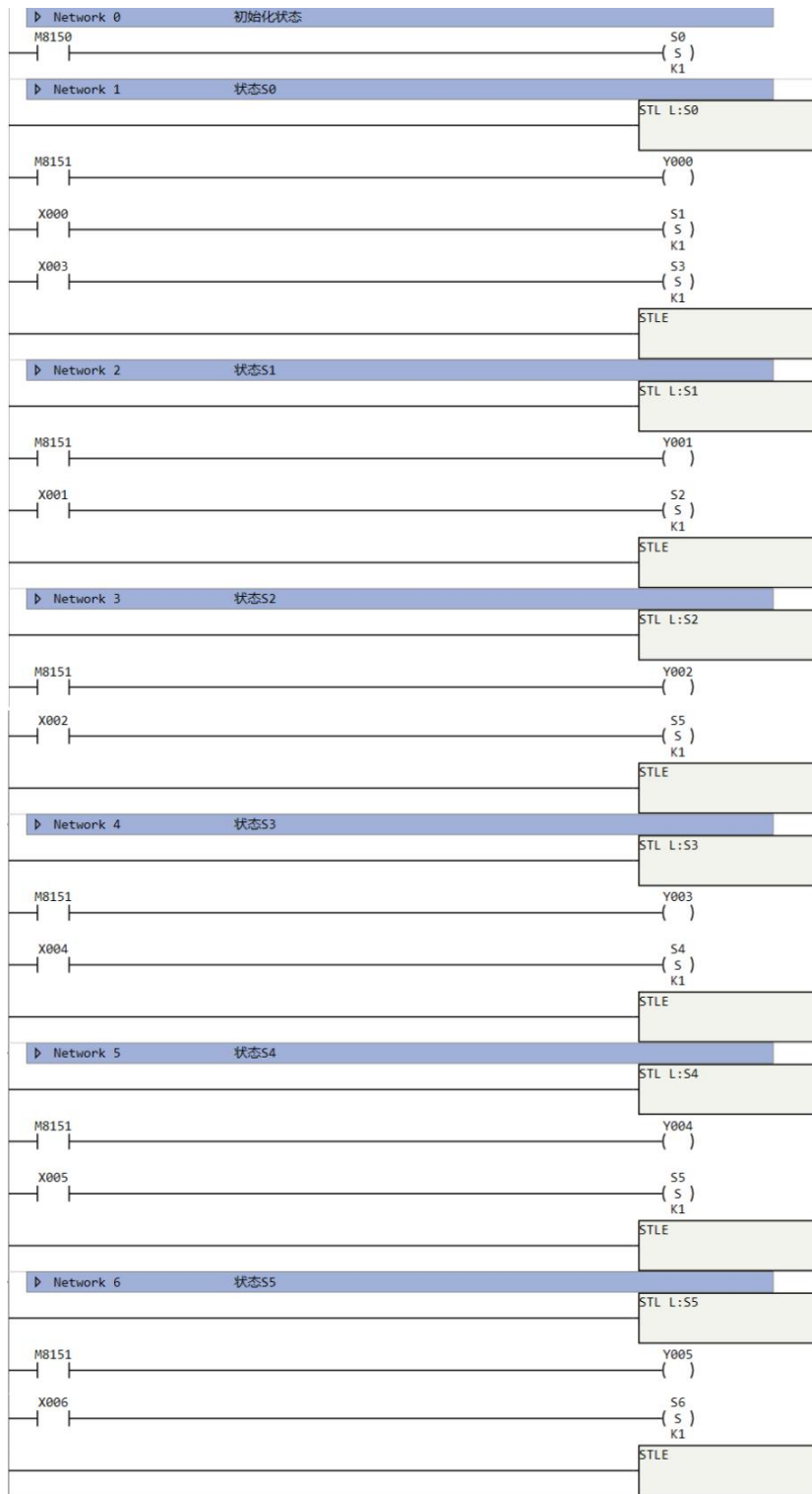


Figure 10.6. 2

2. Parallel branches

In this example, two step ladder branches are parallel, both are executed. [Figure 10.6.3](#) and following diagram show the detail.

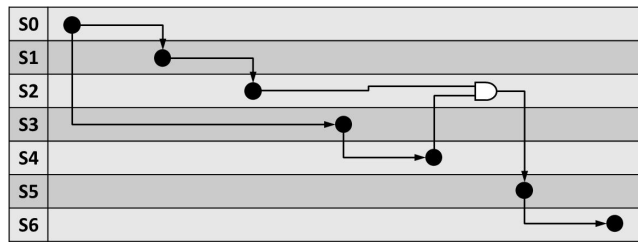


Figure 10.6. 3

Instruction table:

Network 000

```
LD      X000
SET     S0      K1 //go to S0 step ladder when X000 is ON
POP
```

Network 001

```
STL     S0 //start of S0 step ladder
LD      M8151
OUT     Y000
POP
LD      X000
SET     S1      K1 //go to S1 step ladder when X000 is ON
SET     S3      K1 //go to S3 step ladder when X003 is ON
POP
STLE //end of S0 step ladder
```

Network 002

```
STL     S1 //start of S1 step ladder
LD      M8151
OUT     Y001
POP
LD      X001
SET     S2      K1 //go to S2 step ladder when X001 is ON
POP
STLE //end of S1 step ladder
```

Network 003

```
STL     S2 //start of S2 step ladder
LD      M8151
OUT     Y002
POP
STLE //end of S2 step ladder
```

Network 004

```
STL     S3 //start of S3 step ladder
LD      M8151
OUT     Y003
POP
LD      X002
SET     S4      K1 //go to S4 step ladder when X002 is ON
POP
STLE //end of S3 step ladder
```

Network 005

```
STL     S4 //start of S4 step ladder
LD      M8151
OUT     Y004
POP
STLE //end of S4 step ladder
```

Network 006

```
LD      S2
AND     S4
AND     X003
SET     S5      K1 //go to S5 step ladder when X003 is ON
RST     S2      K1 //S2 need to be reset manually
RST     S4      K1 //S4 need to be reset manually
POP
```

Network 007

```

STL      S5 //start of S5 step ladder
LD       M8151
OUT      Y005
POP
LD       X004
SET      S6      K1 //go to S6 step ladder when X004 is ON
POP
STLE    //end of S5 step ladder
    
```

Ladder diagram:

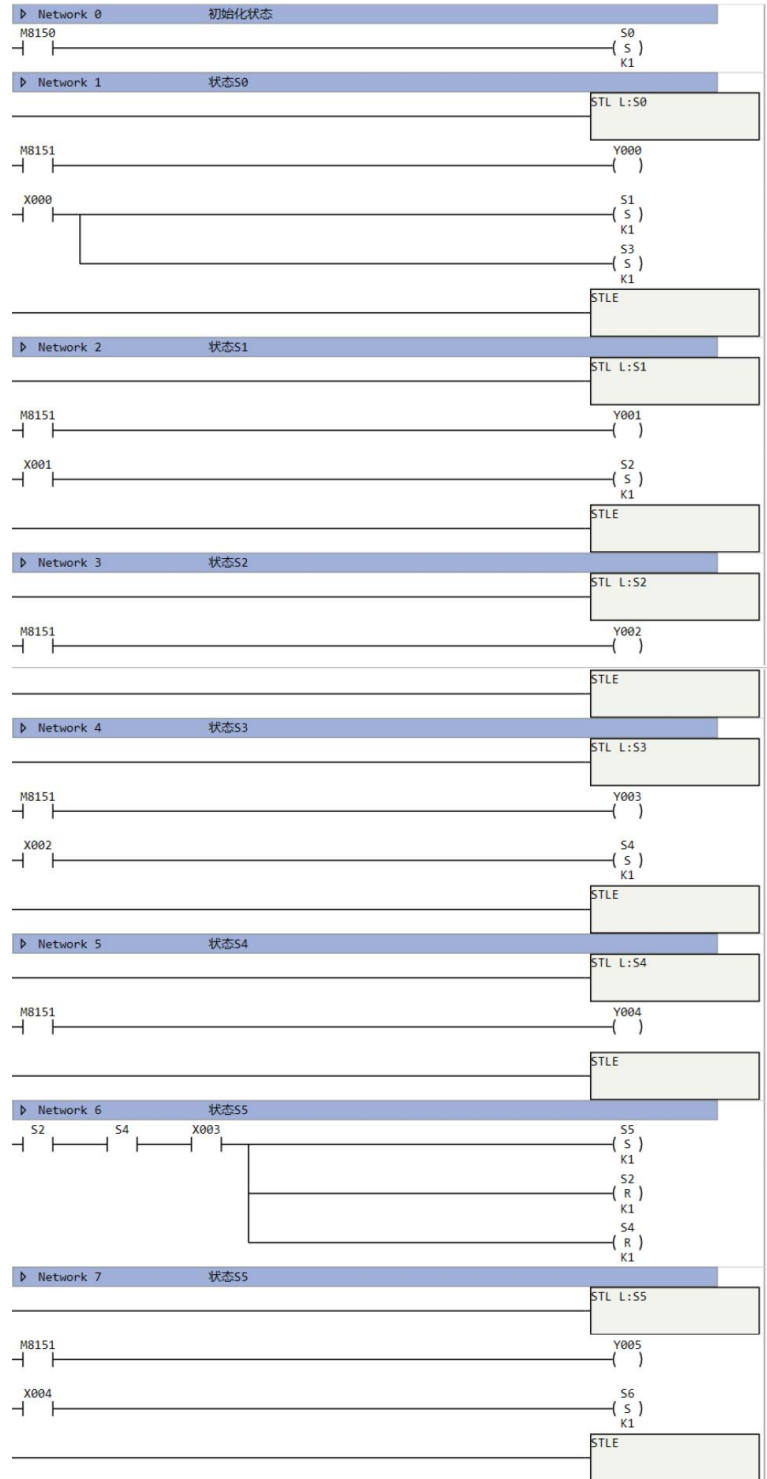


Figure 10.6. 4

RET

Instruction introduction

1. This instruction is used in subroutine program, and not allowed to be used in main program.
2. When this instruction is enabled, subroutine program will not execute left program after this instruction, but back to main program, and continue execution at where subroutine program is called.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool

Table 10.7. 1

Example

1. Main program

Instruction table:

Network 000

```
LD X000
CALL SUB1 //call subroutine program "SUB1"
```

Network 001

```
LD X001
ADD D0 D1 D2 //D2=D0+D1 when X001 is ON
```

Ladder diagram:

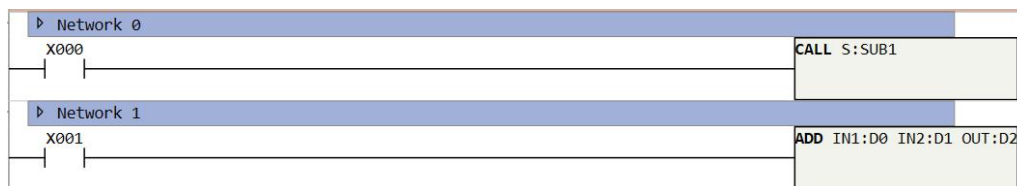


Figure 10.7. 1

2. Subroutine program

Instruction table:

Network 000

```
LD X001
RET //back to main program when X001 is ON
```

Network 001

```
LD X002
SUB D0 D1 D2 //D2=D0-D1 when X001 is OFF and X002 is ON
```

Ladder diagram:

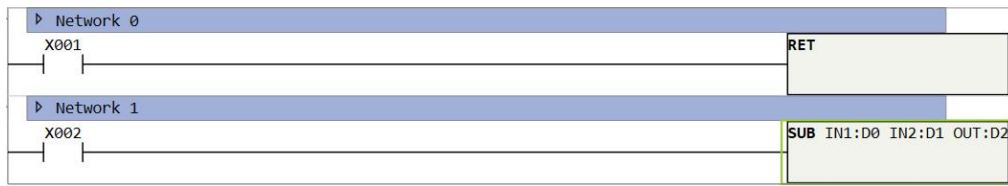


Figure 10.7. 2

BREAK

Instruction introduction

When this instruction is enabled, if it is in a loop part (**FOR** to **NEXT**), it will jump out from this loop and continue program after **NEXT**; If it is in a step ladder (**STL** to **STLE**), it will jump out from this step ladder and continue program after **STLE**. If it is in both a loop part and a step ladder, it will jump out from the loop part.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool

Table 10.8. 1

Example

1. FOR to NEXT

Instruction table:

Network 000

```
LD      X000
FOR     K100 //start of loop, loop for 100 times
```

Network 001

```
LD      X001
INC     D0      D0 //D0 increases in each loop
```

Network 002

```
LD      X002
BREAK  //jump out from loop when X002 is ON
```

Network 003

```
NEXT  //end of loop
```

Ladder diagram:

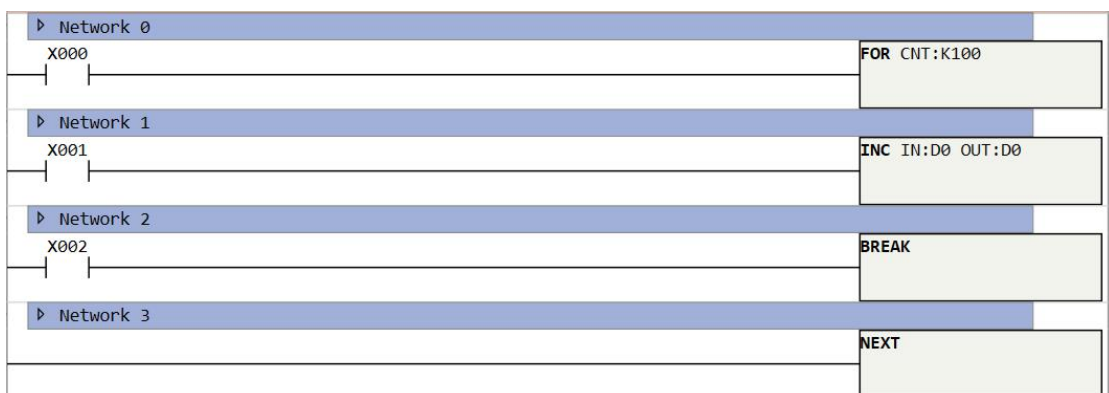


Figure 10.8. 1

2. STL to STLE

Instruction table:

```

Network 000
  STL      S0 //start of step ladder
Network 001
  LD       X000
  BREAK //jump out from step ladder when X000 is ON
Network 002
  LD       X001
  OUT      Y0 //continue this instruction if not jump out
Network 003
  STLE //end of step ladder

```

Ladder diagram:

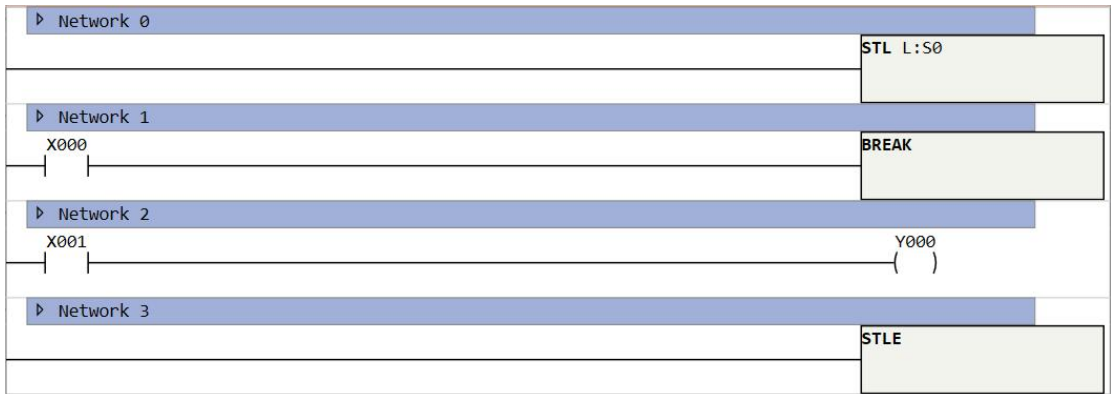


Figure 10.8. 2

IST

Instruction introduction

1. This instruction automatically controls the initial state and special registers in a step ladder program.
2. The 8 continuous bit-registers to control the step ladder program is start with input bit-register (IN). And these bit-registers take effect at their rising-edge, in order their effects are: switch to manual mode (IN), switch to zero-return mode (IN+1), switch to single-step mode (IN+2), switch to single-cycle mode (IN+3), switch to automatic mode (IN+4), zero-return starts (IN+5), start (IN+6), and stop (IN+7).
3. This instruction is related to some special registers (data-registers and bit-registers), details refer to Table 10.9.1. Don't use these registers as usual registers in the program.

Register		Description	Read-write access
D-register	D8048	Record current mode of step ladder program (0 for not enabled, 1 for manual mode, 2 for origin returning mode, 3 for single-step mode, 4 for single-cycle mode, 5 for automatic mode).	R
M-register	M8020	STL transfer disable.	R/W
	M8021	Transfer starts.	R/W
	M8022	Start pulse of IST .	R
	M8023	Zero-return completes.	R
	M8024	Zero-return condition.	R/W
	M8025	All output reset disable.	R/W
	M8026	STL state ON	R
S-register	S0	Initial state of manual mode.	R/W
	S1	Initial state of origin returning mode.	R/W
	S2	Initial state of automatic modes (include single-step mode, single-cycle mode, and automatic mode).	R/W

Table 10.9. 1

Introduction of related registers

The details on special registers and initial state (S0 to S2) which are automatically controlled by the IST instruction is as shown in the equivalent ladder programs below, and the head of control bit-registers is set with M20 (only for reference, cannot be programmed).

a) **D8048**

D8048 is related to current running mode of step ladder program (include manual mode, origin returning mode, single-step mode, single-cycle mode, and automatic mode). Its initial value is 0, and transforms at rising-edge of specified bit-registers. Single-step mode, single-cycle mode, and automatic mode are all under one same

state, so they are all classified into automatic modes, and these 3 modes only transfer when **M8023** is ON (origin returning is completed). Equivalent ladder program refers to [Figure 10.9.1](#).

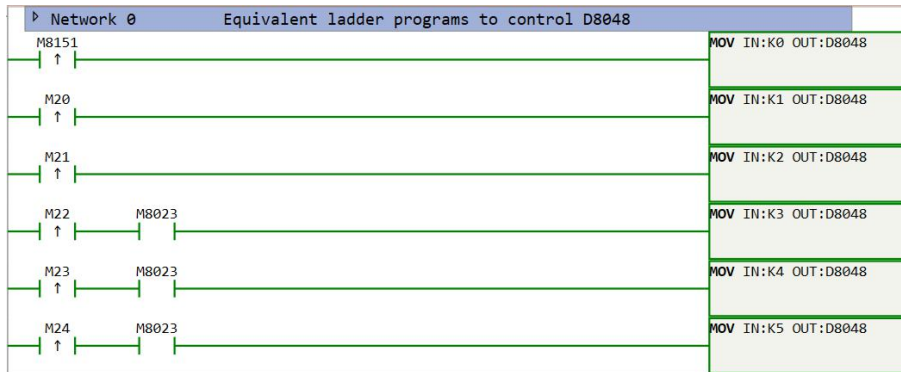


Figure 10.9. 1

b) **M8020**

When **M8020** is ON, transfer of every state is disabled.

c) **M8021**

M8021 turns on only under automatic modes, when step ladder program starts (rising-edge of IN+5), it turns ON; When step ladder program stops (rising-edge of IN+5), it turns OFF; when mode is switched to manual mode or zero-return mode, it turns OFF. Equivalent ladder program refers to [Figure 10.9.2](#).

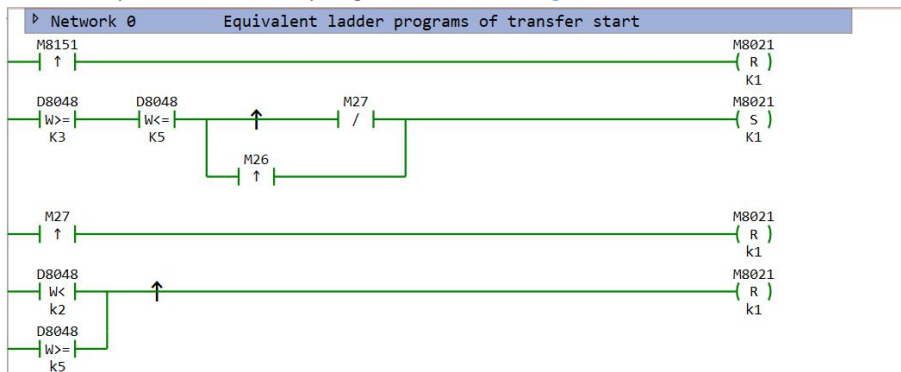


Figure 10.9. 2

d) **M8022**

M8022 outputs a pulse at start operation under automatic mode, or at zero-return start operation under zero-return mode. Equivalent ladder program refers to [Figure 10.9.3](#).

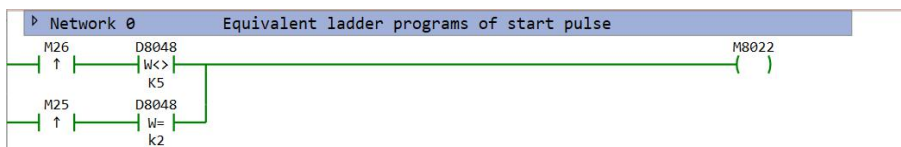


Figure 10.9. 3

e) **M8023**

M8023 only turns ON under manual mode when zero return is complete. Equivalent ladder program refers to [Figure 10.9.4](#).

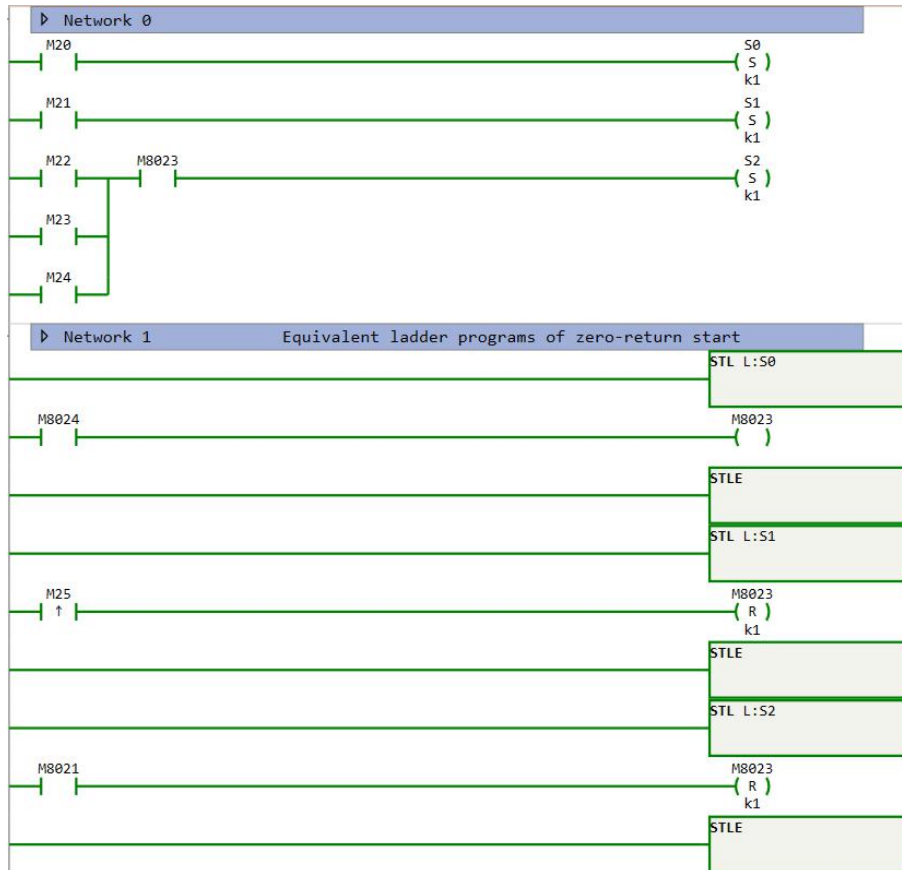


Figure 10.9. 4

- f) **M8024**
M8024 is flag-bit of zero-return, it detects if machine returns to the zero point (need to be assigned by user program), when zero-return is completed, it turns ON, otherwise it remains OFF.
- g) **M8025**
When the mode is switched among manual mode, zero-return mode and automatic mode, all outputs (Y-registers) and operation state relays (S-registers) are reset if the machine is not located at the zero point. If **M8025** has been set to ON in advance, however, only operation state relays are reset.
- h) **M8026**
When either of state relays (S-registers) is ON, **M8026** turns ON. When neither of state relays (S-registers) is ON, **M8026** turns OFF.
- i) **M8027**
When **M8027** is ON, it stores in **D8040** to **D8047** sequentially from small number to big number of the step relay which operates in the step ladder.
- j) State registers (S0, S1, and S2)
S0, S1, and S2 respectively are initial state relays of manual mode, zero-return mode, and automatic mode. When one mode is selected, the corresponding initial start relay will be set, and the others will be reset, meanwhile all the output Y-registers will be immediately reset, user can also set **M8025** to ON to cancel the reset of Y-registers. Equivalent ladder program refers to [Figure 10.9.5](#).

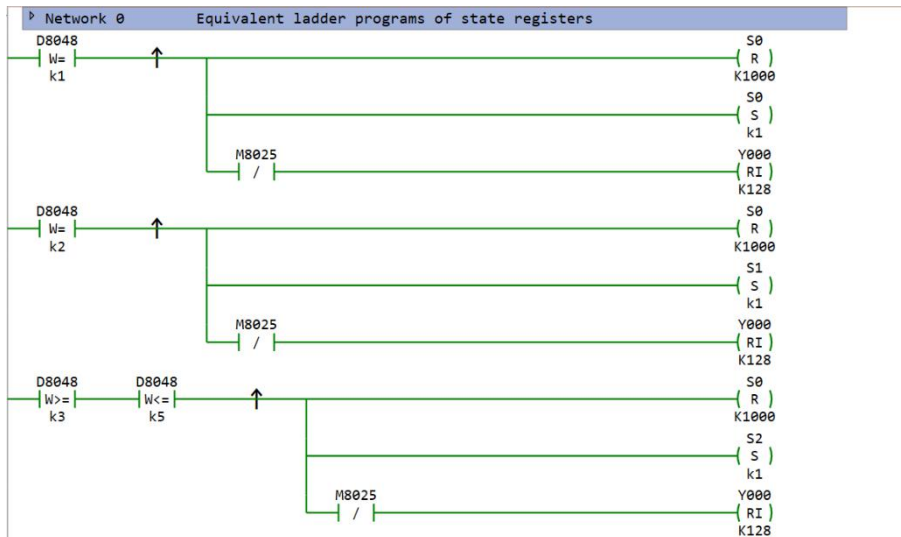


Figure 10.9. 5

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	Head of bit-registers to control the step ladder program	X/M	0/1	bool

Table 10.9. 2

Attention

1. Use this instruction at beginning of the main program, and enable it with **M8151** (close when PLC runs).
2. This instruction can be used only once in the project.

Example

There is a task to separate balls with two kind of weight, the mechanism and working process is designed as [Figure 10.9.6](#) and [Figure 10.9.7](#). The mechanical arm moves up by output Y000, moves down by output Y001, moves right by output Y002, moves left by output Y003; input X000 distinguishes the weight of balls (ON for heavy ones and OFF for light ones), input X001 is the left limit of horizontal moving, X002 is the right limit of heavy balls carry moving, X003 is the right limit of light balls carry moving, X004 is the upper limit of vertical moving, X005 is the lower limit of vertical moving. The working process is:

- ① Move down to lower limit from zero point.
- ② Clamp one ball and judge the weight.
- ③ Move up to upper limit.
- ④ Move right to right limit of heavy/light balls if X000 is ON/OFF.
- ⑤ Move down to lower limit.

- ⑥ Unclamp the ball.
- ⑦ Move up to upper limit.
- ⑧ Move left to left limit.

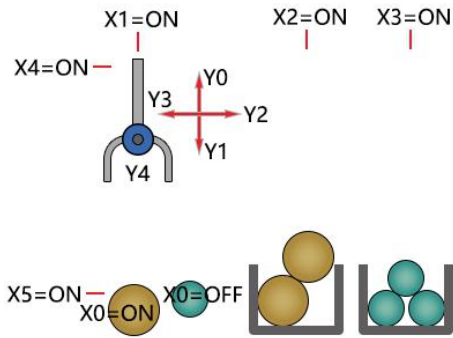


Figure 10.9. 6

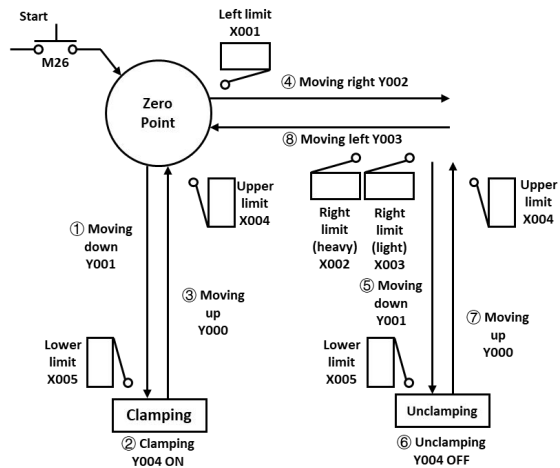


Figure 10.9. 7

The step ladder program shows as below:

Main program sets the zero-return condition **M8024**, enable **IST** instruction, and calls subroutine program of three kinds of modes (manual mode, zero-returning mode, and automatic mode).

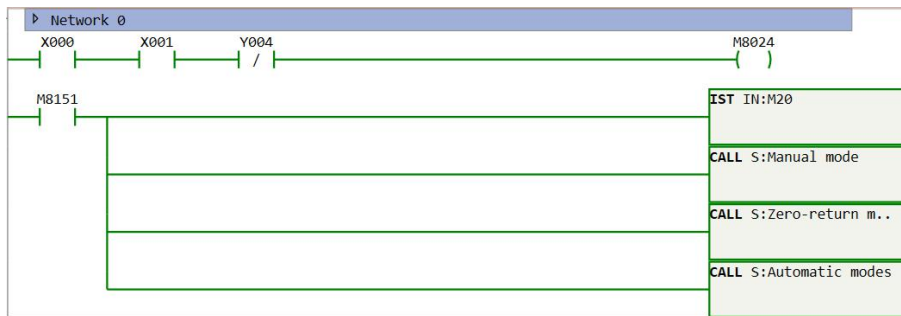


Figure 10.9. 8

Manual mode program provides input to control the output manually. The state number of this mode is S0.

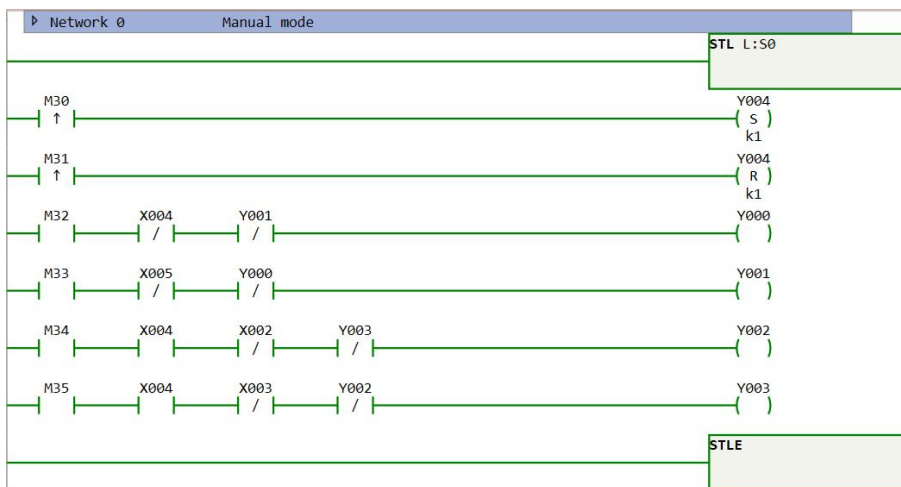


Figure 10.9. 9

Zero-return mode program move the mechanical arm to zero point by set M25. The state number if this mode is S1.

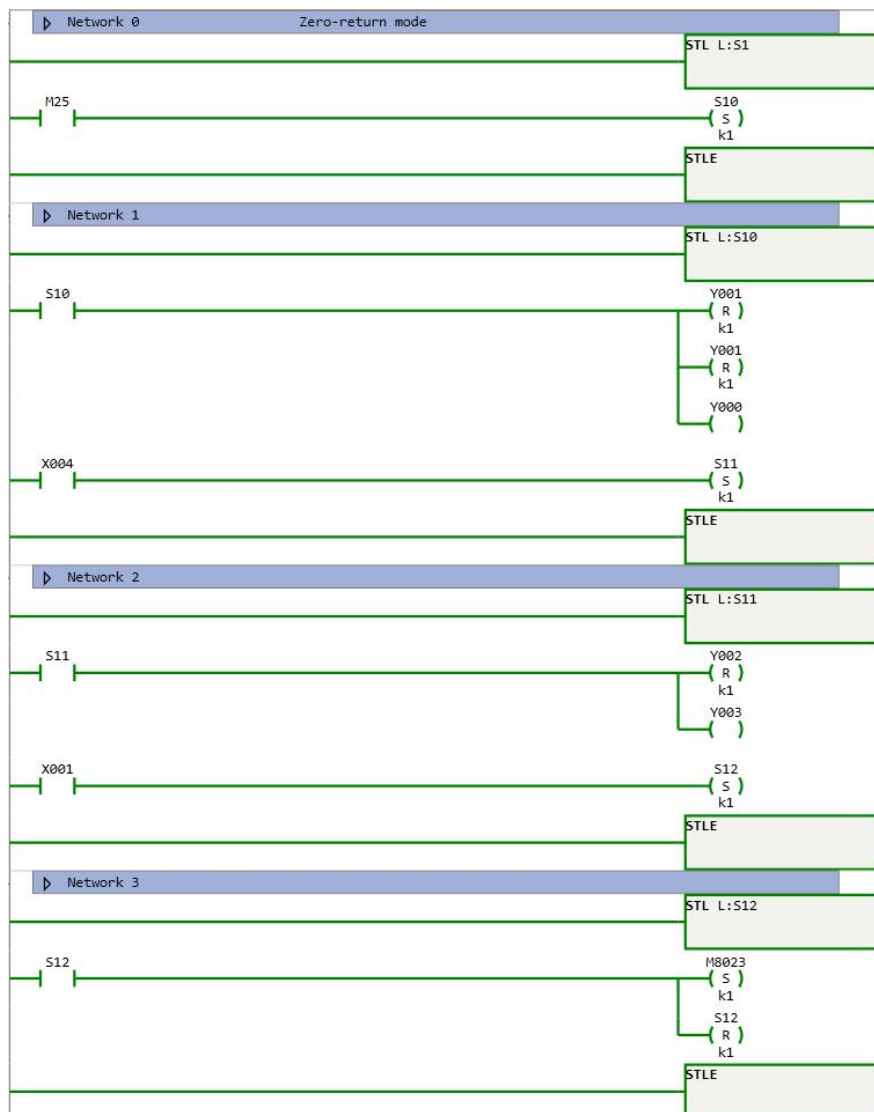


Figure 10.9. 10

Single-step mode, single-cycle mode, and automatic mode share one subroutine program, and their state number is S2.

In single-step mode, process of program runs state by state, and the state won't transfer to next one until there is a rising-edge of M26.

In single-cycle mode, process of program runs a whole cycle, and won't continue next cycle until there is a rising-edge of M26.

In automatic mode, process of program runs automatically.

[Figure 10.9.11](#) shows the start of automatic modes program (S2->S20); [Figure 10.9.12](#) shows branch program of carrying heavy balls (S30->S31->S32); [Figure 10.9.13](#) shows branch program of carrying light balls (S40->S41->S42); [Figure 10.9.14](#) shows program of putting balls and back to initial state(S50->S60->S70->S2).

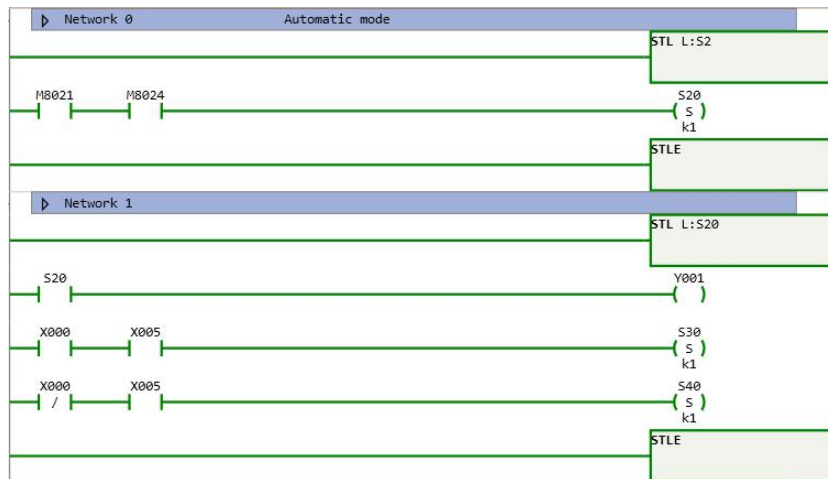


Figure 10.9. 11



Figure 10.9. 12

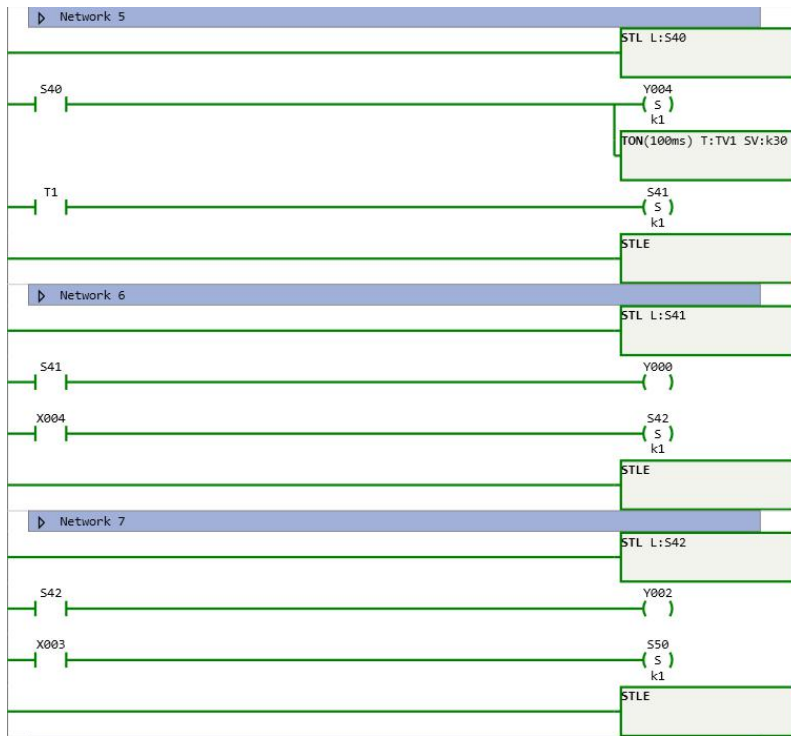


Figure 10.9. 13



Figure 10.9. 14

ISTNEXT

Instruction introduction

1. This instruction takes effects under single-step mode and single-cycle mode of **IST**. When **ISTNEXT** is enabled, state can transfer to next without rising-edge of start operation.
2. **M8020** will be reset when this instruction is enabled so that it can transfer to next state.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool

Table 10.10. 1

Example

Refer to automatic mode part of [example](#) in IST, [Figure 10.10.1](#) shows the case of which **ISTNEXT** is applied.

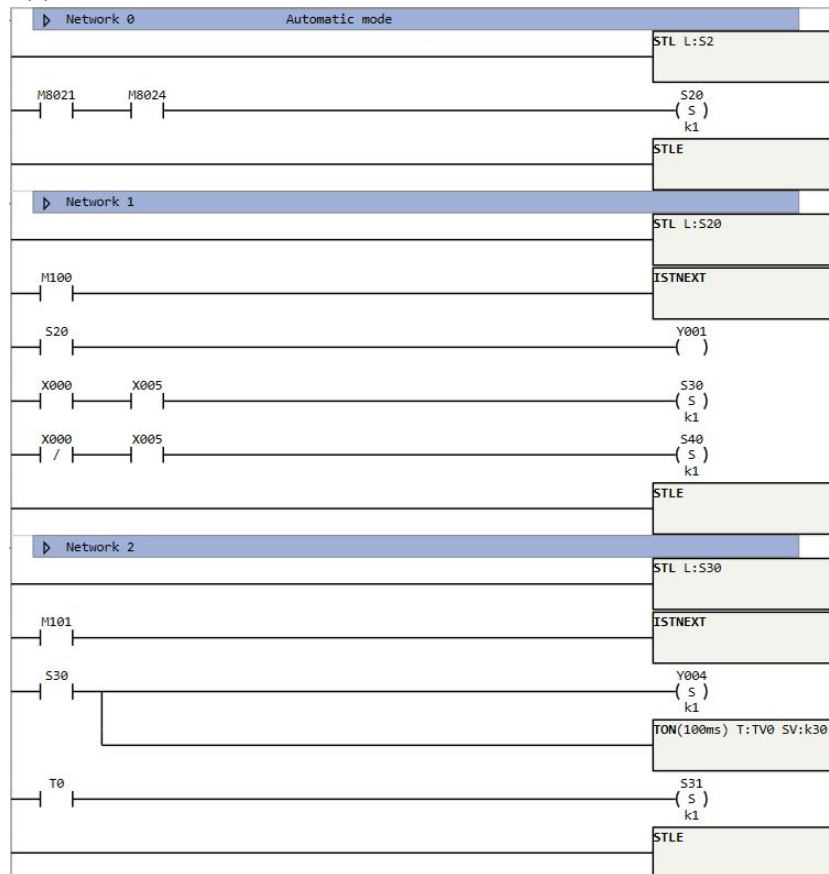


Figure 10.10. 1

WDT

Instruction introduction

1. This instruction resets watchdog timer in a program.
2. User can start using watchdog function and set target value of watchdog timer in setting as shown in [Figure 10.11.1](#). Watchdog will be reset in each scan period automatically, and can also be reset manual by this instruction. If the time of watchdog timer exceeds target time, PLC will stop running and report error code 11 in **D8176**.
3. Timeout of watchdog timer is always caused by endless loop in program.

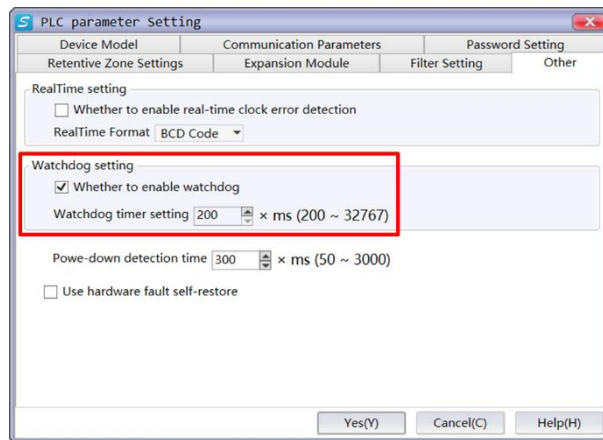


Figure 10.11. 1

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool

Table 10.11. 1

Example

Instruction table:

Network 000

```

LD      M0
WDT    //reset watchdog timer at rising-edge of M0
POP

```

Ladder diagram:



Figure 10.11. 2

Shifting Instructions

Notes

1. Instructions in this chapter do shifting operation on data-registers and continuous bit-registers.
2. There are system special function registers that record status of shifting result:
M8166: When the last bit of shifting operation is 1, this bit-register is ON.
M8167: When the length of shifting operation is 0, this bit-register is ON.

SHL/SHLD

Instruction introduction

1. These instructions shift an input data leftward by (n) bits and output the result to specified register, details refer to [Figure 11.2.1](#). **SHL** is for WORD data, **SHLD** is for DWORD data.
2. This instruction may set **M8166** and **M8167** to ON.



Figure 11.2. 1

Setting data

➤ SHL

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(S)	Source register to shift leftward.	K/H/D/TV/CV/AI/AO/V/Z	-	16-bit code
(N)	Number of bits to shift leftward.	K/H/D	-32768~32767	16-bit signed integer
(D)	Destination register to store the result	D/TV/CV/AO/V/Z	-	16-bit code

Table 11.2. 1

➤ SHLD

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(S)	Source register to shift leftward.	K/H/D/TV/CV/AI/AO/V/Z	-	32-bit code
(N)	Number of bits to shift leftward.	K/H/D	-2147483648~2147483647	32-bit signed integer
(D)	Destination register to store the result	D/TV/CV/AO/V/Z	-	32-bit code

Table 11.2. 2

Example

Instruction table:

Network 000

LD X000

SHL D0 K1 D1 //shift D0 leftward by 1-bit and store result in D1

Ladder diagram:

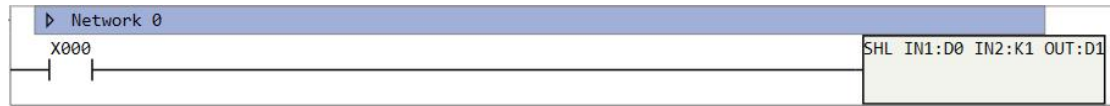


Figure 11.2. 2

SHR/SHRD

Instruction introduction

1. These instructions shift an input data rightward by (n) bits and output the result to specified register, details refer to [Figure 11.3.1](#). **SHR** is for WORD data, **SHRD** is for DWORD data.
2. This instruction may set **M8166** and **M8167** to ON.



Figure 11.3. 1

Setting data

➤ SHR

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(S)	Source register to shift rightward.	K/H/D/TV/CV/AI/AO/V/Z	-	16-bit code
(N)	Number of bits to shift rightward.	K/H/D	-32768~32767	16-bit signed integer
(D)	Destination register to store the result	D/TV/CV/AO/V/Z	-	16-bit code

Table 11.3. 1

➤ SHLD

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(S)	Source register to shift rightward.	K/H/D/TV/CV/AI/AO/V/Z	-	32-bit code
(N)	Number of bits to shift rightward.	K/H/D	-2147483648~2147483647	32-bit signed integer
(D)	Destination register to store the result	D/TV/CV/AO/V/Z	-	32-bit code

Table 11.3. 2

Example

Instruction table:

Network 000

LD X000

SHR D0 K1 D1 //shift D0 leftward by 1-bit and store result in D1

Ladder diagram:

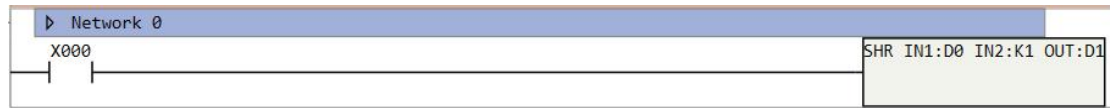


Figure 11.3. 2

ROL/ROLD

Instruction introduction

1. These instructions rotate an input data leftward by (n) bits and output the result to specified register, details refer to [Figure 11.4.1](#). **ROL** is for WORD data, **ROLD** is for DWORD data.
2. This instruction may set **M8166** and **M8167** to ON.

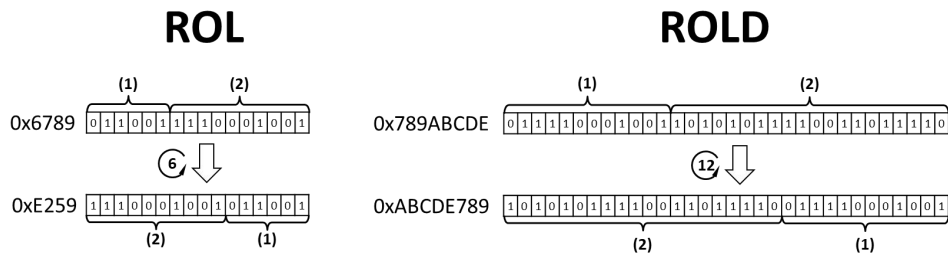


Figure 11.4. 1

Setting data

➤ ROL

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(S)	Source register to rotate leftward.	K/H/D/TV/CV/AI/AO/V/Z	-	16-bit code
(N)	Number of bits to rotate leftward.	K/H/D	-32768~32767	16-bit signed integer
(D)	Destination register to store the result	D/TV/CV/AO/V/Z	-	16-bit code

Table 11.4. 1

➤ ROLD

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(S)	Source register to rotate leftward.	K/H/D/TV/CV/AI/AO/V/Z	-	32-bit code
(N)	Number of bits to rotate leftward.	K/H/D	-2147483648~2147483647	32-bit signed integer
(D)	Destination register to store the result	D/TV/CV/AO/V/Z	-	32-bit code

Table 11.4. 2

Example

Instruction table:

Network 000

LD X000

ROL D0 K1 D1 //rotate D0 leftward by 1-bit and store result in D1

Ladder diagram:

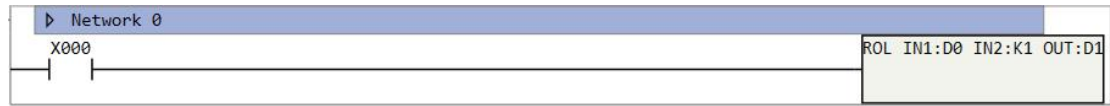


Figure 11.4. 2

ROR/RORD

Instruction introduction

1. These instructions rotate an input data rightward by (n) bits and output the result to specified register, details refer to [Figure 11.5.1](#). **ROR** is for WORD data, **RORD** is for DWORD data.
2. This instruction may set **M8166** and **M8167** to ON.

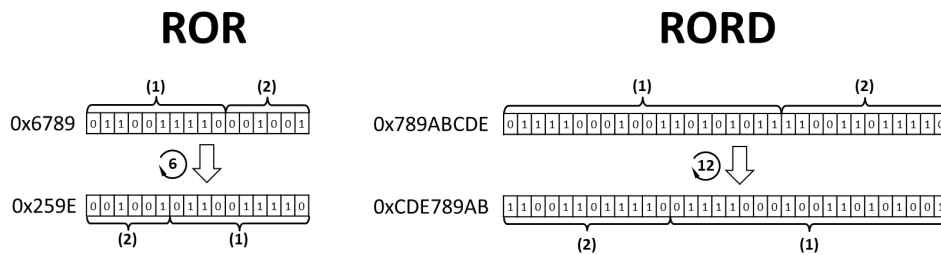


Figure 11.5. 1

Setting data

➤ ROR

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(S)	Source register to rotate rightward.	K/H/D/TV/CV/AI/AO/V/Z	-	16-bit code
(N)	Number of bits to rotate rightward.	K/H/D	-32768~32767	16-bit signed integer
(D)	Destination register to store the result	D/TV/CV/AO/V/Z	-	16-bit code

Table 11.5. 1

➤ RORD

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(S)	Source register to rotate rightward.	K/H/D/TV/CV/AI/AO/V/Z	-	32-bit code
(N)	Number of bits to rotate rightward.	K/H/D	-2147483648~2147483647	32-bit signed integer
(D)	Destination register to store the result	D/TV/CV/AO/V/Z	-	32-bit code

Table 11.5. 2

Example

Instruction table:

Network 000

LD X000

ROR D0 K1 D1 //rotate D0 rightward by 1-bit and store result in D1

Ladder diagram:

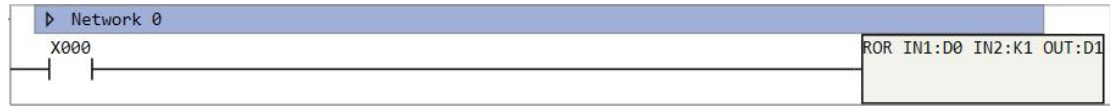


Figure 11.5. 2

SHLB

Instruction introduction

1. This instruction shifts (N1) bit length continuous bit-registers (D) leftward by (N2) bits and fill the empty bits with continuous source bit-registers (S), details refer to [Figure 11.6.1](#).
2. This instruction may set **M8166** and **M8167** to ON.

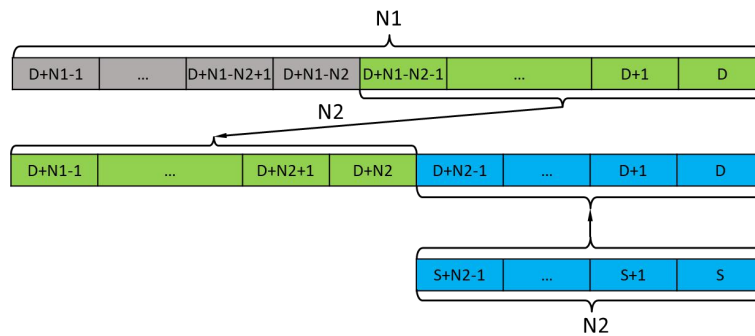


Figure 11.6. 1

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(S)	Head of source bit-registers to fill empty	X/Y/M/S/T/C	-	bool
(D)	Head of destination bit-registers to shift leftward.	Y/M/S/T/C	-	bool
(N1)	Length of source bit-registers.	K/H/D	-32768~32767	16-bit signed integer
(N2)	Length of shifting.	K/H/D	-32768~32767	16-bit signed integer

Table 11.6. 1

Example

Instruction table:

Network 000

LD X000

SHLB M0 Y0 K16 K1 //shift Y0~Y15 1 bit leftward, and fill Y0 with M0

Ladder diagram:



Figure 11.6. 2

SHRB

Instruction introduction

1. This instruction shifts (N1) bit length continuous bit-registers (D) rightward by (N2) bits and fill the empty bits with continuous source bit-registers (S), details refer to [Figure 11.7.1](#).
2. This instruction may set **M8166** and **M8167** to ON.

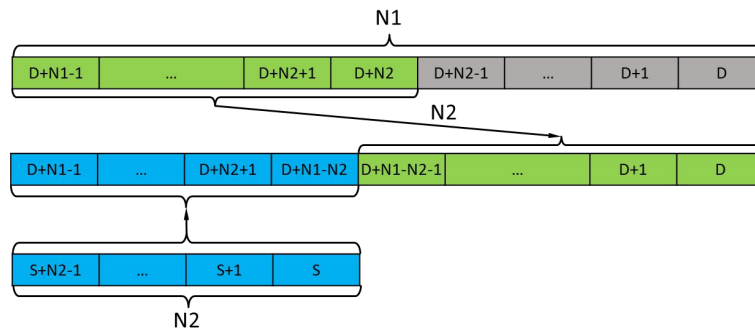


Figure 11.7. 1

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(S)	Head of source bit-registers to fill empty	X/Y/M/S/T/C	-	bool
(D)	Head of destination bit-registers to shift rightward.	Y/M/S/T/C	-	bool
(N1)	Length of source bit-registers.	K/H/D	-32768~32767	16-bit signed integer
(N2)	Length of shifting.	K/H/D	-32768~32767	16-bit signed integer

Table 11.7. 1

Example

Instruction table:

Network 000

LD X000

SHRB M0 Y0 K16 K1 //shift Y0~Y15 1 bit rightward, and fill Y0 with M0

Ladder diagram:



Figure 11.7. 2

Interrupt Instructions

Notes

1. Interrupt is a function for CPU of PLC to handle special cases such as an error, alarm. Interrupt can be triggered by some interrupt factors, user can link these interrupt factors with interrupt program (subroutine program), so that corresponding interrupt program will be executed when interrupt factor occurs.
2. Interrupt has priority, when two interrupts with different priorities are triggered simultaneously, the interrupt with higher priority will be handled firstly. If higher priority interrupt is triggered when lower priority interrupt is being handled, the handling of lower priority interrupt will pause until handling of higher priority interrupt completes.
3. [Table 12.1.1](#) shows the interrupt vector table of external interrupts. edge change interrupts and timer interrupts have lower priority than else. Do not trigger the external interrupt too frequently, otherwise some interrupts may not be handled correctly.

	Interrupt number	Interrupt factor
For all series	K0	rising-edge of X000 input
	K1	falling-edge of X000 input
	K2	edge change of X000 input
	K3	rising-edge of X001 input
	K4	falling-edge of X001 input
	K5	edge change of X001 input
	K6	Interrupt timer 0 reach target time (target time can be set by system special function register D8173 , the range is 1~32767ms, the resolution is 1ms)
	K7	Interrupt timer 1 reach target time (target time can be set by system special function register D8174 , the range is 1~32767ms, the resolution is 1ms)
For FGm series only	K8	Pulse output of Y000 completes
	K9	Pulse output of Y001 completes
	K10	Pulse output of Y002 completes
	K11	Pulse output of Y003 completes
	K12	Pulse output of Y004 completes
	K13	Pulse output of Y005 completes
	K14	Pulse output of Y006 completes
	K15	Pulse output of Y007 completes
	K16	Pulse output of Y010 completes
	K17	Pulse output of Y011 completes
	K18	Counter CV235 counts to target number
	K19	Counter CV236 counts to target number
	K20	Counter CV237 counts to target number
	K21	Counter CV238 counts to target number
	K22	Counter CV239 counts to target number
	K23	Counter CV240 counts to target number

K24	Counter CV241 counts to target number
K25	Counter CV242 counts to target number
K26	Counter CV243 counts to target number
K27	Counter CV244 counts to target number
K28	Counter CV245 counts to target number
K29	Counter CV246 counts to target number
K30	rising-edge of X002 input
K31	falling-edge of X002 input
K32	edge change of X002 input
K33	rising-edge of X003 input
K34	falling-edge of X003 input
K35	edge change of X003 input
K36	rising-edge of X004 input
K37	falling-edge of X004 input
K38	edge change of X004 input
K39	rising-edge of X005 input
K40	falling-edge of X005 input
K41	edge change of X005 input

Table 12.1. 1

ATCH

Instruction introduction

This instruction attaches interrupt factor with subroutine program. Corresponding subroutine program will execute when the interrupt factor occurs if interrupt is enabled.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(EV)	Subroutine program to execute	-	-	32-bit pointer
(INT)	Number of interrupt factor	K	0~41	16-bit unsigned integer

Table 12.2. 1

Example

Instruction table:

Network 000

```
LDP    M0
ATCH  INTR_0      K0 //attach interrupt factor K0 with subroutine program INTR_0
EI //enable the interrupt
```

Network 001

```
LDP    M1
DTCH  K0 //detach interrupt factor K0 from its attached interrupt program
```

Network 002

```
LDP    M2
DI //disable the interrupt
```

Ladder diagram:

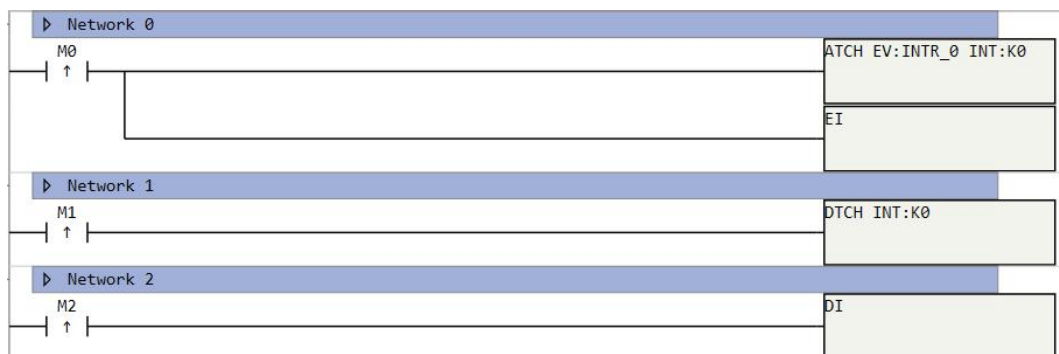


Figure 12.2. 1

DTCH

Instruction introduction

1. This instruction detaches interrupt factor from its attached interrupt program.
2. When **DTCH** is executed, counting of interrupt timer 0 and interrupt timer 1 will be reset, corresponding target number in **D8713** and **D8174** will also be reset.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(INT)	Number of interrupt factor to detach	K	0~41	16-bit unsigned integer

Table 12.3. 1

Example

Refer to [example](#) of **ATCH**.

EI

Instruction introduction

1. This instruction enables the execution of the interrupt program, interrupt program won't be executed unless **EI** is enabled.
2. When PLC enter RUN state, the interrupt is disabled by default.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool

Table 12.4. 1

Example

Refer to [example](#) of **ATCH**.

DI

Instruction introduction

This instruction disables the execution of the interrupt program until the **EI** instruction is enabled.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool

Table 12.5. 1

Example

Refer to [example](#) of **ATCH**.

Real-time Clock Instructions

Notes

1. Instructions in this chapter do read, write, and compare operations on clock data, and the system clock of PLC is called RTC (real-time clock).
2. RTC data of PLC consists of 8 parts: year, month, day, hour, minute, second, reserve part, and week. The RTC data of PLC is stored in BCD format.
3. User can set format of clock data in “other option” of PLC settings, refer to [Figure 13.1.1](#). Format of clock data can be selected with BCD code or BIN code (BIN and BCD refer to [notes](#) of conversion instructions). RTC data of PLC itself is formatted with BCD, and the setting doesn't influence it.

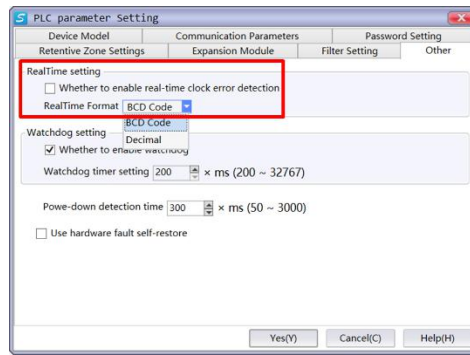


Figure 13.1. 1

4. [Table 13.1.1](#) shows the details of read clock data. (For data register that stores the week, 0 is Sunday, 1 is Monday, 2 is Tuesday, 3 is Wednesday, 4 is Thursday, 5 is Friday, 6 is Saturday)

Number of data-register	Description	Range	
		BCD	BIN
0	Year	0x2000~0x2099	2000~2099
1	Month	0x0001~0x0012	1~12
2	Day	0x0001~0x0031	1~31
3	Hour	0x0000~0x0023	0~23
4	Minute	0x0000~0x0059	0~59
5	Second	0x0000~0x0059	0~59
6	Reserved	-	-
7	Week	0x0000~0x0006	0~6

Table 13.1. 1

TRD

Instruction introduction

This instruction reads RTC data of PLC and store it in specified 8 continuous D-registers (refer to [notes](#) of this chapter) when it is enabled.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(T)	Head of D registers to restore RTC data.	D	-	-

Table 13.2. 1

Example

Instruction table:

Network 000

LD X000

TRD D0 //read RTC data of PLC and store in D0-D7

Ladder diagram:



Figure 13.2. 1

TWR

Instruction introduction

1. This instruction writes RTC data into PLC from specified 8 continuous D-registers (refer to [notes](#) of this chapter) when it is enabled.
2. This instruction should be triggered by edge signal.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(T)	Head of D registers store RTC data.	D	-	-

Table 13.3. 1

Example

Instruction table:

Network 000

LD X000

TWR D0 //write RTC data into PLC from D0~D7

Ladder diagram:



Figure 13.3. 1

TRDS

Instruction introduction

1. This instruction reads RTC data of PLC, calculates how many seconds pass since 2000/1/1 00:00:00, and stores the second number in specified D-register (double-word).
2. This instruction should be triggered by edge signal.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(T)	D register to store the number of seconds.	D	0~ 4294967295	32-bit unsigned integer

Table 13.4. 1

Example

Instruction table:

Network 000

```
LD      X000
TRDS   D0 //write passed seconds into D0D1
```

Ladder diagram:

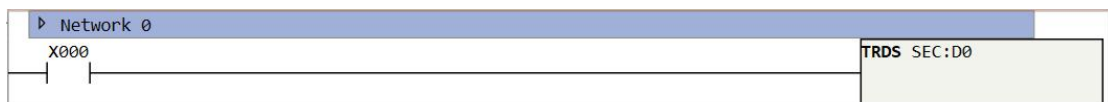


Figure 13.4. 1

TWRS

Instruction introduction

1. According to number of seconds pass since 2000/1/1 00:00:00 stored in specified D-register, this instruction calculates the clock data and writes it into PLC.
2. This instruction should be triggered by edge signal.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(T)	D register stores the number of seconds	D	0~ 4294967295	32-bit unsigned integer

Table 13.5. 1

Example

Instruction table:

Network 000

```
LD      X000
TDRS   D0 //write passed seconds into D0D1
```

Ladder diagram:

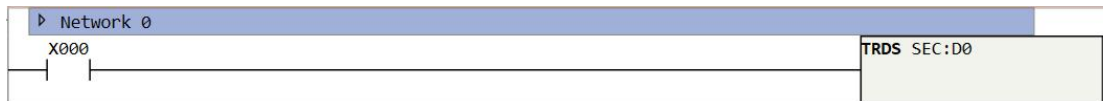


Figure 13.5. 1

TSEC

Instruction introduction

This instruction reads clock data in specified 8 continuous D-registers, calculates how many seconds pass since 2000/1/1 00:00:00, and stores the number of seconds in specified D-register (double-word).

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(T)	Head of D registers store the clock data.	D	-	-
(SEC)	D register to store the number of seconds	D	0~ 4294967295	32-bit unsigned integer

Table 13.6. 1

Example

Instruction table:

Network 000

LD X000

TSEC D0

D10 //transform data in D0~D7 into number of seconds and store in D10

Ladder diagram:

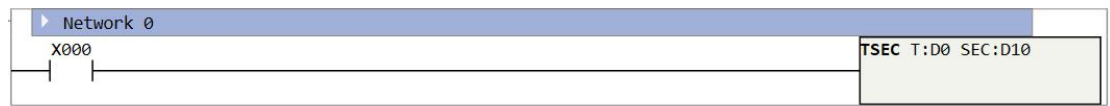


Figure 13.6. 1

RSEC

Instruction introduction

This instruction reads number of seconds in specified D-register, calculates clock data that pass the specified seconds after 2000/1/1 00:00:00, and stores the clock data in specified 8 continuous D-registers.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(SEC)	D register stores the number of seconds	D	0~ 4294967295	32-bit unsigned integer
(T)	Head of D registers to store the clock data.	D	-	-

Table 13.7. 1

Example

Instruction table:

Network 000

LD X000

RSEC D10 D0 //transform seconds in D10 into clock data and store in D0~D7

Ladder diagram:

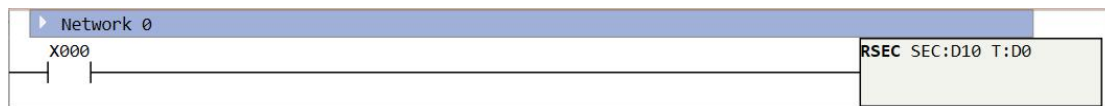


Figure 13.7. 1

TCMP

Instruction introduction

This instruction reads RTC data of PLC, compares it with clock data stored in specified 8 continuous D-registers, and output result to specified bit-registers. If input time is earlier than RTC of PLC, first of output bit-registers is ON; If input time is same to RTC of PLC, second of output bit-registers is ON; If input time is later than RTC of PLC, third of output bit-registers is ON.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(T)	Head of D registers store the clock data.	D	-	-
(OUT)	Head of bit-registers output the comparison result.	Y/M/S	0/1	bool

Table 13.8. 1

Example

Instruction table:

Network 000

LD X000

TCMP D0 Y0 //compare time in D0~D7 with RTC of PLC and output result to Y0~Y2

Ladder diagram:

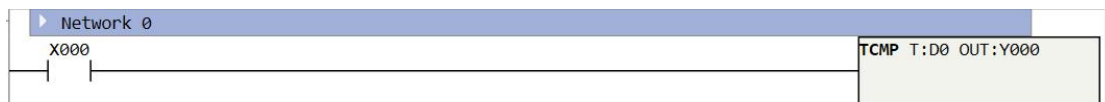


Figure 13.8. 1

TZCP

Instruction introduction

1. This instruction reads RTC data of PLC, compares it with time frame consists of start clock and end clock, and output result to specified bit-registers. If RTC of PLC is before the time frame, first of output bit-registers is ON; If RTC of PLC is within the time frame, second of output bit-registers is ON; If RTC of PLC is after the time frame, third of output bit-registers is ON.
2. End clock must be later than start clock. For example, time frame like [2020/1/1 12:00:00 Wednesday, 2019/1/1 12:00:00 Tuesday] is illegal.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(S)	Head of D registers store the start clock of time frame.	D	-	-
(E)	Head of D registers store the end clock of time frame.	D	-	-
(OUT)	Head of bit-registers output the comparison result.	Y/M/S	0/1	bool

Table 13.9. 1

Example

Instruction table:

Network 000

```

LD      X000
TCMP   D0      D10      Y0 //compare RTC with time frame in D0~D7 and D10~D17, output
                                     result to Y0~Y2
  
```

Ladder diagram:

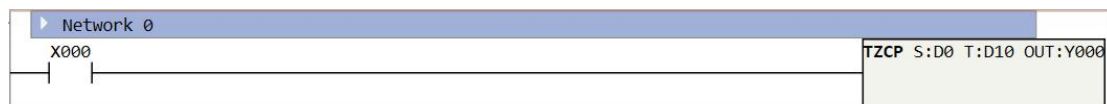


Figure 13.9. 1

WKCMP

Instruction introduction

1. This instruction compares RTC with specified clock data regardless of year, month, and day.
2. This instruction reads RTC data of PLC (without year, month, and day), compares it with time and week data stored in specified 5 continuous D-registers (last 5 words of the whole clock data), and output result to specified bit-registers. If input time and week is earlier than RTC of PLC, first of output bit-registers is ON; If input time and week is same to RTC of PLC, second of output bit-registers is ON; If input time and week is later than RTC of PLC, third of output bit-registers is ON.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(T)	Head of D registers store the time and week data.	D	-	-
(OUT)	Head of bit-registers output the comparison result.	Y/M/S	0/1	bool

Table 13.10. 1

Example

Instruction table:

Network 000

LD X000

WKCMP D0 Y0 //compare time and week in D0~D4 with RTC of PLC and output result to Y0~Y2

Ladder diagram:

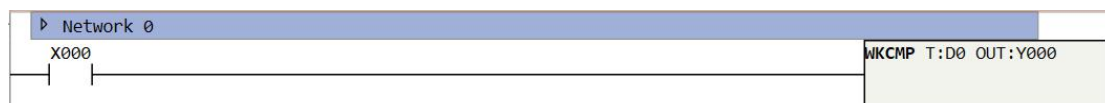


Figure 13.10. 1

WKZCP

Instruction introduction

1. This instruction compares RTC with specified time frame regardless of year, month, and day.
2. This instruction reads RTC data of PLC (without year, month, and day), compares it with time frame consist of start clock (without year, month, and day) and end clock (without year, month, and day), and output result to specified bit-registers. If RTC of PLC is before the time frame, first of output bit-registers is ON; If RTC of PLC is within the time frame, second of output bit-registers is ON; If RTC of PLC is after the time frame, third of output bit-registers is ON.
3. The input start clock and end clock (without year, month, and day) are both stored in 5 continuous D-registers, user can also read from the fourth register of clock data registers.
4. End clock of time frame must be later than start clock of time frame. For example, time frame like [23:59:59 Saturday, 00:00:00 Sunday] is illegal.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(S)	Head of D registers store start clock (without year, month, and day) of time frame.	D	-	-
(E)	Head of D registers store end clock (without year, month, and day) of time frame.	D	-	-
(OUT)	Head of bit-registers output the comparison result.	Y/M/S	0/1	bool

Table 13.11. 1

Example

Instruction table:

Network 000

```
LD      X000
WKZCP  D0      D10      Y0 // compare RTC with time frame in D0~D4 and D10~D14 (without year,
                                month, and day), output result to Y0~Y2
```

Ladder diagram:

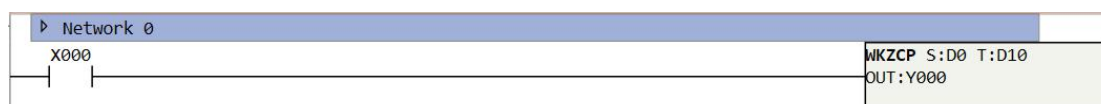


Figure 13.11. 1

CKCMP

Instruction introduction

1. This instruction compares RTC with specified time data only include hour, minute, and second.
2. This instruction reads RTC data of PLC, compares it with time data stored in specified 3 continuous D-registers (the fourth to sixth words of the whole clock data), and output result to specified bit-registers. If input time is earlier than RTC of PLC, first of output bit-registers is ON; If input time is same to RTC of PLC, second of output bit-registers is ON; If input time is later than RTC of PLC, third of output bit-registers is ON.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(T)	Head of D registers store the time data.	D	-	-
(OUT)	Head of bit-registers output the comparison result.	Y/M/S	0/1	bool

Table 13.12. 1

Example

Instruction table:

Network 000

LD X000

CKCMP D0 Y0 //compare time in D0~D2 with RTC of PLC and output result to Y0~Y2

Ladder diagram:

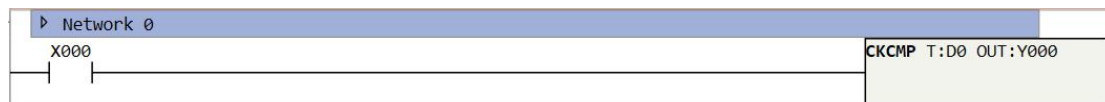


Figure 13.12. 1

CKZCP

Instruction introduction

1. This instruction compares RTC with specified time frame only include hour, minute, and second.
2. This instruction reads RTC data of PLC (only include hour, minute, and second), compares it with time frame consist of start time (only include hour, minute, and second) and end time (only include hour, minute, and second), and output result to specified bit-registers. If RTC of PLC is before the time frame, first of output bit-registers is ON; If RTC of PLC is within the time frame, second of output bit-registers is ON; If RTC of PLC is after the time frame, third of output bit-registers is ON.
3. The input start time and end time are both stored in 3 continuous D-registers, user can also read from the fourth to sixth registers of clock data registers.
4. End time of time frame must be later than start time of time frame. For example, time frame like [23:59:59, 00:00:00] is illegal.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(S)	Head of D registers store start time of time frame.	D	-	-
(E)	Head of D registers store end time of time frame.	D	-	-
(OUT)	Head of bit-registers output the comparison result.	Y/M/S	0/1	bool

Table 13.13. 1

Example

Instruction table:

Network 000

LD X000

CKZCP D0 D10 Y0 // compare RTC with time frame in D0~D2 and D10~D12 (only include hour, minute, and second), output result to Y0~Y2

Ladder diagram:

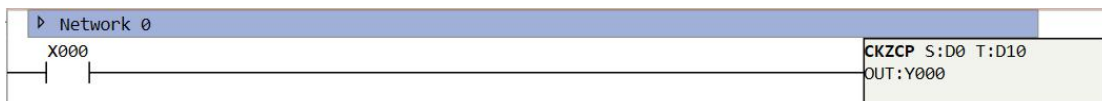


Figure 13.13. 1

Communication Instructions

Notes

Samkoon PLC supports Modbus communication protocol that can be applied for serial ports like RS232, RS485, and USB, the Modbus support at most 127 slave stations. User can set communication parameters in PLC parameter setting as shown in [Figure 14.1.1](#).

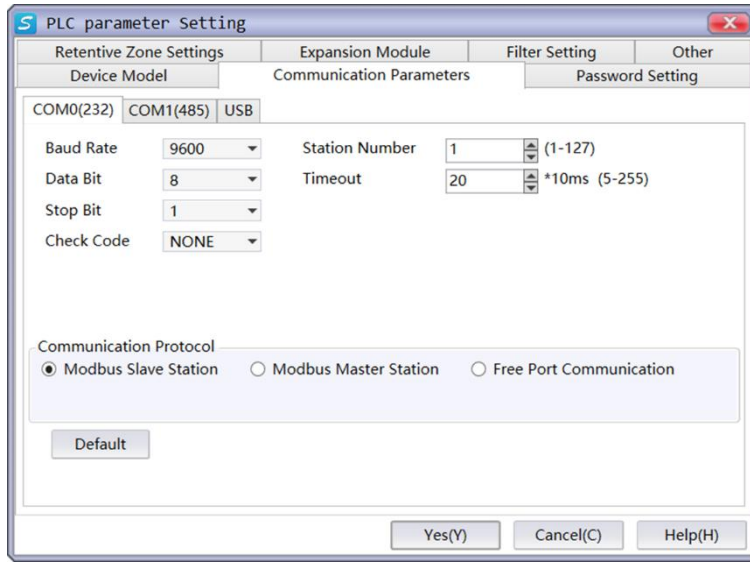


Figure 14.1. 1

When using Modbus communication, firstly user need to create Modbus table before use communication instructions. Shown as [Figure 14.1.2](#) and [Figure 14.1.3](#), user can enter Modbus table form project explorer, and edit the table with operation bar.

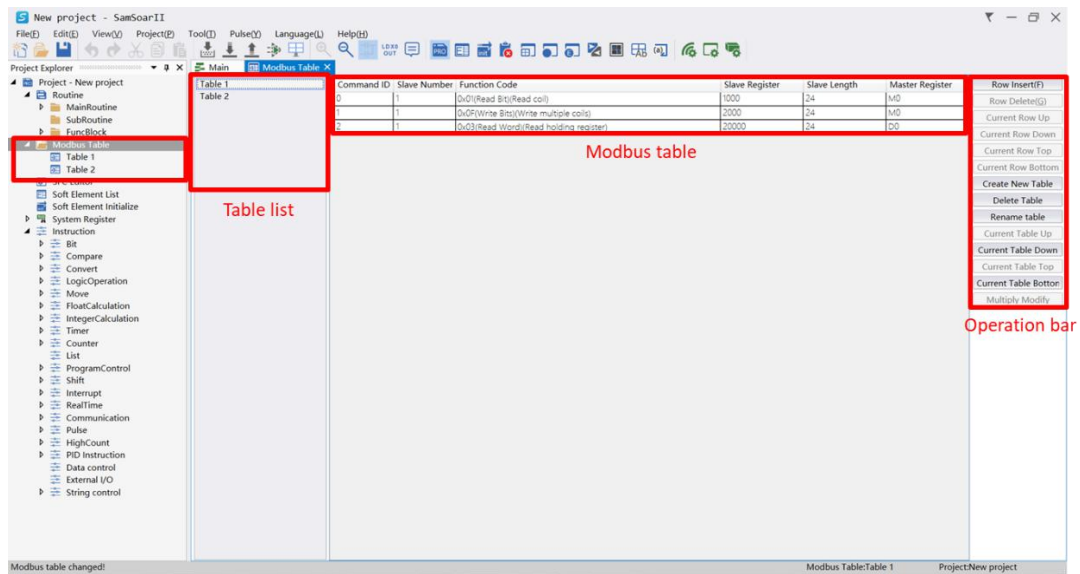


Figure 14.1. 2

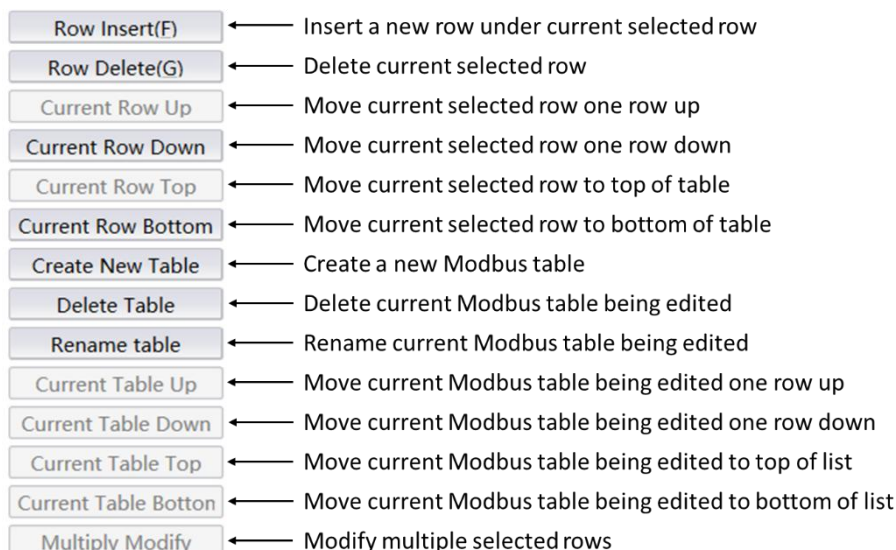


Figure 14.1. 3

Modbus table support function codes 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x0f and 0x10, their function and support register types refer to [Table 14.1.1](#). By setting slave register, slave length, and master register in Modbus table, data of specified length can be read from slave registers and store in master register, or be copied from master registers and write into slave registers. The address of slave register is marked with offset number, [Table 14.1.2](#) shows the offset number of different type of registers.

Function code	Function	Support register types
0x01	Read coil	X/Y/M/S/T/C
0x02	Read disperse input	X
0x03	Read holding register	D/TV/CV/CV32/AI/AO
0x04	Read input register	-
0x05	Write single coil	Y/M/S/T/C
0x06	Write single register	D/TV/CV/CV32/AI/AO
0x0f	Write multiple coils	Y/M/S/T/C
0x10	Write multiple registers	D/TV/CV/CV32/AI/AO

Table 14.1. 1

Register type	Range	Data type	Read function code	Write function code	offset number
X	0~127	bit	0x0x/0x02	0x05	0
EX	0~511	bit	0x01/0x02	0x05	512
Y	0~127	bit	0x01	0x05/0x0f	10000
EY	0~511	bit	0x01	0x05/0x0f	10512
M	0~8191	bit	0x01	0x05/0x0f	30000
S	0~999	bit	0x01	0x05/0x0f	50000
T	0~255	bit	0x01	0x05/0x0f	60768
C	0~255	bit	0x01	0x05/0x0f	60512
D	0~8191	word	0x03/0x04	0x06/0x10	40000
CV	0~199	word	0x03	0x06/0x10	60000
TV	0~255	word	0x03	0x06/0x10	60256
CV32	200~255	dword	0x03	0x06/0x10	61024
AI	0~31	word	0x03	0x06/0x10	20000
EAI	0~79	word	0x03	0x06/0x10	20100
AO	0~31	word	0x03	0x06/0x10	20512
EAO	0~79	word	0x03	0x06/0x10	20612

Table 14.1. 2

EX, EY, EAI, EAO are registers of extension module. For CV32 type register, each 2 continuous offset numbers make a 32-bit data, for example: offset number 61024 is low word of CV200, offset number 61025 is high word of CV200.

MBUS

Instruction introduction

1. This instruction is used for master station to communicate with slave station by using Modbus Table (refer to [notes](#)).
2. Firstly, user need to set communication parameters of serial port before using this instruction as shown in [Figure 14.2.1](#).
 - Station number: number of master station or slave station, the devices in same communication network must be set with different station number.
 - Baud rate: frequency of data signal, the unit is baud (length of a single character, always be 1 byte) per second, can be set with 1200, 2400, 4800, 9600, 19200, 38400, 57600, or 115200. The baud rate of master and slave must be same.
 - Data bit: length of data signal of each data frame.
 - Stop bit: length of stop signal, marks the end of one data frame.
 - Check bit: bit that locates after data bit, can be set with none, even, or odds. If none, there is no check bit; If even, check bit is 1 if there are even number of high-level signals in data bit; If odds, check bit is 1 if there are odds number of high-level signals in data bit.
 - Timeout and retransmission times: if slave doesn't return message for the set timeout, master will resend data, reset timeout, and wait for return message from slave. If reach the set retransmission times, master will report error code 0x10 in specified data-register.
 - Modbus interval: time interval between each data frame.
3. This instruction assigns a D register to store communication information that record status and error of Modbus communication. [Table 14.2.1](#) shows the information code and corresponding meaning. This instruction also assigns a D register to store the current command ID of Modbus table, it will increase or loop when corresponding command is executed, user can also write it to control which command to execute.

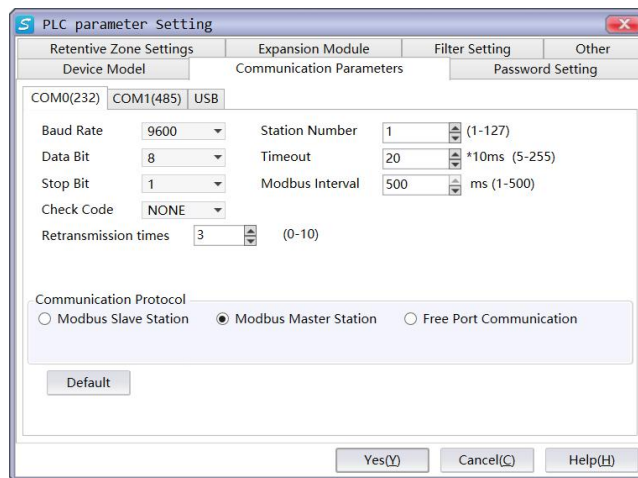


Figure 14.2. 1

Code	Description
0x01	Illegal function code
0x02	Illegal address
0x03	Illegal data
0x04	Slave operation failed
0x05	Command processing
0x06	Slave station is busy
0x07	Illegal data type
0x08	Illegal command ID
0x09	Illegal password
0x10	Communication timeout
0x12	Master/Slave station setting error
0x13	Slave number is same as master number
0x14	Address of register exceeds limit
0x15	Command execution failed
0x18	Received information frame error (length error, CRC check error)
0x20	Parameter is unmodifiable
0x21	Parameter is unmodifiable when running
0x22	Parameter is protected by password

Table 14.2. 1

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(COM)	Selected serial port to communicate.	-	-	-
(TBL)	Selected Modbus table.	-	-	-
(WR)	Data-register to store communication information code.	D		binary code
(WR_ID)	Data-register to store the current command ID of Modbus table.	D		16-bit unsigned integer

Table 14.2. 2

Attention

1. Setting of timeout should be adjusted by baud rate, if the baud rate is low, the slow data transmission may need longer timeout. Additionally, if the data size to transmit is big, transmission may also need longer timeout.
2. When a serial is used for Modbus communication, it can't be used for other purpose such as download/upload, monitor, and free port communication.
3. If one slave of Modbus communication is disconnected, master will still call on it, and this will cause lag on communication. For quicker response, user can set shorter timeout and less retransmission times.

Example

This example uses 3 Modbus tables, and a single table is for a single slave station. Communication will loop among 3 slave stations, and user can also select which slave station to communicate by write slave number in D1234.

Modbus table:

Slave 1	Command ID	Slave Numb	Function Code	Slave Register	Slave Length	Master Register
Slave 2	0	1	0x03(Read Word)(Read holding register)	60200	1	D0
Slave 3	1	1	0x01(Read Bit)(Read coil)	30000	10	M0
	2	1	0x03(Read Word)(Read holding register)	40000	10	D100
Slave 1	Command ID	Slave Numb	Function Code	Slave Register	Slave Length	Master Register
Slave 2	0	2	0x03(Read Word)(Read holding register)	60200	1	D200
Slave 3	1	2	0x01(Read Bit)(Read coil)	30000	10	M200
	2	2	0x03(Read Word)(Read holding register)	40000	10	D300
Slave 1	Command ID	Slave Numb	Function Code	Slave Register	Slave Length	Master Register
Slave 2	0	3	0x03(Read Word)(Read holding register)	60200	1	D400
Slave 3	1	3	0x01(Read Bit)(Read coil)	30000	10	M200
	2	3	0x03(Read Word)(Read holding register)	40000	10	D500

Figure 14.2. 2

Instruction table:

Network 000

```

LDP      M8183
MPS
AW=     D1234      K1
AW=     D5001     K2
MOV     K2        D1234 //go to slave 2 when communication of slave 1 is completed
MOV     K0        D5003 //initiate command ID of slave 2
MRD
AW=     D1234     K2
AW=     D5003     K2
MOV     K3        D1234 //go to slave 3 when communication of slave 2 is completed
MOV     K0        D5005 //initiate command ID of slave 3
MRD
AW=     D1234     K3
AW=     D5005     K2
MOV     K3        D1234 //go to slave 1 when communication of slave 3 is completed
MOV     K0        D5001 //initiate command ID of slave 1
MPP
RST     M8183     K1
POP
LD      M6000
MPS
AW=     D1234     K1
MBUS    K1        Slave 1   D5000   D5001 //enable Modbus communication if slave 1
MRD
AW=     D1234     K1
MBUS    K1        Slave 2   D5002   D5003 //enable Modbus communication if slave 2
MRD
AW=     D1234     K1
MBUS    K1        Slave 3   D5004   D5005 //enable Modbus communication if slave 3
POP
  
```

Ladder diagram:

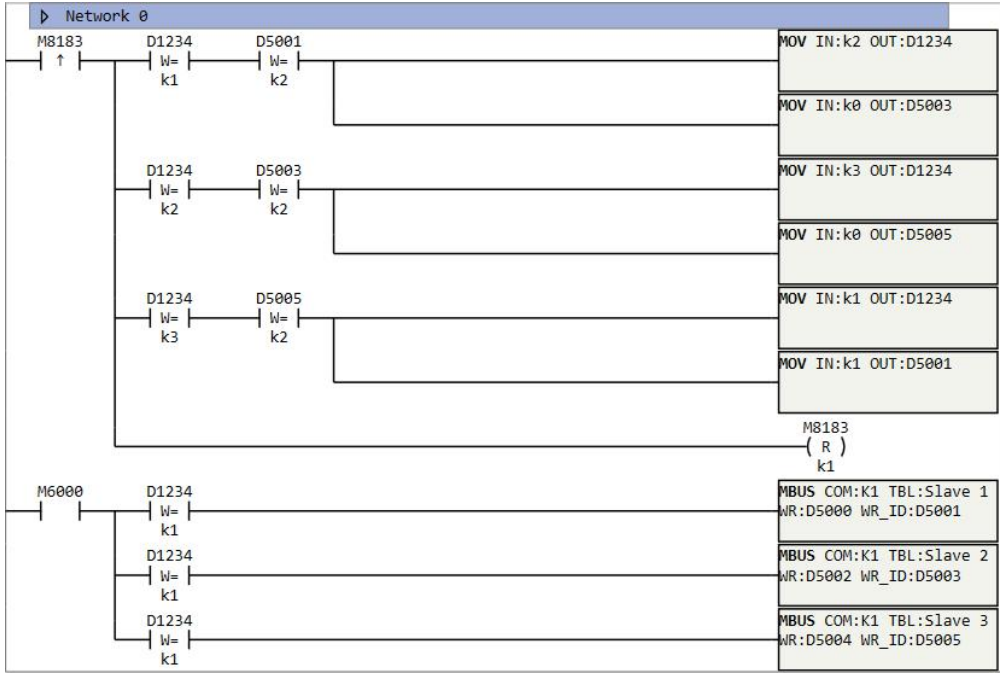


Figure 14.2. 3

SEND

Instruction introduction

1. This instruction is applied to free port communication, it transmits data in specified D-registers (ADDR) with specified length (LEN) to specified serial port (COM).
2. There are system special function registers that record transmission status of corresponding serial ports, refer to [Table 14.3.1](#).
3. There is an extra parameter differs from Modbus communication: Buffer bit. When buffer bit is selected with 8, transmission only reads low byte of D-registers; When buffer is selected with 16, transmission reads and writes high byte and low byte of D-registers. User can set this parameter in communication parameters setting.

Register	Description
M8176	Turns ON when serial port COM0 is transmitting.
M8177	Turns ON when serial port COM1 is transmitting.
M8180	Turns ON when transmission of serial port COM0 is completed.
M8181	Turns ON when transmission of serial port COM1 is completed.

Table 14.3. 1

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(COM)	Selected serial port to send data.	-	-	-
(ADDR)	Head of data-registers to read	D	-	-
(LEN)	Data length to read and send	K/H/D	0~512	16-bit unsigned integer

Table 14.3. 2

Attention

1. User need to select communication protocol as free port communication in communication parameters setting.
2. When a serial is used for free port communication, it can't be used for other purpose such as download/upload, monitor, and Modbus communication.
3. Modification of communication parameters takes effects after downloads again.
4. Free port communication is half-duplex, so that the **SEND** instruction and [REV](#) instruction can nor work currently.

Example

1. Example 1

Instruction table:

Network 000

LDP	M0			
SEND	K1	D0	K21	//select COM1, head register is D0, data length is 21; //if the buffer is 8-bit, all low byte in D0~D20, 21-byte in whole, will be sent //if the buffer is 16-bit, all data in D0~D10 and low byte of D11, 21-byte in whole, will be sent //SEND is trigger by rising-edge here, if it is enabled by normal close coil, SEND will execute during each scan period

Ladder diagram:

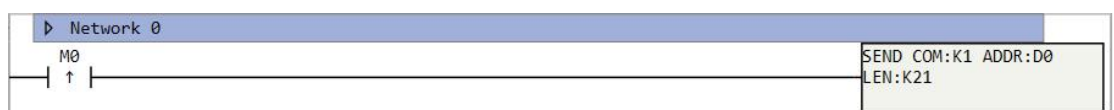


Figure 14.3. 1

2. Example 2

Instruction table:

Network 000

LD	M0			
MPS				
MVBLK	D2000	D1	D0	//move data block of D0 length from D2000 to D1
MPP				
ANDI	M8180			//disable SEND when transmission is completed
MEP				
SEND	K1	D1	D0	//send date to COM0 when M0 is ON

Network 001

LD	M8180			
REV	K0	D1001	D1000	//enable REV after transmission is completed

Network 001

LDF	M8178			//close under a falling-edge when reception is completed
MOV	D1000	D3000		//move data length from D1000 to D3000
MVBLK	D1001	D3001	D1000	//move data block of D1000 length from D1001 to D3001
RST	M8180	K1		//reset M8180 when transmission is completed

Ladder diagram:

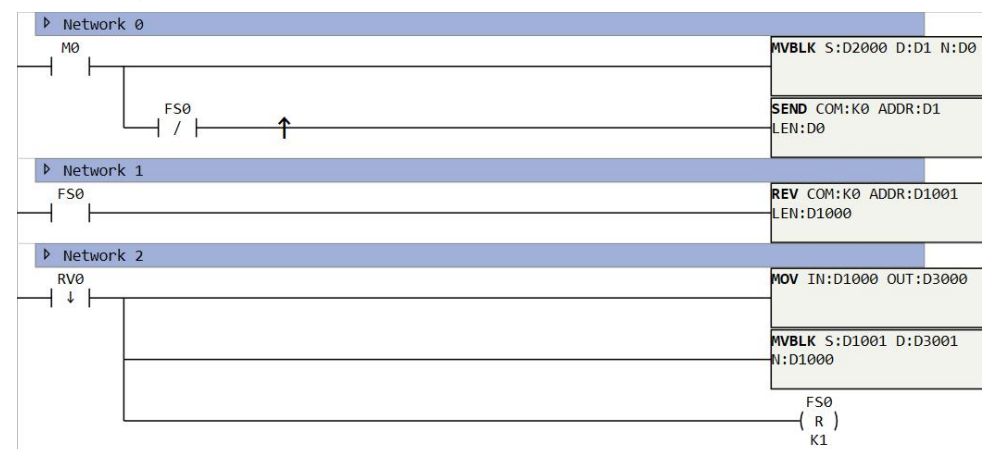


Figure 14.3. 2

REV

Instruction introduction

1. This instruction is applied to free port communication, it receives data from specified serial port (COM), write data into specified D-registers (ADDR), and write data length into specified D-register (LEN).
2. There are system special function registers that record reception status of corresponding serial ports, refer to [Table 14.4.1](#).
3. There is an extra parameter differs from Modbus communication: Buffer bit. When buffer bit is selected with 8, reception only writes low byte of D-registers; When buffer is selected with 16, reception writes high byte and low byte of D-registers. User can set this parameter in communication parameters setting.

Register	Description
M8178	Turns ON when serial port COM0 is receiving.
M8179	Turns ON when serial port COM1 is receiving.
M8182	Turns ON when reception of serial port COM0 is completed.
M8183	Turns ON when reception of serial port COM1 is completed.

Table 14.4. 1

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(COM)	Selected serial port to receive data.	-	-	-
(ADDR)	Head of data-registers to write.	D	-	-
(LEN)	Data length to receive and write	D	-	-

Table 14.4. 2

Attention

1. User need to select communication protocol as free port communication in communication parameters setting.
2. When a serial is used for free port communication, it can't be used for other purpose such as download/upload, monitor, and Modbus communication.
3. Modification of communication parameters takes effects after downloads again.
4. Free port communication is half-duplex, so that the [SEND](#) instruction and **REV** instruction can nor work currently.

Example

1. Example 1

Instruction table:

Network 000

LDP	M0			
REV	K1	D0	D1000	//select COM1, head register is D0, data length is D1000; Assume that COM1 receives data: 0x10 0x11 0x12 0x13 0x14 //if the buffer is 8-bit, D1000=5, D0=0x0010, D1=0x0011, D2=0x0012, D3=0x0013, D4=0x0014 //if the buffer is 8-bit, D1000=5, D0=0x1110, D1=0x1312, D2=0x0014

Ladder diagram:

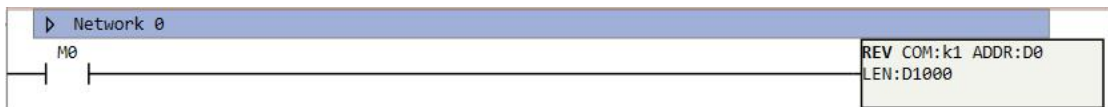


Figure 14.4. 1

2. Example 2

Instruction table:

Network 000

LD	M0			
REV	K0	D1	D0	//receive data from COM0 when M0 is ON and store data in D1

Network 001

LDP	M8182			
SEND	K0	D1001	D1000	//enable SEND after reception is completed
RST	M8182	K1		//reset M8182 automatically

Network 002

LDF	M8178			//close under a falling-edge when reception is completed
MVBLK	D1	D2000	D0	//move stored receive data to D2000

Ladder diagram:

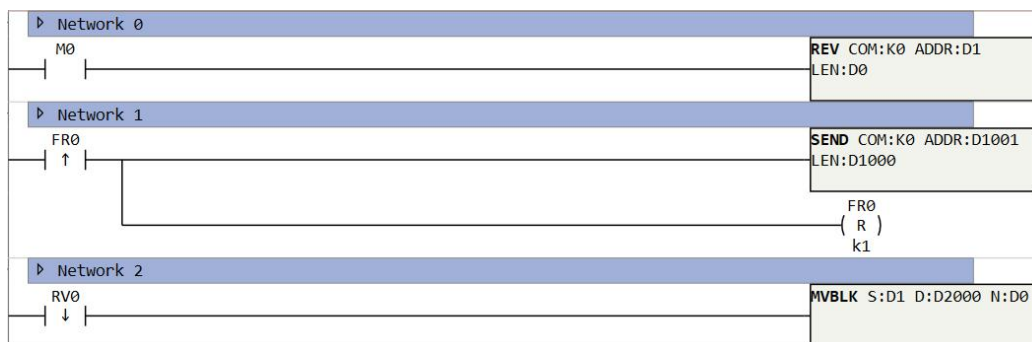


Figure 14.4. 2

CRC

Instruction introduction

1. This instruction calculates specified CRC mode (MODE) check sequence of data with specified length (LEN) store in specified data-registers (DATA).
2. Unit of data length is byte, the data must be stored in D-registers. What should be noted is, the high byte of data register is at right, the low byte of data register is at left.
3. Modes of CRC refer to [Table 14.5.1](#).
4. Generated check sequence is also called redundancy check bit, it will be stored closely after the origin data, these 2 parts constitute CRC code (Cyclic redundancy check code).

CRC mode	Number	Polynomial	Code length
CRC-16	K0	$x^{16}+x^{15}+x^{12}+1$	2 bytes
CRC-12	K1	$x^{12}+x^{11}+x^3+x^2+x+1$	2 bytes
CRC-8	K2	x^8+x^2+x+1	1 byte
CRC-CCITT	K3	$x^{16}+x^{12}+x^5+1$	2 bytes
CRC-32	K4	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$	4 bytes
CRC-32C	K5	$x^{32}+x^{28}+x^{27}+x^{26}+x^{25}+x^{23}+x^{22}+x^{20}+x^{19}+x^{18}+x^{14}+x^{13}+x^{11}+x^{10}+x^9+x^8+x^6+1$	4 bytes

Table 14.5. 1

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(DATA)	Data registers store the origin data.	D	-	binary code
(LEN)	Length of origin data.	K/D	-	16-bit unsigned integer
(MODE)	CRC mode to generate check code.	D	0-5	16-bit unsigned integer

Table 14.5. 2

Example

Instruction table:

Network 000

```

LD      M0
MOV     H3412  D0
MOV     H7856  D1
CRC     D0      K4      K0 //select CRC-16 mode, data length is 4 bytes, calculate CRC check code of
                                0x12345678 and store result 0x107B in D2

```

Ladder diagram:

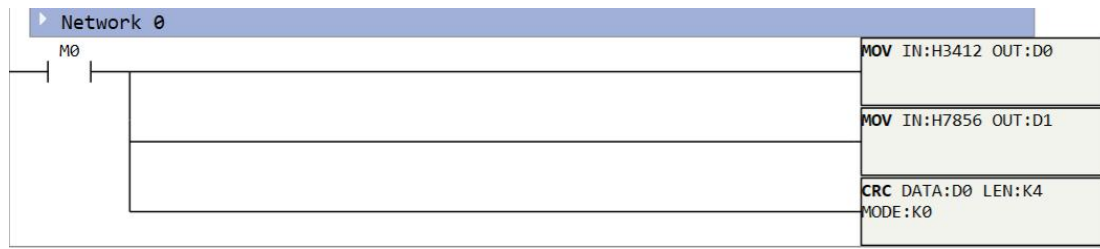


Figure 14.5. 1

CRCHECK

Instruction introduction

1. This instruction checks CRC code under specified CRC mode (MODE) and output result to specified bit-register (RES), the origin data with specified length (LEN) starts from specified data-register (DATA).
2. The CRC code must be stored in D-registers. What should be noted is, the high byte of D-register is at right, the low byte of D-register is at left.
3. Modes of CRC refer to [Table 14.5.1](#).
4. Output bit is ON when check passed, is OFF when check not passed.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(DATA)	Data registers store the origin data.	D	-	binary code
(LEN)	Length of origin data.	K/D	-	16-bit unsigned integer
(MODE)	CRC mode to generate check code.	D	0~5	16-bit unsigned integer

Table 14.6. 1

Example

Instruction table:

Network 000

```

LD      M0
MOV     H3412  D0
MOV     H7856  D1
MOV     H107B  D1
CRCCHECK D0    K6      K0      M1 //origin data is 0x12345678, redundancy check bit is 0x7B10,
                                     check is passed

```

Ladder diagram:

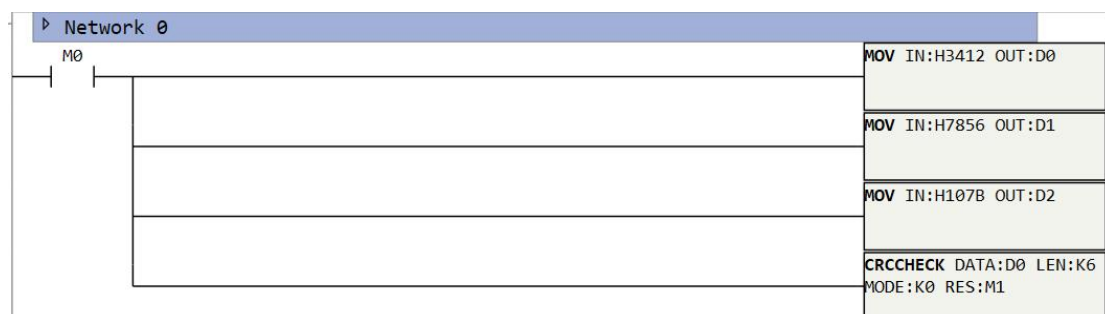


Figure 14.6. 1

Pulse Instructions

Notes

1. So far, for existing series, **FGs_16MT/FGs_32MT/FGRB_C8X8T** owns 2 high-speed pulse output ports(Y0-Y1), and highest frequency of pulse outputting is 200kHz; **FGs_64MT/FGRS_C8X8T/FGRE_C8X8T** owns 4 high-speed pulse output ports(Y0-Y3), and highest frequency of pulse outputting is 200kHz; **FGm_64MT** owns 10 high-speed pulse output ports(Y0-Y7, Y10-Y11), and highest frequency of pulse outputting is 500kHz. Relay type PLC does not support high-speed pulse.
2. high-speed pulse output can be only counted with high-speed counter but not inner counter, also it can't be counted by edge triggering of Y points.
3. In light load situation, the duration of transistor on OFF state will be longer. When less response time is needed in this situation, it is better to add load resistance or enlarge load current.
4. Pulse cumulative counting registers (D8140~D8158) are important registers. They are readable and writable, when be written with new value, the counting will continue on base of new written value.
5. The time high-speed pulse is outputting, value in pulse cumulative counting registers is discrete and varying. When users need to compare the value, use less than or more than instructions, but not equal instruction.
6. When an output point is used for high-speed pulse output, it can't be used as a common output point, so that it will not response to ON or OFF operation.
7. If frequency of pulse output is more than 200k, it's better to add pull-up resistor in output port (resistance is recommended to select 1k when pull-up voltage is 24V).
8. There are special function system registers that store information of instruction running. User can modify some writable registers of them to control. For all pulse instructions, there are registers that record pulse direction, pulse count, pulse count overflow, and pulse outputting status, details refer to [Table 15.1.1](#); For multistage pulse instructions (PLSR, POLYLINE, e.g.), there are registers that record current segment number and error segment number, details refer to [Table 15.1.2](#); For interpolation instructions (POLYLINE, LINE, ARC, e.g.), there are registers that record platform outputting status, details refer to [Table 15.1.3](#).

	Pulse direction register	Counter overflow flag register	Pulse outputting status register	Pulse accumulation counter	Pulse Output port
Data type	bool	bool	bool	32-bit signed integer	-
Description	Show pulse direction of corresponding output port, 1 for positive, 0 for negative	Show if corresponding counter is overflow, 1 for overflow, 0 for not	Show if corresponding output port is outputting pulse, 1 for outputting, 0 for not	Record how many pulses that corresponding output port outputs	Corresponding pulse output port
Register	M_8102 (r)	M_8118 (r)	M_8134 (r)	D_8140 (r/w)	Y000
	M_8103 (r)	M_8119 (r)	M_8135 (r)	D_8142 (r/w)	Y001
	M_8104 (r)	M_8120 (r)	M_8136 (r)	D_8144 (r/w)	Y002
	M_8105 (r)	M_8121 (r)	M_8137 (r)	D_8142 (r/w)	Y003
	M_8106 (r)	M_8122 (r)	M_8138 (r)	D_8144 (r/w)	Y004
	M_8107 (r)	M_8123 (r)	M_8139 (r)	D_8146 (r/w)	Y005
	M_8108 (r)	M_8124 (r)	M_8140 (r)	D_8148 (r/w)	Y006
	M_8109 (r)	M_8125 (r)	M_8141 (r)	D_8150 (r/w)	Y007
	M_8110 (r)	M_8126 (r)	M_8142 (r)	D_8152 (r/w)	Y010
M_8111 (r)	M_8127 (r)	M_8143 (r)	D_8154 (r/w)	Y011	

Table 15.1. 1

	Error segment number register	Current segment number register	Pulse Output port
Data type	16-bit unsigned integer	16-bit unsigned integer	-
Description	Show which segment of path that error occurs	Show which segment of path that current motion point locates in	Corresponding pulse output port
Register	D_8108 (r)	D_8124 (r)	Y000
	D_8109 (r)	D_8125 (r)	Y001
	D_8110 (r)	D_8126 (r)	Y002
	D_8111 (r)	D_8127 (r)	Y003
	D_8112 (r)	D_8128 (r)	Y004
	D_8113 (r)	D_8129 (r)	Y005
	D_8114 (r)	D_8130 (r)	Y006
	D_8115 (r)	D_8131 (r)	Y007
	D_8116 (r)	D_8132 (r)	Y010
D_8117 (r)	D_8133 (r)	Y011	

Table 15.1. 2

	Platform outputting status register
Data type	bool
Description	Show if corresponding platform is outputting, 1 for outputting, 0 for not.
Register	M_8144 (1st) (r)
	M_8145 (2nd) (r)
	M_8146 (3rd) (r)
	M_8147 (4th) (r)
	M_8148 (5th) (r)

Table 15.1. 3

PLSF

Instruction introduction

1. **PLSF** instruction outputs pulse with variable frequency, and there is no transition when frequency transform.
2. When the instruction is enabled, pulse will start outputting with set frequency, and the frequency can be adjusted during running, pulsing stops till instruction is disabled. When frequency is set more than maximum, it will work as maximum.
3. This instruction does not include pulsing direction port output, if there is need for direction output, user can define any output port as direction port. Shown as [Figure 15.2.1](#), Y2, Y3 and Z-phase are all user-defined.

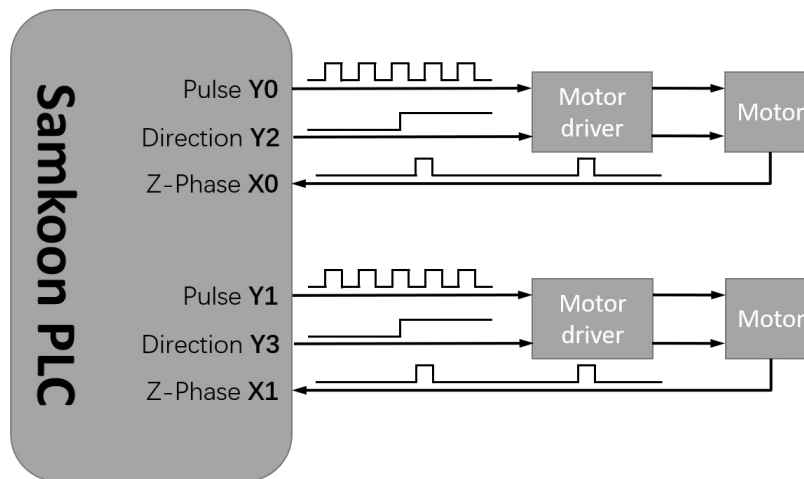


Figure 15.2. 1

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(F)	Frequency of pulse, the unit is Hz.	K/H/D	■ FGs/FGRB/FGRE/FGRS 0~200000 ■ FGm 0~500000	32-bit unsigned integer
(OUT)	Port that outputs the pulse.	Y	-	bool

Table 15.2. 1

Attention

1. Pulse frequency parameter of **PLSF** is 32-bit integer, which occupies two D registers.
2. This instruction outputs pulse without direction, so that the pulse accumulation counter (**D8140-D8158**) can only count in positive direction.

Example

Instruction table:

Network 000

LDP M8151 //enable when PLC run, must be triggered with rising-edge to avoid repeating assignment
 MOVD K400 D0 //original frequency

Network 001

LDP M1
 MOVD K900 D0 //variable frequency stored in D0D1

Network 002

LDP M2
 MOVD K650 D0 //variable frequency stored in D0D1

Network 003

LDP M3
 MOVD K500 D0 // variable frequency stored in D0D1

Network 004

LD M0
 PLSF D0 Y000 //when M0 is ON, Y0 outputs pulse with frequency stored in D0D1 and stop till M0 is OFF, frequency transforms following with value in D0D1 transforms.

Ladder diagram:



Figure 15.2. 2

PWM

Instruction introduction

1. This instruction outputs pulse with variable frequency and variable duty ratio, and there is no transition when frequency transform.
2. When the instruction is enabled, pulse will start outputting with set frequency and duty cycle, the frequency and duty cycle can be adjusted during running, pulsing stops till instruction is disabled.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(F)	Frequency of pulse, the unit is Hz.	K/H/D	■ FGs/FGRB/FGRE/FGRS 0~200000 ■ FGm 0~500000	32-bit unsigned integer
(DC)	Duty cycle of pulse, ratio that signal width occupy in a pulse cycle.	K/H/D	0~100	8-bit unsigned integer
(OUT)	Port that outputs the pulse.	Y	-	bool

Table 15.3. 1

Attention

1. Pulse frequency parameter of **PWM** is 32-bit integer, which occupies two D registers.
2. This instruction outputs pulse without direction, so that the pulse accumulation counter (**D8140-D8158**) can only count in positive direction.
3. Frequency distortion will occur when pulse in high-speed, it's better not to set pulse frequency too high when there is need for precise pulse width control.

Example

Instruction table:

Network 000

```
LD      M0
PWM    D0      D2      Y000 //When M0 is ON, Y0 outputs pulse with frequency stored in D0D1, and
                        duty cycle stored in D2D3
```

Ladder diagram:

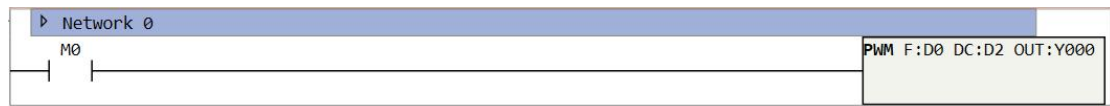


Figure 15.3. 1

PWMS

Instruction introduction

1. This instruction outputs pulse with variable pulse period and variable pulse width.
2. When the instruction is enabled, pulse will start outputting with set pulse period and pulse width, the pulse period and pulse width can be adjusted with immediate effect during running, pulsing stops till instruction is disabled.
3. Time unit of this instruction can be selected with millisecond and microsecond. As wave form distortion may occur when pulse in high-speed, system will automatically limit pulse width when microsecond time unit is selected, for example, when pulse width is set 1us, system will adjust it to 2us to keep the wave form.
4. This instruction provides **M8062** register to transform level output sequence, this register decide outputting effective level or noneffective level at first in a pulse cycle. When **M8062** is ON, effective level at first, when **M8062** is OFF, noneffective level at first. What should be noted is, M8062 should be set before **PWMS** is enable, transforming state of **M8062** during running won't take effect.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(U)	Time unit of pulse period parameter and pulse width parameter, 1 for millisecond and 0 for microsecond.	K/H	0~1	16-bit unsigned integer
(P)	Period of pulse, the unit is millisecond or microsecond.	K/H/D	<ul style="list-style-type: none"> ● millisecond 2~65535 ● microsecond 5~65535 	16-bit unsigned integer
(DP)	width of pulse, the unit is millisecond or microsecond	K/H/D	0~65535	16-bit unsigned integer
(OUT)	Port that outputs the pulse.	Y	-	bool

Table 15.4. 1

Attention

1. Pulse cycle parameter and pulse width parameter of **PWMS** is 16-bit integer, which occupy only one D register.
2. This instruction outputs pulse without direction, so that the pulse accumulation counter (**D8140-D8158**) can only count in positive direction.

3. Frequency distortion will occur when pulse in high-speed, it's better not to set pulse frequency too high when there is need for precise pulse width control.

Example

Instruction table:

Network 000

LD	M0				
PWMS	K0	D0	D1	Y000	//When M0 is ON, Y0 outputs pulse with pulse cycle (ms) stored in D0, and pulse width (ms) stored in D1

Ladder diagram:

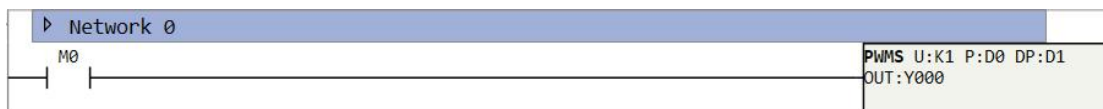


Figure 15.4. 1

PLSY

Instruction introduction

1. This instruction outputs a certain amount of pulse with variable frequency.
2. When the instruction is enabled, pulse will start outputting with set frequency, and the frequency can be adjusted during running, pulsing stops till instruction is disabled. When pulse outputting count reach target pulse number, the pulse will stop.
3. This instruction does not include pulsing direction port output, if there is need for direction output, user can define any output port as direction port. (Refer to introduction of [PLSF](#))

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(F)	Frequency of pulse, the unit is Hz.	K/H/D	■ FGs/FGRB/FGRE/FGRS 0~200000 ■ FGm 0~500000	32-bit unsigned integer
(P)	Target pulse number of outputting.	K/H/D	0~4294967296	32-bit unsigned integer
(OUT)	Port that outputs the pulse.	Y	-	bool

Table 15.5. 1

Attention

1. Frequency parameter and target pulse number of **PLSY** are both 32-bit integer, which occupy two D registers.
2. This instruction outputs pulse without direction, so that the pulse accumulation counter (**D8140-D8158**) can only count in positive direction.

Example

Instruction table:

Network 000

```
LDP M8151 //enable when PLC run, must be triggered with rising-edge to avoid repeating assignment
MOVD K400 D0 //original frequency
```

Network 001

```

LDP    M1
MOVD   K900    D0 //variable frequency stored in D0D1
Network 002
LDP    M2
MOVD   K650    D0 //variable frequency stored in D0D1
Network 003
LDP    M3
MOVD   K500    D0 //variable frequency stored in D0D1
Network 004
LD     M0
PLSY   D0      D2      Y000 //when M0 is ON, Y0 outputs pulse with frequency stored in D0D1 and
                             stop till value in D2D3 reaches target pulse number, frequency
                             transforms
Network 005
LDF    M8134
RST    M0      K1 //when pulse stop, reset M0 to disable PLSF

```

Ladder diagram:

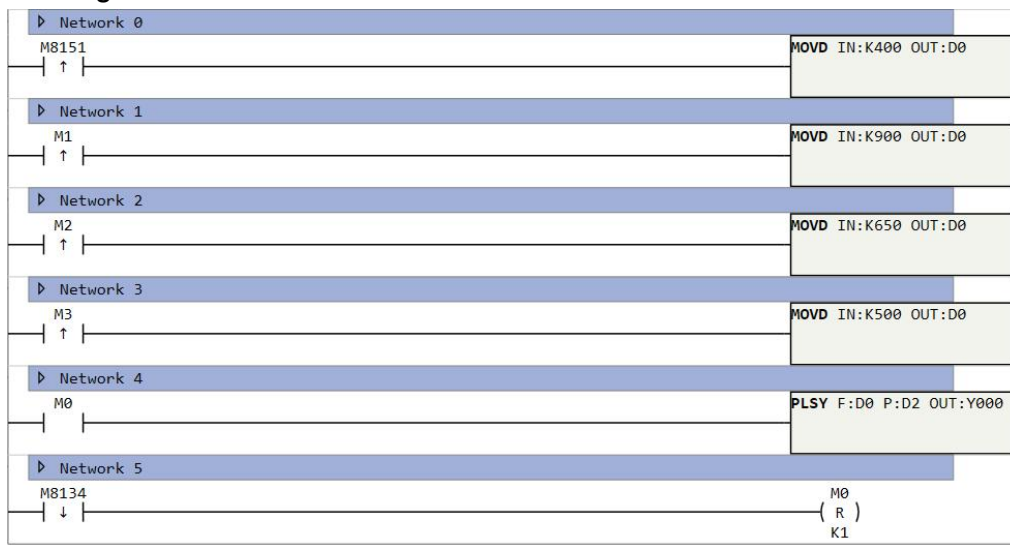


Figure 15.5. 1

Sequence chart:

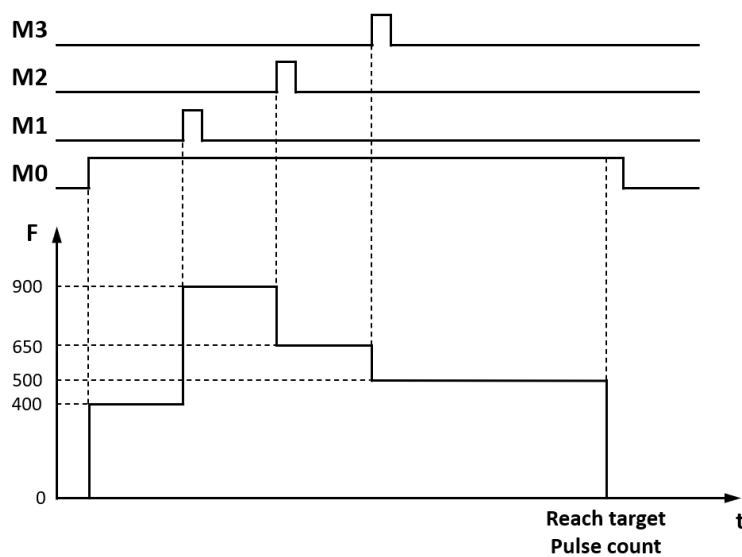


Figure 15.5. 2

PLSR/PLSA/EPLSR

Instruction introduction

1. **PLSR**, **PLSA** and **EPLSR** are all multistage pulse outputting instructions, they output pulse with set frequency and target pulse number, and the parameters can be adjusted dynamically. Target pulse number of **PLSR** is set relatively, target pulse of **PLSA** is set absolutely, **EPLSR** can set acceleration/deceleration time of each segments separately.
2. When **PLSR/PLSA/EPLSR** is enabled, pulse will start outputting with set frequency, and transfer to next segment when meet target pulse number, the transition will be completed in acceleration/deceleration time. And all these running parameters are stored in a piece of D-register area start from first address Dn, [Figure 15.6.1](#) shows the case that first address is D0.
3. When target pulse number of one segment is set -1, pulse outputting will be fixed in the segment till corresponding **PLSNEXT** instruction is enabled. When frequency and target pulse number of one segment are set 0, the segment will be end segment, pulse outputting will stop in this segment regardless of subsequent D-registers.
4. For one single segment of whole, set start frequency of the segment as F_s (Hz), set end frequency of the segment as F_e (Hz), set sum of acceleration and deceleration time as T (s), set pulse count of this segment as N . And these parameters should meet following condition:

$$(F_s + F_e)T \leq 2N$$

5. For **PLSR**, direction outputting is decided by sign of frequency or target pulse number in first segment (if one of them is negative, the direction outputting is negative), and it won't change in subsequent segments. For **PLSA**, direction outputting is decided by target pulse number in each segment, and it will change automatically. For **EPLSR**, direction outputting will also change automatically according to actual pulse count.
6. **M8068** is pulse direction outputting polarity register, when it is ON, all pulse direction outputting will reverse. **M8069** is curve type register, when it is ON, pulse outputting accelerates and decelerates as S-curve type, otherwise as T-curve type.
7. Frequency parameter and target pulse number parameters stored in D-register can be modified in running, but modification is only permitted in segments after current running one.

Setting data

PLSR/PLSA and EPLSR create a piece of D-registers area to store segments parameters. Shown as [Figure 15.6.1](#), for PLSR/PLSA, each four D-registers store parameters of one segment; For EPLSE, each six D-registers store parameters of one segment.

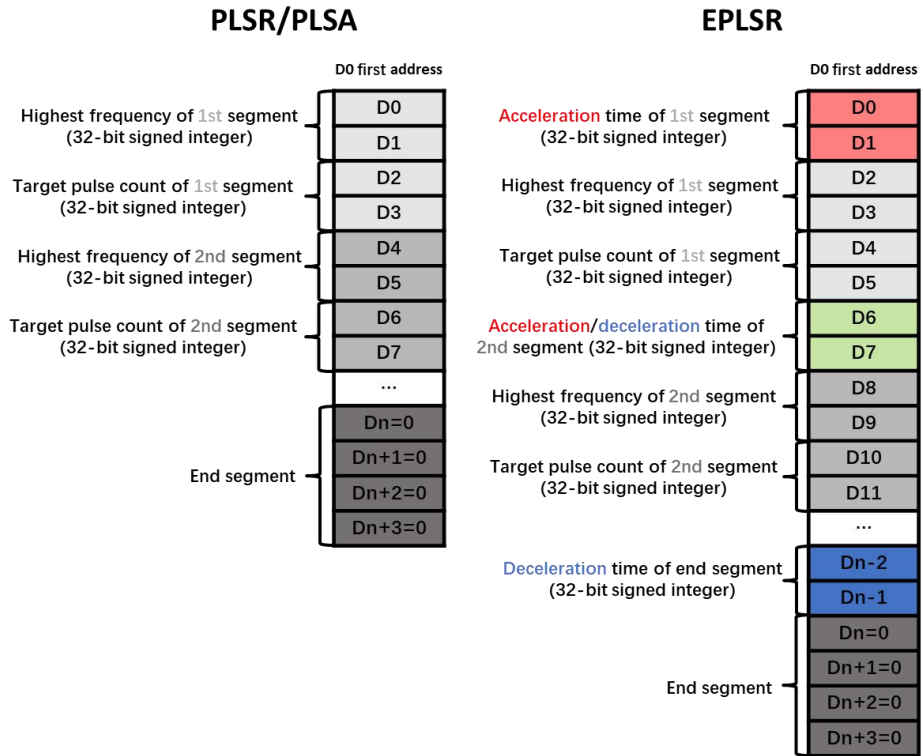


Figure 15.6. 1

➤ **PLSR/PLSA**

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(D)	First address of segments parameters.	D	-	32-bit pointer
(T)	Acceleration/deceleration time when frequency transits.	K/D	0~65535	16-bit unsigned integer
(DIR)	port that output pulse direction.	Y	-	bool
(OUT)	Port that outputs the pulse.	Y	-	bool

Table 15.6. 1

➤ **EPLAR**

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(D)	First address of segments parameters.	D	-	32-bit pointer
(DIR)	port that output pulse direction.	Y	-	bool
(OUT)	Port that outputs the pulse.	Y	-	bool

Table 15.6. 2

Attention

1. Pulse frequency parameter of **PLSR/PLSA/EPLSR** is 32-bit integer, which occupies two D registers.
2. This instruction outputs pulse with direction, so that the pulse accumulation counter (**D8140-D8158**) can count in both directions.

- Pulse outputting stops at segment with 0 frequency and 0 target pulse number, notice that when modifying parameters in running.
- Pulse accumulation counters only count when pulse instruction is abled, the count will clear once instruction is disabled.

Example

- Use **MOVD** instruction to set parameters of PLSR:

Instruction table:

Network 000

```

LDP      M8151
MOVD     K100000  D0 //set pulse frequency of 1st segment to D0D1
MOVD     K200000  D2 //set target pulse number of 1st segment to D2D3
MOVD     K200000  D4 //set pulse frequency of 2nd segment to D4D5
MOVD     K300000  D6 //set pulse frequency of 1st segment to D6D7
FMOVD    K0  D8  K2 //end segment, set 0 to D8-D10

```

Network 001

```

LD       M0
PLSR     D0      K100  Y000  Y004 //PLSR start running when M0 is ON

```

Ladder diagram:

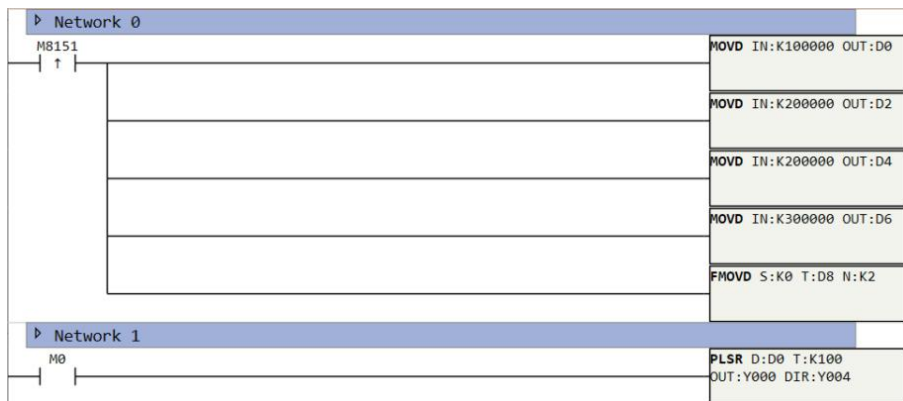


Figure 15.6. 2

- Use function block to set parameters of **PLSR**:
Find PLSR instruction in project explorer through click path: Project -> instruction -> Pulse -> **PLSR**, right click **PLSR**, and user can set function name and block of function. Create this function and there will be a **CALLM** instruction block in ladder diagram (refer to [Figure 15.6.3](#) and [Figure 15.6.4](#)).



Figure 15.6. 3

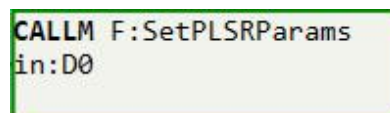


Figure 15.6. 4

Double click the CALLM instruction block, user can fill value in the table to set parameters. The Frequency chart briefly shows how the frequency transform (refer to [Figure 15.6.5](#) and [Figure 15.6.6](#)).

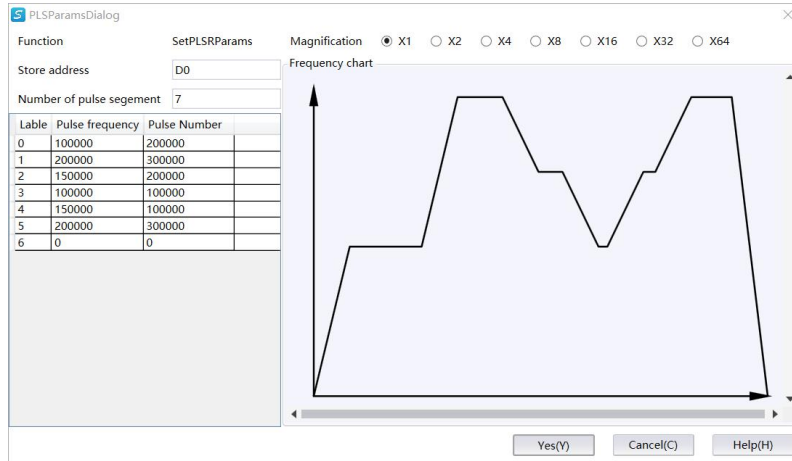


Figure 15.6. 5

```

1  [PLSRParams]
2  void SetPLSRParams (DWORD in)
3  {
4      in[0] = 100000;The highest frequency of pulse segment 1
5      in[1] = 200000;The pulse number of pulse segment 1
6      in[2] = 200000;The highest frequency of pulse segment 2
7      in[3] = 300000;The pulse number of pulse segment 2
8      in[4] = 150000;The highest frequency of pulse segment 3
9      in[5] = 200000;The pulse number of pulse segment 3
10     in[6] = 100000;The highest frequency of pulse segment 4
11     in[7] = 100000;The pulse number of pulse segment 4
12     in[8] = 150000;The highest frequency of pulse segment 5
13     in[9] = 100000;The pulse number of pulse segment 5
14     in[10] = 200000;The highest frequency of pulse segment 6
15     in[11] = 300000;The pulse number of pulse segment 6
16     in[12] = 0;The highest frequency of pulse segment 7
17     in[13] = 0;The pulse number of pulse segment 7
18 }

```

Figure 15.6. 6

Instruction table:

- Network 000
 - LD M8151
 - CALLM SetDPLSRParams D0 //set parameters by function
 - MOVD K100 D100 //set acceleration/deceleration time
- Network 001
 - LD M0
 - PLSR D0 D100 Y000 Y004 //PLSR start running when M0 is ON
- Network 002
 - LDF M8134 //pulse outputting is over
 - RST M0 K1 //reset M0

Ladder diagram:

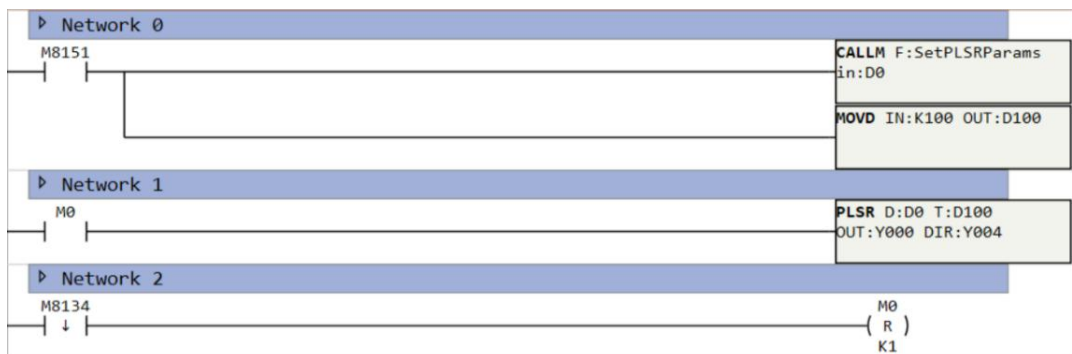


Figure 15.6. 7

PLSNEXT

Instruction introduction

This instruction switches current segment of multistage pulse outputting to next segment. When instruction is enabled and the pulse outputting is in stable frequency, the pulse outputting will accelerate/decelerate to frequency of next segment instantly.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(OUT)	Port that outputs the pulse.	Y	-	bool

Table 15.7. 1

Attention

1. This instruction takes no effect when pulse is accelerating or decelerating. When next segment is end segment and pulse is in stable frequency, enabling this instruction will stop the pulse immediately.
2. This instruction can only be used for **PLSR/EPLSR**.
3. This instruction must be enabled with edge trigger, otherwise the pulse will be in a mess.

Example

Instruction table:

Network 000

```
LD      M8151
MOVD   K100000      D0 //set frequency of 1st segment
MOVD   HFFFFFFF    D2 //set target pulse number of 1st segment to maximum, wait for PLSNEXT
MOVD   K200000     D4 //set frequency of 2nd segment
MOVD   K9999999    D6 // set target pulse number of 2nd segment
FMOVD  K0          D8      K2 //end segment
MOVD   K100        D100 //set acceleration/deceleration time
```

Network 001

```
LD      M0
PLSR   D0          D100   Y000   Y010 //PLSR start running when M0 is ON
```

Network 002

```
LDF   M8134 //pulse outputting is over
RST   M0      K1 //reset M0
```

Network 003

```
LDP   X000 //use rising-edge of X000 to trigger
PLSNEXT Y000 //switch pulse of Y000
```

Ladder diagram:

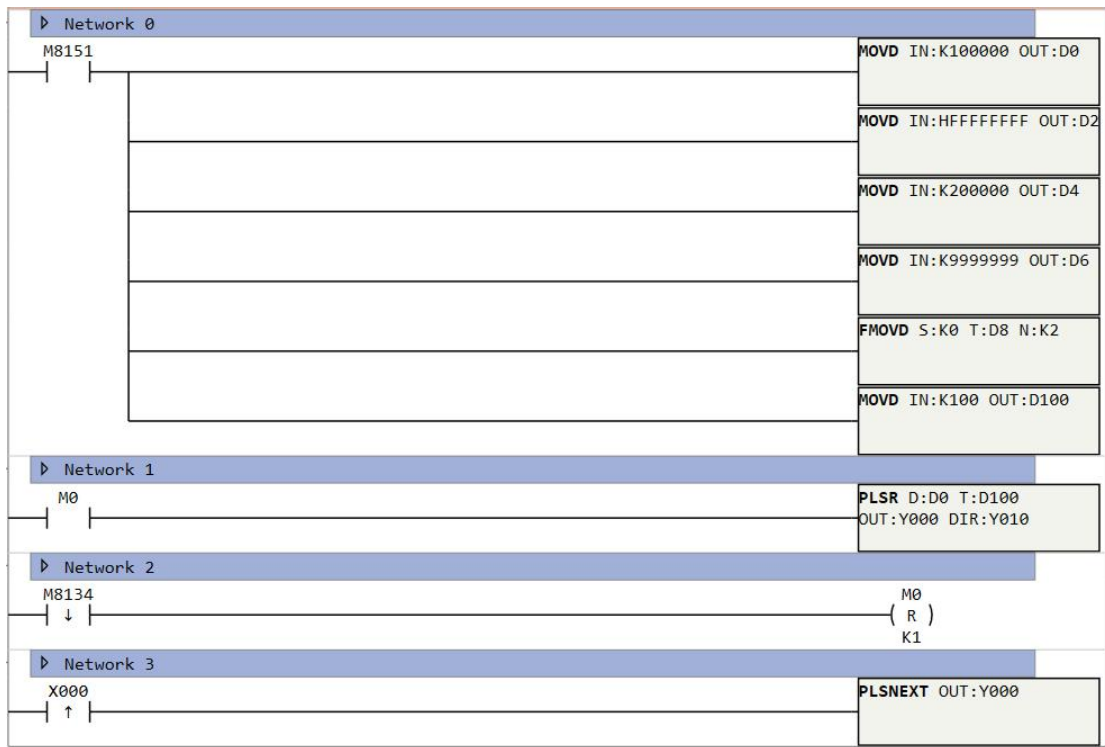


Figure 15.7. 1

PLSSTOP

Instruction introduction

When this instruction is enabled, corresponding pulse output port will pulse stop outputting as selected stop mode. If stop mode is immediate stop, the pulse outputting will stop immediately without deceleration time; if stop mode is slowing down to stop, the pulse outputting will stop in set deceleration time.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(MODE)	Stop mode of pulse outputting, 0 for immediate stop, 1 for slowing down to stop	K	0~1	8-bit unsigned integer
(TIME)	Deceleration time of slowing down to stop.	K/H/D	0~65535	16-bit unsigned integer
(OUT)	Port that outputs the pulse.	Y	-	bool

Table 15.8. 1

Attention

This instruction must be enabled by edge trigger.

Example

Instruction table:

```

Network 000
  LDP    M11      //trigger with rising-edge of M11
  PLSSTOP Y000   K1      K100 //pulse outputting of Y0 stops in 100ms
  
```

Ladder diagram:

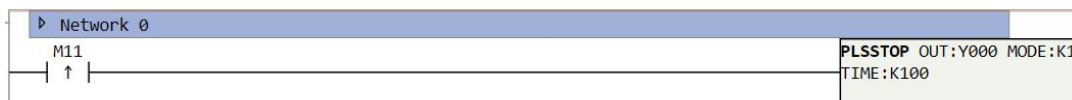


Figure 15.8. 1

DRVI/DRVA

Instruction introduction

1. **DRVI/DRVA** outputs pulse with set frequency, set target pulse number, set acceleration time and deceleration time, **DRVI** is set with relative target pulse number and **DRVA** is set with absolute target pulse number. When the instruction is enabled, pulse starts to accelerate to set frequency in set acceleration time, and decelerates to 0 in set deceleration time and reach set target pulse number. When the instruction is disabled, pulse stop immediately.
2. These instructions can be controlled by special function register **M8068**, when **M8068** is ON, direction port output will reverse.
3. This instruction can be set curve mode by special function register **M8069**, when **M8069** is ON, the frequency curve is S-shape type, when it is OFF, the frequency curve is T-shape type.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(F)	Frequency of pulse outputting, the unit is Hz.	K/H/D	■ FGs/FGRB/FGRE/FGRS 0~200000 ■ FGm 0~500000	32-bit unsigned integer
(P)	Target number of pulse outputting to reach.	K/H/D	-2147483648~2147483647	32-bit signed integer
(A)	Acceleration time of pulse outputting.	K/H/D	0~65535	16-bit unsigned integer
(D)	Deceleration time of pulse outputting	K/H/D	0~65535	16-bit unsigned integer
(OUT)	Port that outputs the pulse.	Y	-	bool
(DIR)	Port that outputs direction of pulse	Y	-	bool

Table 15.9. 1

Attention

This instruction outputs pulse with direction, so that the pulse accumulation counter (**D8140-D8158**) can count in both directions.

Example

Instruction table:

Network 000

```

LDP    M8151
MOVD   K100000  D0 //target pulse number stores in D0D1
MOVD   K50000  D2 //pulse frequency stores in D2D3
MOV    K100    D4 //acceleration time stores in D4
MOV    K200    D5 //deceleration time stores in D5
    
```

Network 001

```

LD     M0
DRVI   D0  D2  D4  D5  Y1  Y6 //pulse starts outputting when M0 is ON
    
```

Ladder diagram:

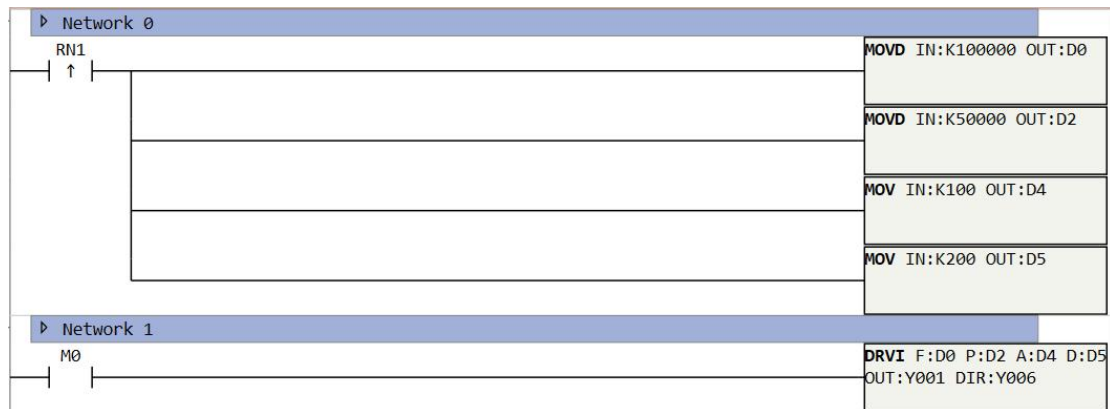


Figure 15.9. 1

POLYLINEF/POLYLINEI

Instruction introduction

1. POLYLINEI/POLYLINEF instructions differs for the parameter types (integer and float), correspondingly they use pls (pulse count) and mm (millimeter) as units.
2. These instructions output pulse to control motion, of which path is combined by lines and arcs. Pulse frequency transits smoothly through joints when two path segments are joined collinearly or tangentially, otherwise pulse decelerates or accelerates near the joints. the configuration of lines and arcs refers to introduction of LINEI/LINEF and ARCI/ARCF instructions.
3. POLYLINE parameters can be mapped to D-registers for dynamic control, user can modify value in D-registers to modify path parameters, appending value to D-registers to add new segments. Additionally, modifying value mapped to velocity parameters in D-registers while motion running is allowable, so that the velocity can be adjust dynamically. If user switches mapping mode to "Only mapping to D-registers", there is no need to input any parameters value in the instruction, the motion path will totally be decided by the assigned data sequence in D-registers. parameter mapping rule details refer to following introduction.

Polyline system setting

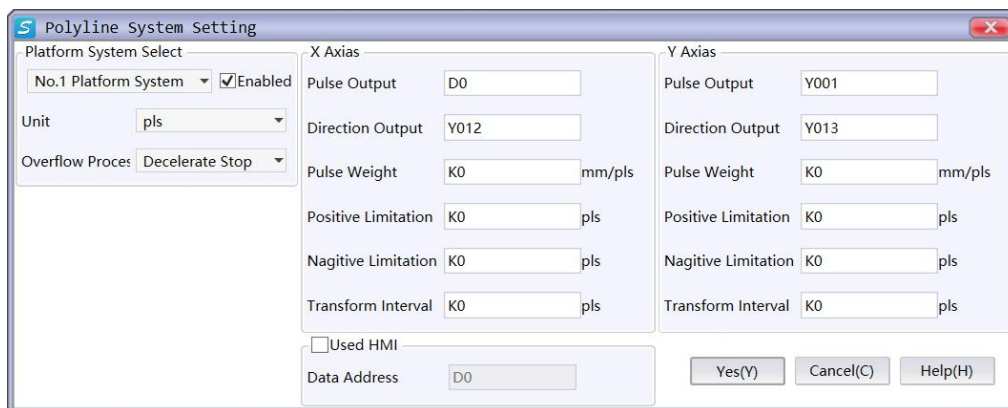


Figure 15.10. 1

When using any interpolation instruction (**POLYLINEI/POLYLINEF**, **LINEI/LINEF**, **ARCI/ARCF**, and **BLOCK**, e.g.), platform system must be set at first. The polyline system creates a platform with 2 axes, and all interpolations are based on this platform.

➤ Brief introduction

1. The click path of polyform system setting is: Main menu -> Pulse -> Pulse platform.
2. There are 5 platform systems to select, they take effect after enabled.
3. there are two kinds of unit to select: mm (millimeter) and pls (pls count). Millimeter

is more recommendable, as the millimeter unit is more realistic, less error-prone, and easy to convert. The unit selection only affects setting of the platform system, actual unit of motion depends on instruction type.

4. Overrun processing is used together with limit of X/Y axis. System will alarm when motion of next segment is going to break the limit, if "Ignored" is selected, overrun will be ignored; If "Slowing down and stop" is selected, motion will stop in end-point of last segment within the limit.
5. Pulse output port must be set with high-speed pulse output port (refer to [first of pulse instruction notes](#)), direction output port can be set with any Y port. And all pulse output ports and direction output ports of enabled platform system must not be repeated.
6. Pulse equivalent means distance that actuator motions per single pulse. In actual case, this parameter can be influenced by many factors, such as motor subdivision, electronic gear ratio, mechanical transmission ratio, and so on. Pulse equivalent is core parameter that link set coordinate to actual measure, it must be accurate.
7. Limit position is theoretical boundary position of platform system, it should be set according to actual situation. When motion breaks the boundary, system will process as selected overrun processing mode.
8. Transmission clearance means clearance that happens when transmission shaft reverses and meets a gap (always caused by abrasion and assembly of mechanical structure). System will compensate the clearance in several interpolation cycle when corresponding axis reverses. There is no need to set this parameter in coarse tuning, but in fine tuning. This parameter is not suitable to be set too big, it is better to adjust mechanical structure when clearance is big. Additionally, this parameter is not applicable for high-speed situation.

➤ **Attention**

1. Pulse equivalent of X-axis and Y-axis should be set equal as far as possible, if not, the system unit should choose pls (pulse count) to avoid conversion troubles.
2. When pulse equivalent of X-axis and Y-axis is set different, all calculation of pulse count unit without relation to specified axis (such as calculation of velocity) will be based on X-axis pulse equivalent.
3. All pulse output ports and direction output ports of enabled platform system must not be repeated.

Operation introduction

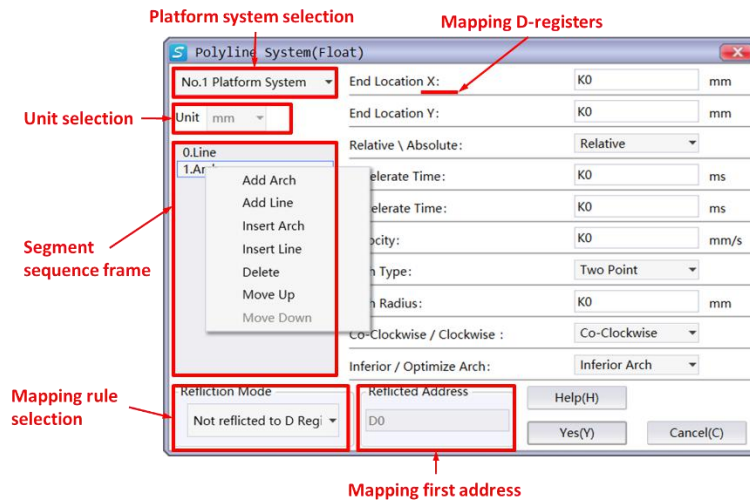


Figure 15.10. 2

1. First, the platform system must be selected and enabled (refer to [platform system setting](#)), otherwise the instruction can't run normally.
2. User can choose mm (millimeter) or pls (pulse count) as parameter unit, respectively correspond to POLYLINEF and POLYLINEI instruction. POLYLINEF is more recommendable, as the millimeter unit is more realistic, less error-prone, and easy to convert.
3. Right click in the left segment-sequence-frame to choose operation. Add, insert, delete, and move operations are selectable. (Configuration of lines and arcs refers to introduction of LINEI/LINEF and ARCI/ARCF).
4. Position and velocity parameters can be input with K or D type data. In the situation velocity parameter is input with D type data, changing the value of relative D-register can also change velocity in real time. Additionally, velocity would not change immediately in acceleration or deceleration, but after acceleration or deceleration. It should be noted that, when using MOV type instruction to assign the D-registers for path parameters, the selection of MOVD and MOVF is decided by the POLYLINE instruction (POLYLINEI/POLYLINEF) type. POLYLINEI is instruction using integer parameters of which unit is pulse count, so that the value must be assigned by MOVD. POLYLINEF is instruction using float parameters of which unit is millimeter, so that the value must be assigned by MOVF. And the unit of acceleration and deceleration parameter is millisecond, it is a 16-bit integer independent of instruction type, it should be assigned by MOV.
5. "Mapping to D-registers" mode means, in the time instruction is enabled, all the set parameters (including position, velocity) will be stored in mapped D-registers in specific order. the mapping rule refers to [Figure 15.10.3](#), also the mapped D-registers address will show before the input box once user select this mode. What should be noted is, the execution of segments interpolation is prospective. The segment after the one in execution will be processed and not permitted to be modified. Modification is only

permitted 2 or more segments after the one in execution, otherwise prospective mistake will appear, lead to confusion and even running error. This function is only applicable when few modifications on path is needed.

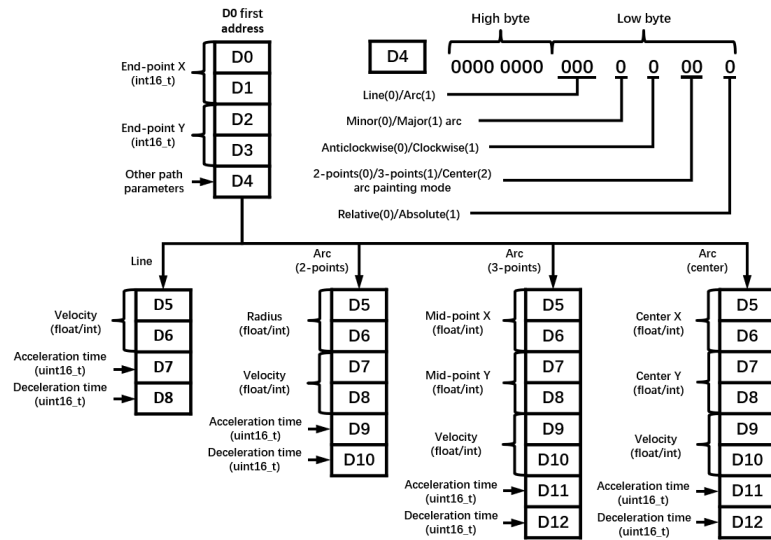


Figure 15.10. 3

- When "Only mapping to D registers" mode is selected, path will only be controlled by values in mapped D-registers, and values input in the instruction takes no effect in this mode. The mapping rule refers to [Figure 15.10. 4](#). Different from "Mapping to D-registers" mode, there is a block of D-registers that save global velocity parameters. If the velocity mode control bit of segments in other parameters is set with 1, the velocity of this segment will be controlled by local velocity parameters (highlighted with red in the figure), otherwise it will be controlled by global velocity parameters. This function is applicable when batch modification on path velocity is needed.

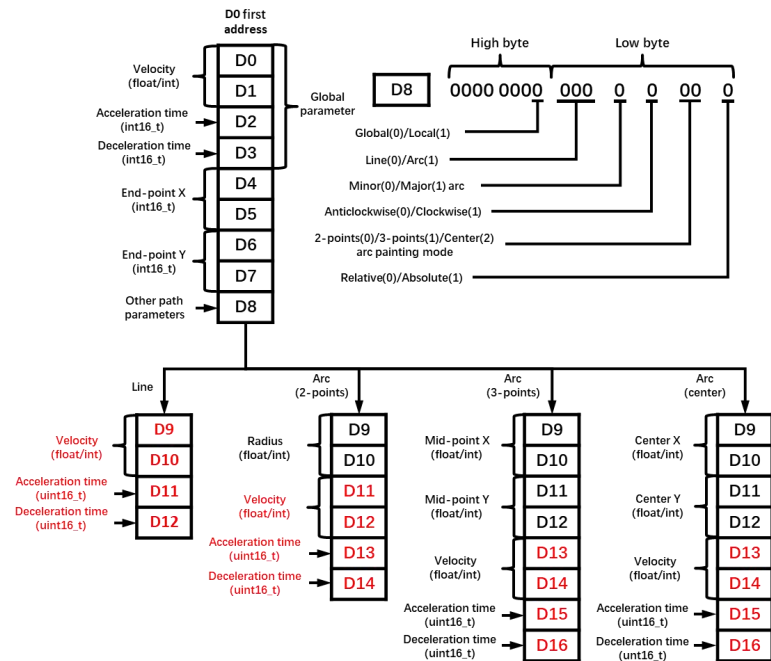


Figure 15.10. 4

- In "Only mapping to D registers" mode, using MOV type instructions to assign value to mapped D-registers will be troublesome and error-prone. Therefore, function templates "OnlyDMMappingI" and "OnlyDMMappingF" are designed for this work. Users can import and use the function blocks refer to [Figure 15.10.5](#) and [Figure 15.10.6](#), function details refer to the annotation in it (the function files locate in "PlaneSys" folder of software installation directory)

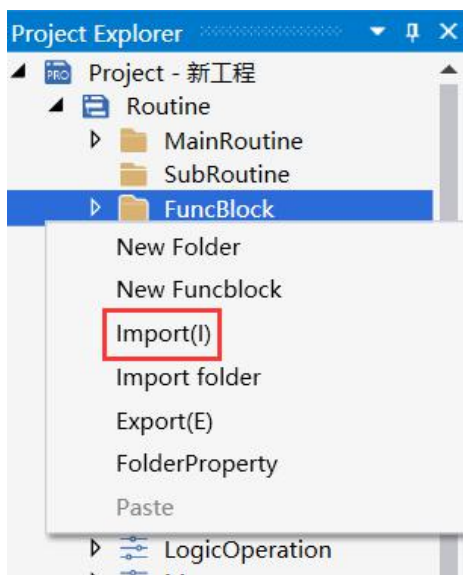


Figure 15.10. 5

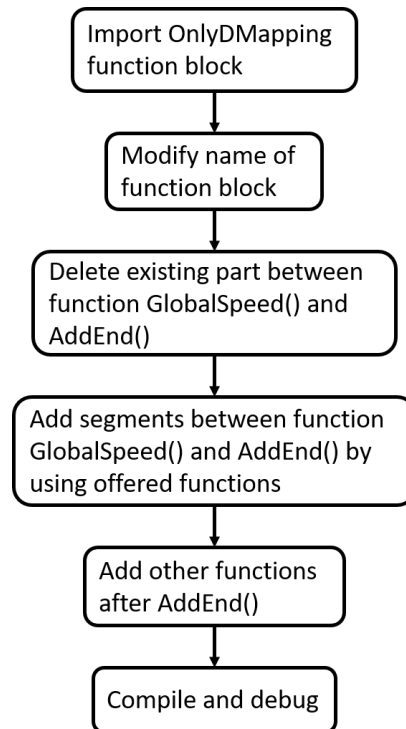


Figure 15.10. 6

Attention

- This instruction outputs pulse with direction, so that the pulse accumulation counter (**D8140-D8158**) can count in both positive and negative direction.
- Acceleration and deceleration time will take no effect if the value is set too small.
- When pulse equivalents of X and Y axes are set different in platform system, by default the real velocity will be calculated with pulse equivalent of X axis.
- M8134 register can only show if Y0 output point is outputting. If there is need to get running state of platform system, use M8144 register that show ON when any axis of platform system outputs pulse.

Example

Use this instruction to paint graph as [Figure 15.10.7](#), interpolation velocity of line is 100mm/s and adjustable, interpolation speed of arc is 50mm/s and fixed.

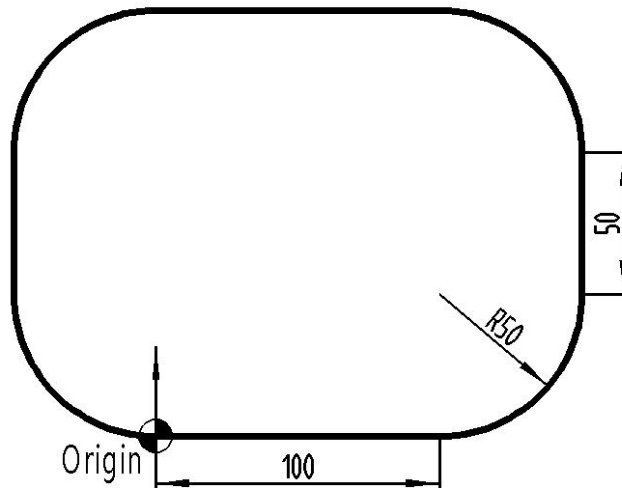


Figure 15.10. 7

Ladder diagram:

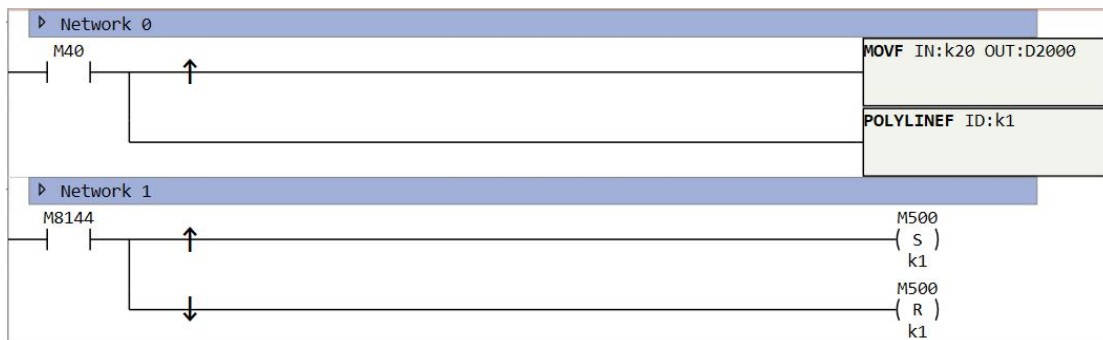


Figure 15.10. 8

1. Not mapping to D-registers

Add lines and arcs in sequence, and set position parameters of them. For speed, line is set with D-type data and assigned value with **MOVD/MOVF** according to instruction type, Arc is set with K-type data K50.

2. Mapping to D-registers

Add lines and arcs in sequence, set first address of mapping D-registers, and set position parameters of them. Adjust value in corresponding mapping register can adjust speed when interpolation running.

3. Only mapping to D-registers

Import function block OnlyDMappingF and set as following code. Additionally, trigger function block once to assign value before enabling interpolation instruction.

```
ParaAddr = GlobalSpeedF(ParaAddr, 100, 200, 200);
```

```
ParaAddr = AddLineF(ParaAddr, 100, 0, 0);
ParaAddr = Add2ArcF_S(ParaAddr, 50, 50, 50, 0, 0, 0, 50, 200, 200);
ParaAddr = AddLineF(ParaAddr, 0, 100, 0);
ParaAddr = Add2ArcF_S(ParaAddr, -50, 50, 50, 0, 0, 0, 50, 200, 200);
ParaAddr = AddLineF(ParaAddr, -100, 0, 0);
ParaAddr = Add2ArcF_S(ParaAddr, -50, -50, 50, 0, 0, 0, 50, 200, 200);
ParaAddr = AddLineF(ParaAddr, 0, -100, 0);
ParaAddr = Add2ArcF_S(ParaAddr, 50, -50, 50, 0, 0, 0, 50, 200, 200);
```

```
ParaAddr = AddEndF(ParaAddr);
```

LINEF/LINEI

Instruction introduction

1. LINEI/LINEF instructions differs for the parameter types (integer and float), correspondingly they use pls (pulse count) and mm (millimeter) as units.
2. These instructions complete straight-line path from current point to end point in specified platform system.
3. relations of line path length (L), velocity (V), acceleration time (T_{ac}) and deceleration time (T_{de}) should meet condition below (value of these parameters will be adjusted automatically if their relations do not meet the condition):

$$0.5 \times (T_{ac} + T_{de}) \times V \leq L$$

If the parameters were set too unreasonable to adjust automatically, instruction may stop outputting pulse when error was checked. Users can modify parameters correspond to the error code saved in **D8176** register.

Operation introduction

1. First, the polyline number must be selected and enabled (refer to platform system setting), otherwise the instruction can't run normally.
2. User can choose mm (millimeter) or pls (pulse count) as parameter unit, respectively correspond to **LINEF** and **LINEI** instruction. **LINEF** are more recommendable, as the millimeter unit is more realistic, less error-prone, and easy to convert.
3. End-point X and end-point Y are the coordinates of final position of line path, these parameters can be input with K or D type data. The selection of **MOVD** and **MOVF** to assign value of coordinates parameters is decided by the LINE instruction (**LINEI/LINEF**) type.
4. When relative coordinate mode is selected, the coordinate of end point is based on offset from current point. When absolute coordinate mode is selected, the coordinate of end point is based on offset from origin point. These 2 modes refer to [Figure 15.11.1](#).

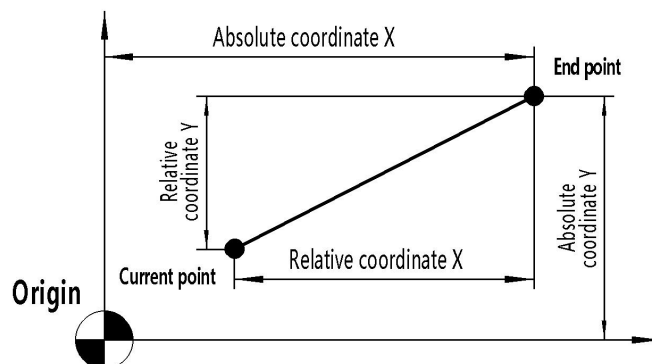


Figure 15.11. 1

- Acceleration and deceleration time parameters can be set with K or D type data. If set with D type data, during initialization instruction will read value from D registers as acceleration and deceleration time. Acceleration and deceleration time parameters are 16-bits integer, must be assigned with **MOV** instruction. Limited by interpolation period (4ms) of algorithm, frequency hopping will be too sharp to make acceleration and deceleration effective if time was set too small. The acceleration and deceleration of interpolation is asymmetric s-shape acceleration and deceleration, details of shape refer to [Figure 15.11.2](#).

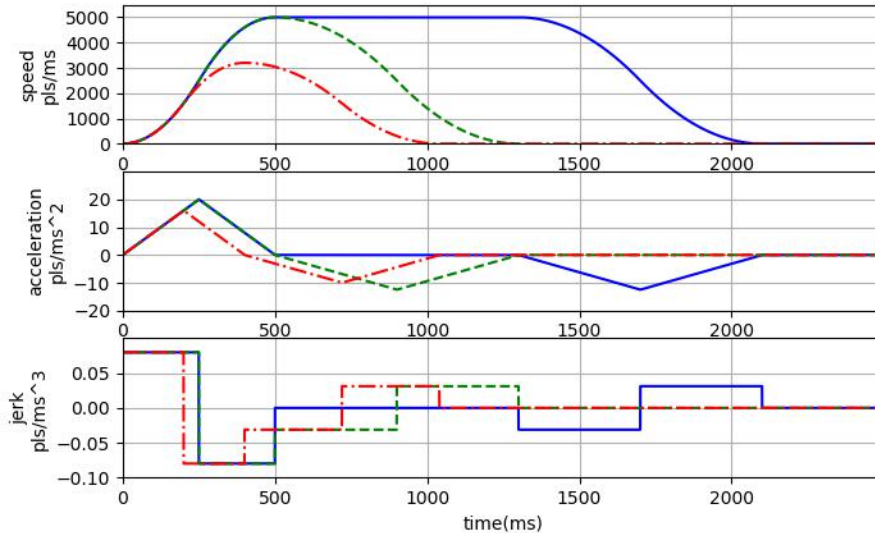


Figure 15.11. 2

- Speed parameter means resultant velocity of X-axis and Y-axis, this parameter can be set with K or D type data. If set with K type data, the speed of line path will be non-adjustable; if set with D type data, the speed can be adjusted by the value in corresponding D register. Also, the selection of **MOVD** and **MOVF** to assign value of speed parameters is decided by the **LINE** instruction (**LINEI/LINEF**) type. If pulse equivalents of X and Y axis are set different, system will calculate speed with pulse equivalents of X-axis by default. When acceleration time, deceleration time and speed parameters are set illegal, system will adjust these parameters to make them legal. The adjustment refers to [Figure 15.11.2](#) (blue line means normal case, green line means limit case, red line means adjusted case).

Attention

- This instruction outputs pulse with direction, so that the pulse accumulation counter (**D8140-D8158**) can count in both directions.
- Acceleration/deceleration takes no effect when it is set too small.
- When unit is set with pls, the unit of speed is pls/s. If pulse equivalent of X-axis and Y-axis are set different, by default the resultant velocity is calculated on base of X-axis pulse equivalent.

- Special function register **M8134** to **M8153** only shows if corresponding pulse output port is outputting, if there is need to detect if interpolation is running, user should use **M8144** to **M8148** that shows if corresponding platform system is running.

Example

Paint a line from current point to end-point, when interpolation starts, M500 is set, when interpolation is over, M500 is reset. Platform system set Y0 port as X-axis pulse outputting port, set Y1 as Y-axis pulse outputting port.

Ladder diagram:

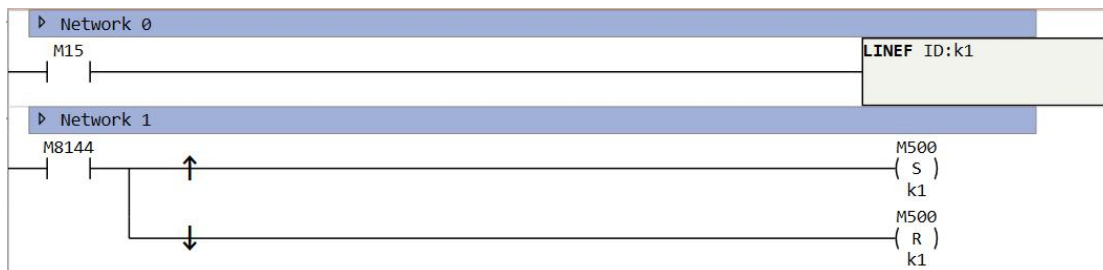


Figure 15.11. 3

ARCF/ARCI

Instruction introduction

1. ARCI/ARCF instructions differs for the parameter types (integer and float), correspondingly they use pls (pulse count) and mm (millimeter) as units.
2. These instructions complete arc path from current point to end point in specified platform system.
3. relations of line path length (L), velocity (V), acceleration time (T_{ac}) and deceleration time (T_{de}) should meet condition below (value of these parameters will be adjusted automatically if their relations do not meet the condition):

$$0.5 \times (T_{ac} + T_{de}) \times V \leq L$$

If the parameters were set too unreasonable to adjust automatically, instruction may stop outputting pulse when error was checked. Users can modify parameters correspond to the error code saved in **D8176** register.

4. In 2-point arc painting mode, it should be noted when setting parameters, length from end point to current point must be less than arc radius, otherwise system will regard the length as diameter and re-profiling arc path.
5. In 3-point arc painting mode, it should be noted when setting parameters, end point, mid-point and current point must not be collinear.
6. In center arc painting mode, it should be noted when setting parameters, center point must be at middle perpendicular of link line between end point and current point.
7. The path of arc interpolation is composed with micro line path, there will be chord error between actual path and theoretic path (refer to [Figure 15.12.1](#)). When the arc radius parameter is small and interpolation speed parameter is big, even if the parameters is legal, chord error still can't be ignored. The relations of chord error, interpolation speed and arc radius refer to [Figure 15.12.2](#), chord error will be prominent in high speed and short radius. System will lower the speed to avoid big chord error, also there should not be arc with too small radius in actual use.

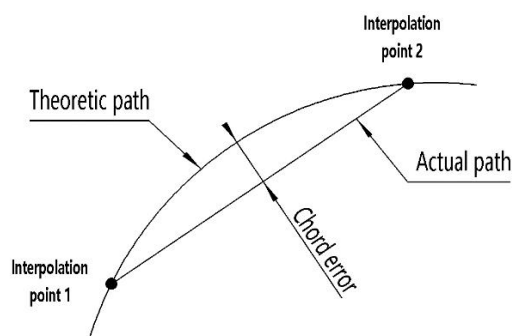


Figure 15.12. 1

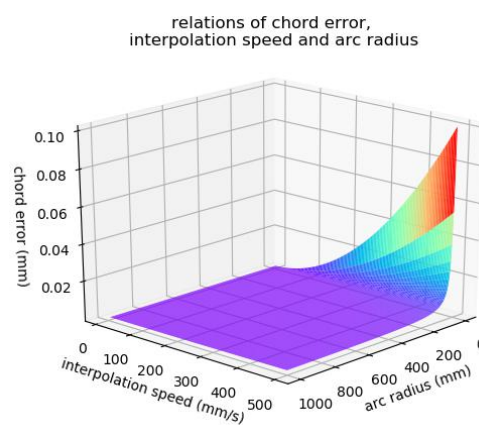


Figure 15.12. 2

Operation introduction

1. First, the polyline number must be selected and enabled (refer to platform system setting), otherwise the instruction can't run normally.
2. User can choose mm (millimeter) or pls (pulse count) as parameter unit, respectively correspond to **LINEF** and **LINEI** instruction. **LINEF** are more recommendable, as the millimeter unit is more realistic, less error-prone, and easy to convert.
3. End-point X and end-point Y are the coordinates of final position of arc path, these parameters can be input with K or D type data. The selection of **MOVD** and **MOVF** to assign value of coordinates parameters is decided by the LINE instruction (**LINEI/LINEF**) type.
4. When relative coordinate mode is selected, the coordinate of end point is based on offset from current point. When absolute coordinate mode is selected, the coordinate of end point is based on offset from origin point. These 2 modes refer to [Figure 1.11.1](#).
5. Acceleration and deceleration time parameters can be set with K or D type data. If set with D type data, during initialization instruction will read value from D registers as acceleration and deceleration time. Acceleration and deceleration time parameters are 16-bits integer, must be assigned with **MOV** instruction. Limited by interpolation period (4ms) of algorithm, frequency hopping will be too sharp to make acceleration and deceleration effective if time was set too small. The acceleration and deceleration of interpolation is asymmetric s-shape acceleration and deceleration, details of shape refer to [Figure 15.11.2](#).
6. Speed parameter means resultant velocity of X-axis and Y-axis, this parameter can be set with K or D type data. If set with K type data, the speed of line path will be non-adjustable; if set with D type data, the speed can be adjusted by the value in corresponding D register. Also, the selection of **MOVD** and **MOVF** to assign value of speed parameters is decided by the **ARC** instruction (**ARCI/ARCF**) type. If pulse equivalents of X and Y axis are set different, system will calculate speed with pulse equivalents of X-axis by default. When acceleration time, deceleration time and speed parameters are set illegal, system will adjust these parameters to make them legal. The adjustment refers to [Figure 15.11.2](#).
7. Arc painting modes include 2-points mode, 3-points mode and center mode, these modes use different parameters. In 2-points mode, refer to [Figure 15.12.3](#), user need to confirm which path to go between current point and end point decided by turning direction (clockwise or anti-clockwise) and arc length (minor or major arc). In 3-points mode, refer to [Figure 15.12.4](#), user only need to confirm a mid-point between current point and end point. In center mode, refer to [Figure 15.12.5](#), user only need to confirm the center of arc.

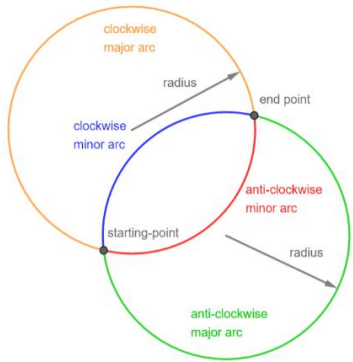


Figure 15.12. 3

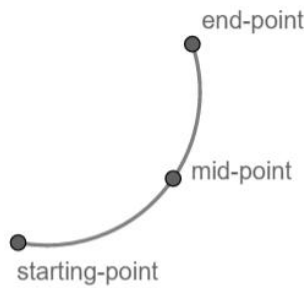


Figure 15.12. 4

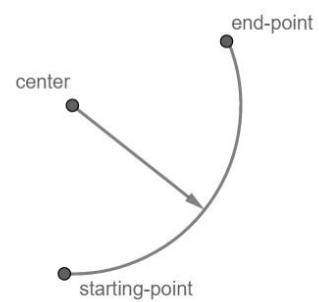


Figure 15.12. 5

Attention

1. In 2-points arc painting mode, length from end point to current point must be less than arc radius.
2. In 3-points arc painting mode, end point, mid-point and current point must not be collinear.
3. In center arc painting mode, center must be at middle perpendicular of link line between end point and current point.
4. End-point must not overlap with current point.

Example

Paint an anticlockwise minor arc form current point to end-point with 100mm radius, when interpolation starts, M500 is set, when interpolation is over, M500 is reset. Platform system set Y0 port as X-axis pulse outputting port, set Y1 as Y-axis pulse outputting port.

Ladder diagram:

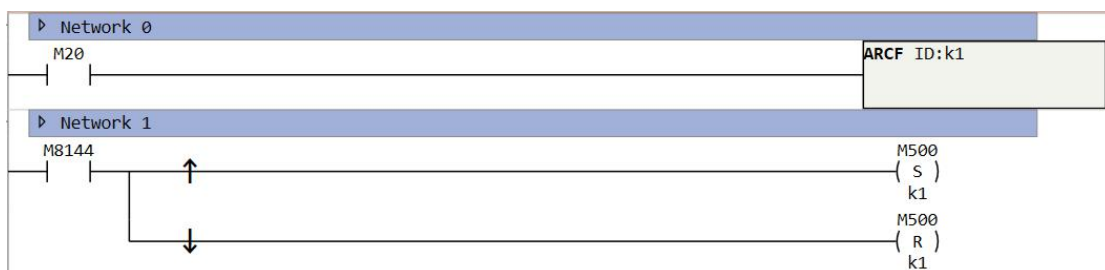


Figure 15.12. 6

BLOCK

Instruction introduction

1. This instruction controls pulse to paint imported graph in specified platform system.
2. For the present, this instruction only supports imported graph from AutoCAD that combined with line and arc, additionally it only supports DXF file format.

Operation introduction

1. First, using AutoCAD software to paint graph with line and arc only, and what should be noted is, user should confirm the size of platform system to avoid the painted graph crossing the border of platform system. imported graphs are all painted with absolute coordinate, so the origin point of platform system need to overlap with the origin point of world coordinates in AutoCAD, set size unit to millimeter, and check if all the segments are necessary to paint in graph, and it is recommended to save graph as 2014 DXF.
2. Set and enable platform system.
3. Import graph from click path: main menu -> pulse -> import graph. Refer to [Figure 15.13.1](#).
4. After completion of import, choose platform system which the graph located within, and set the speed parameter.
5. Add a **BLOCK** instruction in the ladder graph, choose which graph to paint and set flag-bit. During the drawing process, some graph can't be completed with one stroke, transition lines are needed among these graph parts. Those transition lines need not to paint are called imaginary lines (dotted lines), those lines need to paint are called real lines (solid lines). Flag-bit is used to distinguish between real lines and imaginary lines, it shows 1 in real lines and shows 0 in imaginary lines.

Attention

1. Acceleration and deceleration time will take no effect if the value is set too small.
2. During painting process, speed parameter will be adjusted automatically if it is illegal, the speed will slow down according to shapes of graph path.
3. Size of imported graph file is not permitted to be too big. otherwise graph is not able to be downloaded.

Example

Painting graph as [Figure 15.13.2](#), import file Luffy.dxf as [Figure 15.13.1](#) shows, set flag-bit as

Y6, set and enable platform system, set velocity with D0, set acceleration time with K200, set deceleration with K200.

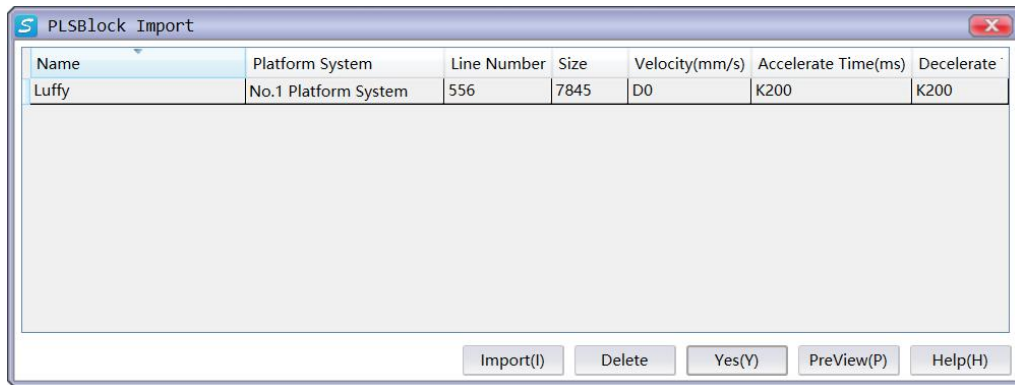


Figure 15.13. 1



Figure 15.13. 2

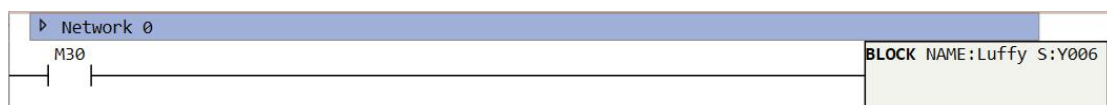


Figure 15.13. 3

PAUSE

Instruction introduction

1. This instruction can only be used for **POLYLINE** and **BLOCK**.
2. This instruction takes effects during painting process of POLYLINE and BLOCK. pulse outputting of corresponding platform system will stop when PAUSE is enabled, and continue when PAUSE is disabled, PAUSE dose not influence paths of graph painting.

Operation introduction

1. At first, set relevant parameters of **POLYLINE** or **BLOCK** instruction.
2. Set platform system of PAUSE instruction in accord with platform system in **POLYLINE** or **BLOCK** instruction.
3. Enable **PLOYLINE/BLOCK** instruction to start graph painting as setting. Enable PASUE instruction when pause of painting is needed, then painting will slow down and stop, the deceleration time is in accord with deceleration time set in PLOYLINE/BLOCK instruction. Disable PAUSE instruction when continuation of painting is needed.

Attention

1. Pause may not always respond immediately, **PAUSE** will not start during acceleration/deceleration of graph painting, but when speed getting stable.
2. During pausing, do not disable the corresponding interpolation instruction, otherwise when the interpolation instruction is enabled, the graph painting will restart from beginning but not point where it pauses.
3. After pause, the interpolation instruction will still occupy the pulse and direction ports. The stop state after pause is unable and not permitted to be remove.

Example

Ladder diagram:

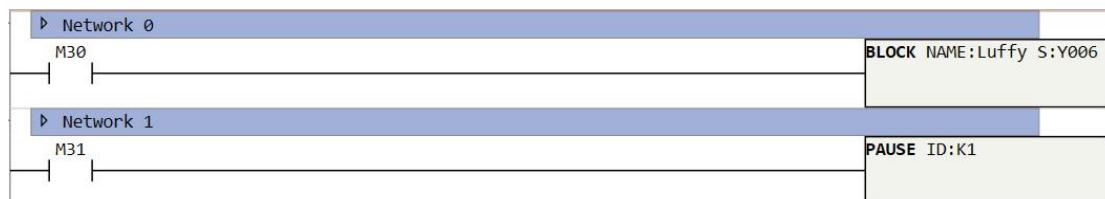


Figure 15.14 1

ZRNR

Instruction introduction

This instruction controls workbench to return to zero from any position. When this instruction is enabled, according to default zero returning direction and mode, pulse will be generated and stop besides particular range until meets zero signal. When this instruction is disabled, the pulse will stop immediately.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(RS)	Returning speed of workbench.	K/H/D	-2147483648 to 2147483647	32-bit unsigned integer
(CS)	Crawl speed of workbench.	K/H/D	-2147483648 to 2147483647	32-bit signed integer
(A)	Acceleration time of pulse outputting.	K/H/D	0 to 65535	16-bit unsigned integer
(D)	Deceleration time of pulse outputting.	K/H/D	0 to 65535	16-bit unsigned integer
(OS)	Origin signal input port.	X/M	-	bool
(PL)	Positive limit signal input port.	X/M	-	bool
(NL)	Negative limit signal input port.	X/M	-	bool
(OUT)	Port that outputs the pulse.	Y	-	bool
(DIR)	Port that outputs direction of pulse	Y	-	bool

Table 15.15. 1

Parameter introduction

1. Crawl speed and returning speed parameters can be inputted with 32-bit K/D/H(hex) type value. When both speed parameters are set with minus value, the default zero returning direction will be changed, additionally the zero-signal position will also be changed.
2. Pulsing port and pulsing direction assign which port to output pulse and pulse direction.
3. Zero signal, positive/negative limit signal and Z-phase signal input ports should be inputted with ports that connect to corresponding sensor (signal input port can be inputted with X port or M register), among them, positive/negative limit signal and Z-phase signal are optional.

4. Polarity of zero signal and positive/negative limit signal should be decided by actual signal polarity of corresponding sensor (normally-open or normally-closed).
5. In most cases (start position being beyond zero signal area), default zero returning direction decides the motion direction of workbench from start position to zero.
6. There are 2 kinds of zero returning mode, near-point signal mode and zero positive-edge mode, zero returning path and position of 2 modes are different (refer to schematic plot of zero returning).
7. Together, zero returning mode and default zero returning direction decide the final zero position (on which side of zero signal area).

Attention

1. After using this instruction to complete zero returning action, pulse accumulation counters of corresponding output ports will be reset automatically as origin point.
2. Filtering mode of sensors connected to X input ports should be set OFF, or there will be large error on final zero position.
3. When the signals of pulsing ports are high level, the direction of workbench motion (motor rotation) is positive direction, the corresponding limit is positive limit; otherwise, the direction is negative direction, the corresponding limit is negative limit.
4. During zero returning, the outputs of pulsing direction is controlled by PLC automatically according to actual position of workbench, no need for manual control (to make sure the outputs of pulsing direction are influenced by other instructions).
5. If the default zero returning direction is selected positive, in most cases (start position being beyond zero signal area) the pulsing direction starts with high level, the workbench moves positively; If the default zero returning direction is selected negative, it's opposite. User can select appropriate default zero returning direction according to actual case (such as in which area start position located when zero returning starts).
6. User should affirm final zero position before using. Refer to [Figure 15.15.1](#), it shows the final zero position on near-point signal mode. Final zero position will locate in side of zero signal area near positive limit if default zero returning direction is positive, or near negative limit if default zero returning direction is negative.

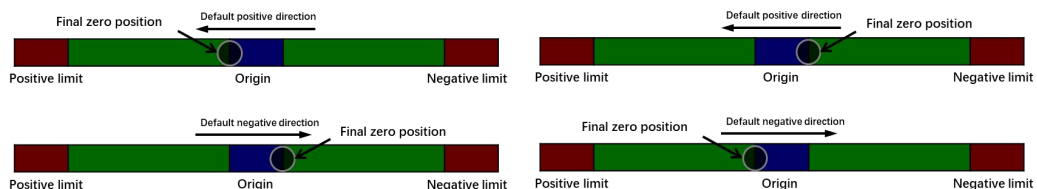


Figure 15.15 1

7. Positive/negative limit signal and Z-phase signal are optional, if neither of 2 limit signals is selected, zero returning from any position can't be realized (the following zero returning paths that relate to unselected limit signals will not be realized, others paths that unrelated to unselected limit signals can be realized still). User can select Z-phase signal as need (Z-phase pulsing of servo can be connected to PLC. What should be noted is the signal must be in 24V level, please transform it if not, otherwise PLC will not

- recognize the signal).
8. If workbench stops when entering positive/negative limit area during zero returning, and the error message shows "Zero signal lost", please check if the sensor of zero signal works normally. After troubleshooting, zero returning still can work by enabling instruction. If workbench stops during zero returning and the error message shows "Acceleration and deceleration distance is too long", this indicates the workbench reach and even overstep the limit signal area, please check position of workbench and lower acceleration and deceleration time.
 9. If workbench can't decelerate to crawl speed in zero signal area, please widen zero signal area appropriately or lower acceleration and deceleration time to make enough area for workbench slowing down to crawl speed.
 10. To ensure safety, please make sure the limit signal area is wide enough for workbench to decelerate to 0.
 11. To ensure safety, it is recommended to select polarity of positive/negative signal as normally-closed to avoid limit signal losing efficacy.
 12. returning error can be influenced by following 3 factors: Final motion speed when meet zero position (if the speed is lower, the error is smaller; Scan period of PLC (if the scan period is shorter, the error is smaller). Motion inertia of workbench (if the inertia is smaller, the error is smaller).

Schematic plot of zero returning

On case of negative default zero returning direction, the following schematic plots show how workbench motion during zero returning in 2 modes (In the plots, circle within number means point during zero returning path, red arrow means accelerated motion, green arrow means uniform motion, blue arrow means decelerated motion).

1. Start position locates between positive limit signal area and zero signal area.
 - Near-point signal mode, as show in [Figure 15.15.2](#):

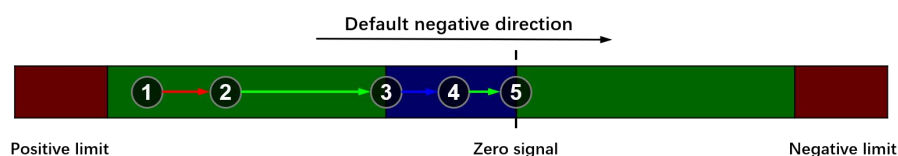


Figure 15.15.2

- ①→②: Workbench accelerates to returning speed in negative direction.
- ③→④: Workbench enters zero signal area and decelerates to crawl speed.
- ⑤: Workbench stops at side of zero signal area, if Z-phase is selected, workbench stops at Z-phase signal.
- Zero positive-edge mode, as shown in [Figure 15.15.3](#):

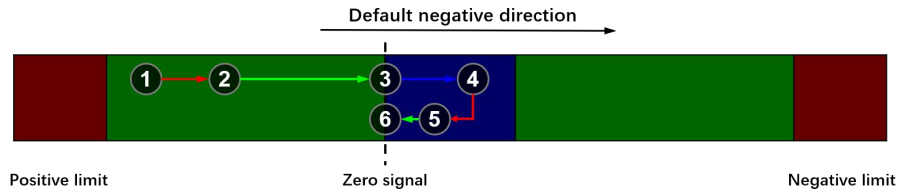


Figure 15.15.3

- ①→②: Workbench accelerates to returning speed in negative direction.
- ③→④: Workbench enters zero signal area and decelerates to 0.
- ④→⑤: Workbench accelerates to crawl speed in opposite direction.
- ⑥: Workbench stops at side of zero signal area, if Z-phase is selected, workbench stops at Z-phase signal.

2. Start position locates in positive limit signal area.

- Near-point signal mode, as show in [Figure 15.15.4](#):

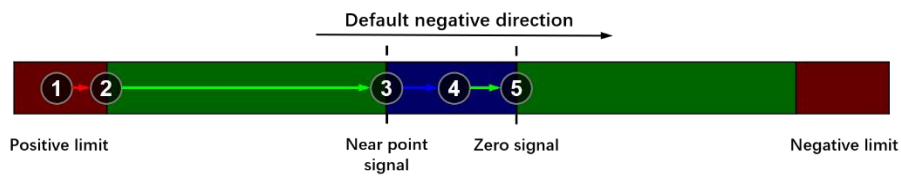


Figure 15.15.4

- ①→②: Workbench accelerates to returning speed in negative direction.
- ③→④: Workbench enters zero signal area and decelerates to crawl speed.
- ⑤→⑥: Workbench stops at side of zero signal area, if Z-phase is selected, workbench stops at Z-phase signal.

⑦:

- Zero positive-edge mode, as shown in [Figure 15.15.5](#):

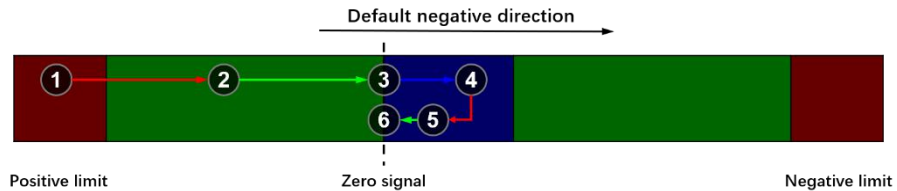


Figure 15.15.5

- ①→②: Workbench accelerates to returning speed in negative direction.
- ③→④: Workbench enters zero signal area and decelerates to 0.
- ④→⑤: Workbench accelerates to crawl speed in opposite direction.
- ⑥: Workbench stops at side of zero signal area, if Z-phase is selected, workbench stops at Z-phase signal.

3. Start position locates in zero signal area.

- Near-point signal mode, as show in [Figure 15.15.6](#):

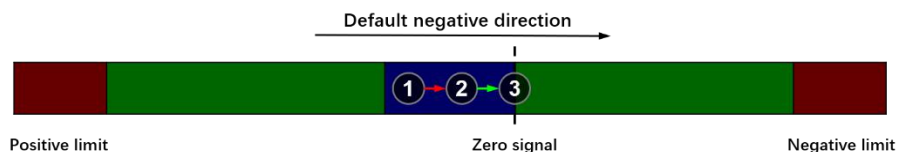


Figure 15.15.6

- ①→②: Workbench accelerates to crawl speed in negative direction.

③: Workbench stops at side of zero signal area, if Z-phase is selected, workbench stops at Z-phase signal.

- Zero positive-edge mode, as shown in [Figure 15.15.7](#):

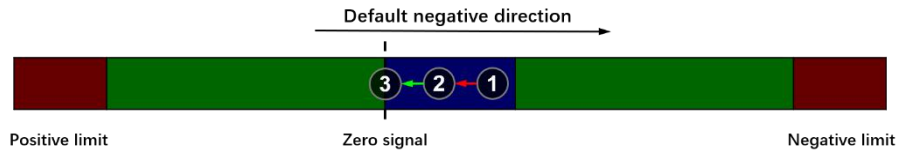


Figure 15.15 7

①→②: Workbench accelerates to returning speed in positive direction.

③: Workbench stops at side of zero signal area, if Z-phase is selected, workbench stops at Z-phase signal.

4. Start position locates between zero signal area and negative limit signal area.

- Near-point signal mode, as show in [Figure 15.15.8](#):

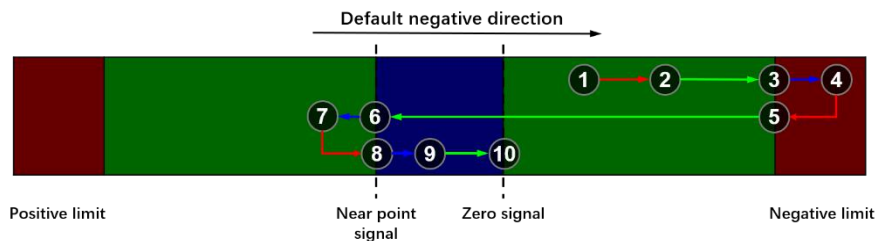


Figure 15.15 8

①→②: Workbench accelerates to returning speed in negative direction.

③→④: Workbench reaches negative limit and decelerates to 0.

④→⑤: Workbench accelerates to returning speed in positive direction.

⑥→⑦: Workbench leaves zero signal area and decelerates to 0.

⑦→⑧: Workbench accelerates to returning speed in negative direction.

⑧→⑨: Workbench enters zero signal area and decelerates to crawl speed in negative direction.

⑩: Workbench stops at side of zero signal area, if Z-phase is selected, workbench stops at Z-phase signal.

- Zero positive-edge mode, as shown in [Figure 15.15.9](#)

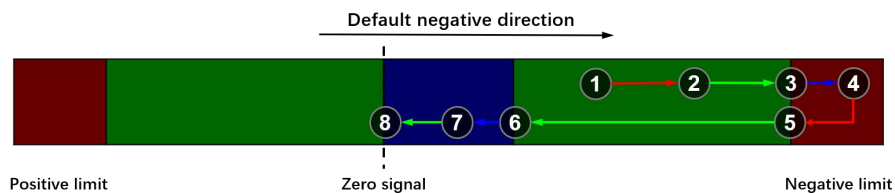


Figure 15.15 9

①→②: Workbench accelerates to returning speed in negative direction.

③→④: Workbench enters negative limit and decelerates to 0.

④→⑤: Workbench accelerates to returning speed in opposite direction.

⑥→⑦: Workbench enters zero signal area and decelerates to crawl speed.

⑧: Workbench stops at side of zero signal area, if Z-phase is selected, workbench stops at Z-phase signal.

5. Start position locates in positive limit signal area.

- Near-point signal mode, as show in [Figure 15.15.10](#):

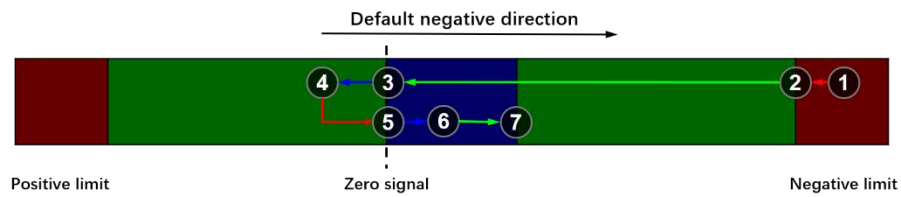


Figure 15.15 10

- ①→②: Workbench accelerates to returning speed in positive direction.
- ③→④: Workbench leaves zero signal area and decelerates to 0.
- ④→⑤: Workbench accelerates to returning speed in negative direction.
- ⑤→⑥: Workbench enters zero signal area and decelerates to crawl speed in negative direction.
- ⑦: Workbench stops at side of zero signal area, if Z-phase is selected, workbench stops at Z-phase signal.

- Zero positive-edge mode, as shown in [Figure 15.15.11](#)

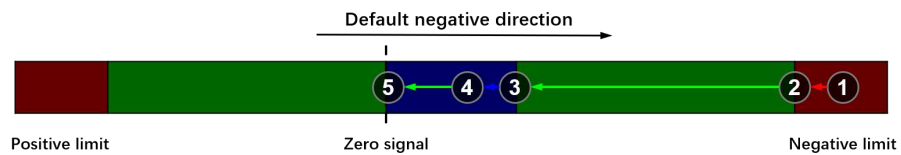


Figure 15.15 11

- ①→②: Workbench accelerates to returning speed in positive direction.
- ③→④: Workbench enters zero signal area and decelerates to crawl speed in positive direction.
- ⑤: Workbench stops at side of zero signal area, if Z-phase is selected, workbench stops at Z-phase signal.

JOG

Instruction introduction

1. This instruction can generate jog pulse with direction by setting frequency and acceleration/deceleration time. when this instruction is enabled, pulse accelerates to specified frequency in uniform acceleration, when this instruction is disabled, the pulse will not stop immediately, but decelerates to 0 in uniform deceleration.
2. This instruction can judge the sign of frequency automatically, and adjust the pulsing direction of Y output points. for example, when frequency is 100kHz the pulsing direction is positive, when frequency is -100kHz the pulsing is negative, there is no need to change pulsing direction manually.
3. This instruction provides symmetric (same acceleration and deceleration time) T-shape and S-shape position control, it is S-shape when control bit located in special function register D8069 is 1, it is T-shape when the bit is 0.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(F)	Frequency of pulse, the unit is Hz.	K/H/D	■ FGs/FGRB/FGRE/FGRS 0 to 200000 ■ FGm 0 to 500000	32-bit unsigned integer
(T)	Acceleration/deceleration time when frequency transits.	K/D	0 to 65535	16-bit unsigned integer
(DIR)	port that output pulse direction.	Y	-	bool
(OUT)	Port that outputs the pulse.	Y	-	bool

Table 15.16. 1

Attention

This instruction outputs pulse with direction, so that the pulse accumulation counter (D8140-D8158) can count in both directions.

Example

Instruction table:

Network 000

LD M0
JOG K100000 K100 Y0 Y4 //JOG start outputting pulse when M0 is ON

Ladder diagram:

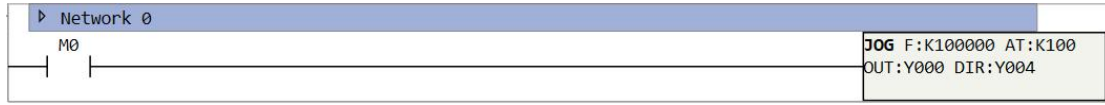


Figure 15.16. 1

FOLLOW

Instruction introduction

This instruction outputs pulse that follows value in corresponding register, the frequency of pulse is decided by how fast the value changes, the counts of pulse is decided by how much the value changes.

Operation introduction

1. At first, refer to ladder diagram of example, set first address of following value D0, output point Y0 and direction point Y12.
2. Value in address D0 (32-byte value) is the target value that pulse follows, in this example the target value is in CV235. What should be noted is that, if the value changes too fast, the corresponding frequency may be beyond limit. Additionally, the follow is only decided by the change of target value, the original value before instruction enabled takes no effects to the follow.
3. Value in address D2 is performance parameter, the value is limited from 0 to 100. When performance parameter is big, motion will follow target strictly with short delay, the follow will be stiff. When performance parameter is small, follow will be slow and not so strict, but more flexible, more suitable to response shock. if there is no need for such strict follow, it is recommended to set this parameter to 50.
4. Value in address D3 is feedforward compensation ratio parameter, the value is limited from 0 to 100. As following actual movements is hysteretic, to keep positional accuracy during process of movements, proper feedforward compensation can be added. If there is no need for high positional accuracy, it is recommended to set this parameter to 0.
5. Value in address D4 is multiplication coefficient, variation of followed value increases by multiplying by this coefficient. Value in address D5 is division coefficient, variation of followed value decreases by dividing this coefficient. These 2 coefficients can be set from 1 to 100. The result after multiplication and division is used as pulsing quantity, and result will be treated as extremum if beyond the extremum.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(ARG)	First address of follow parameters.	D	-	32-bit pointer
(DIR)	Port that outputs direction of pulse	Y	-	bool
(OUT)	Port that outputs the pulse.	Y	-	bool

Table 1.17. 2

Attention

1. This instruction outputs pulse with direction, so that the pulse accumulation counter (**D8140-D8158**) can count in both directions.
2. Target value should not change sharply during following, otherwise frequency of pulse may exceed the maximum frequency limit.

Example

Instruction table:

Network 000

```
LD      M8151
EHCNT  CV235   K2   K0   K9999999 //enable counter when running
```

Network 001

```
LD      M0
MOVD   CV235   D0 //set instruction to follow the counter CV235
MOV    K20     D2 //set performance parameter
MOV    K0      D3 //set feedforward compensation ratio parameter
MOV    K1      D4 //set multiplication coefficient
MOV    K1      D5 //set division coefficient
FOLLOW D0      Y000   Y012 //FOLLOW starts when M0 is ON
```

Ladder diagram:

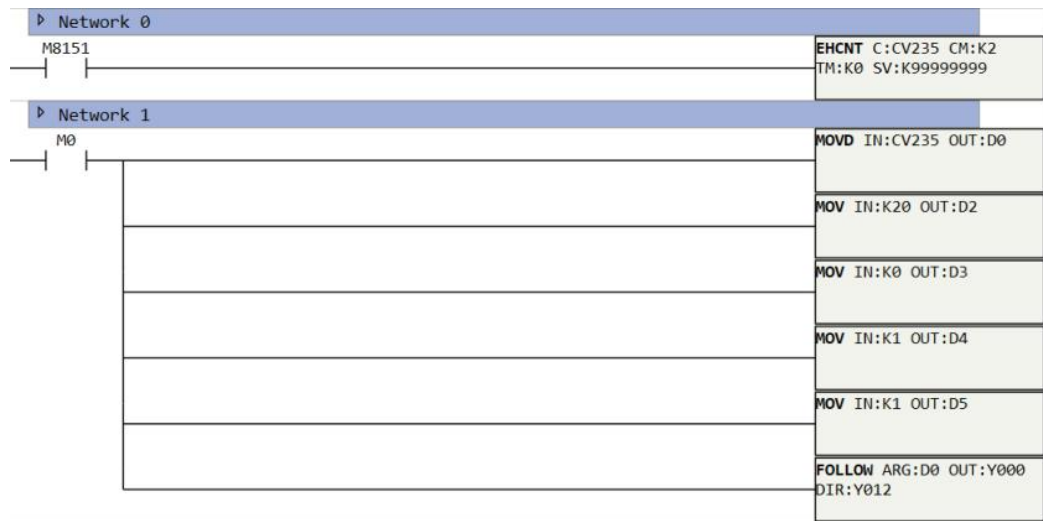


Figure 15.17. 1

ECAM

Instruction introduction

In the material cutting industry, tradition methods mostly use stopping shear and stopping cut, these methods are inefficient and may not meet requirements of material molding process. In the packaging and printing industries that need real-time phase following, tradition methods mostly use Mechanical CAM to realize.

Mechanical CAM and electronic CAM

Refer to [Figure 15.18.1](#) that shows how mechanical CAM work, the component that drives rotation of CAM is input-axis, the component that pushed by CAM is output-axis. every position of input-axis corresponds to one certain position of output-axis, this corresponding relation is decided by the profile of CAM, which can be expressed with a function relation between position of input-axis and output-axis. In actual processing, not all parts of CAM profile play a key role, only some points and segments of CAM profile work in processing, by using these points and segments, combined with appropriate mathematical models, function relation between position of main shaft and driven shaft can be build. On base of the function relation, position of output-axis can be calculated with position of input-axis, so that PLC can output pulse to control output-axis to move to corresponding position as CAM profile does, this is how the electronic CAM works.

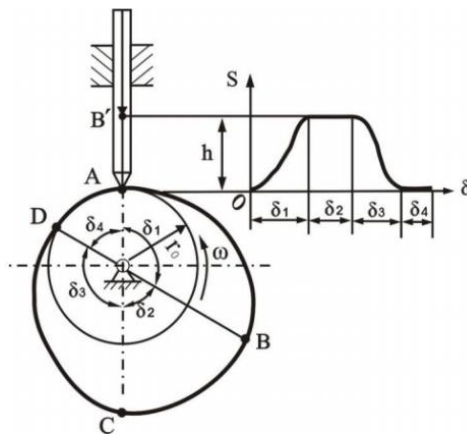


Figure 15.18. 1

Compared with mechanical CAM, electronic CAM is more flexible, more convenient to modify profile, and cheaper to maintain. For the present ECAM instruction has 3 mode: flying saw mode, flying shear mode and universal CAM mode. Among them, flying saw mode and flying shear mode are 2 specific application scenarios of universal CAM, mainly apply to profile cutting, fixed-length paper cutting and other fields. For these 3 modes, ECAM instruction provides interactive graphical interface to set parameters, additionally user can preview and adjust the motion curve of CAM.

Brief introduction of flying saw

flying saw mode of EACM instruction controls driven cutting machine motion to follow input-axis. Cutting machine cuts material by using synchronous area in motion, then slow down to 0 and back to origin point rapidly, waiting for next following motion. This kind of cutting motion can keep the input-axis ongoing and improve efficiency.

As shown in the [Figure 15.18.2](#), modules of flying saw consist of feed drive of input-axis, position feedback of input-axis, linear following of output-axis and cutting module. Usually, servo or transducer constitutes feed drive of input-axis, encoder constitutes position feedback of input-axis that feedbacks current position and velocity of input-axis. Servo with lead screw constitutes linear following of output-axis that drives motion of cutting module. Cutting module mainly driven by hydraulic pressure, pressure system or servo drive. Compared with implanting flying saw program into servo system, using PLC to realize this program will be easier to modify. ECAM instruction makes the control of flying saw more convenient and efficient.

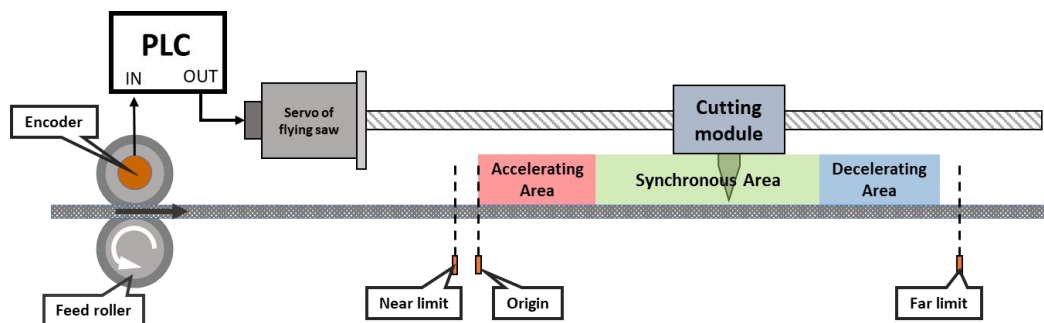


Figure 15.18. 2

Main driving part is driven by servo or transducer, it drives press roller to feed. Position and velocity of input-axis are feedbacked by encoder that connects with PLC input. Servo of flying saw gets pulse signals output from PLC set by flying saw function, and drive cutting module to position located on output-axis correspond to CAM curve, then cutting module cuts according to marks on synchronous area. Origin is used for reset of cutting module, near limit and far limit is used for limitation protection.

flying saw need to configure cutting length, and generate flying saw curve automatically according to curve parameters, mechanical parameters and so on. The curve consists of 4 segments: accelerating area, synchronous area, decelerating area, and reverse area. these 4 segments refer to [Figure 15.18.3](#).

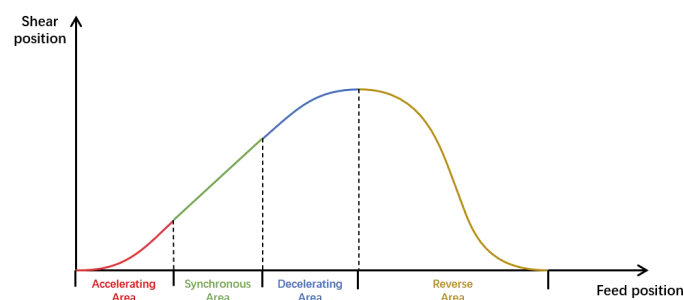


Figure 15.18. 3

In the figure, horizontal axis means position of input-axis, vertical axis means position of output-axis. material length equal to sum of distance that input-axis walks in 4 area, length of synchronous area equals to synchronous velocity (same as velocity of input-axis in stable running) multiplies by cutting time (need to be more than actual cutting time used). What should be noted is, acceleration and deceleration will be more stable if accelerating area and decelerating area is longer. When material length is fixed, longer accelerating area and decelerating area will result in shorter synchronous area, and synchronous area must be long enough for cutting need.

Brief introduction of flying shear

Same as flying saw, flying shear keep output-axis and input-axis synchronous when cutting to make cutting ongoing and efficient. What is different is the motion mode of output-axis, for flying saw it is translational motion, for flying shear it is rotational motion.

As shown in [Figure 15.18.4](#), module of flying shear consists of feed drive of input-axis, position feedback of input-axis and cutting wheel. Usually the cutting wheel is driven by servo.



Figure 15.18. 4

As shown in [Figure 15.18.5](#), motion area of flying shear is divided into 3 parts: accelerating area, synchronous area, and decelerating area. As the rolling motion is a circulation, there is no reverse area.

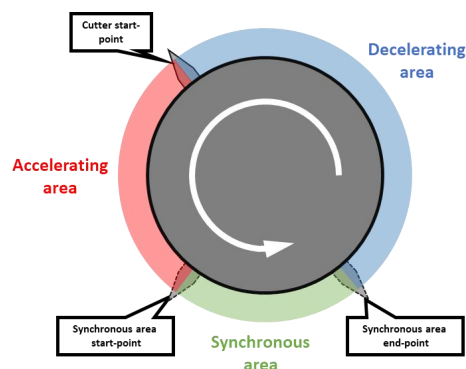


Figure 15.18. 5

To generate flying shear curve, user need to configure cutting length, angle of accelerating area, angle of synchronous area, number of cutters and so on. the curve refers to [Figure 15.18.6](#).

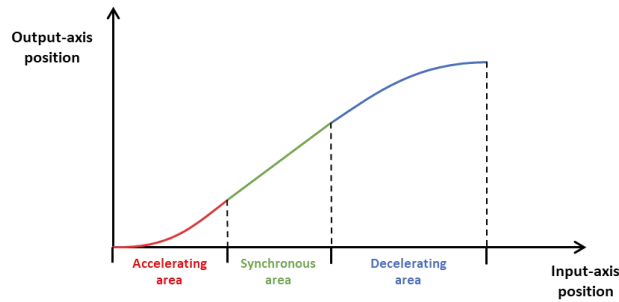


Figure 15.18. 6

In the Figure, horizontal axis means position of input-axis, vertical axis means angle position of output-axis.

Brief introduction of general CAM

Flying saw and flying shear is common application of CAM. For more wise and flexible use, there is general CAM that can generate motion curve by key-joints.

Shown as [Figure 15.18.7](#), user can set any target key-joints (consist of X/Y coordinates and slope), system will generate motion curve of CAM.

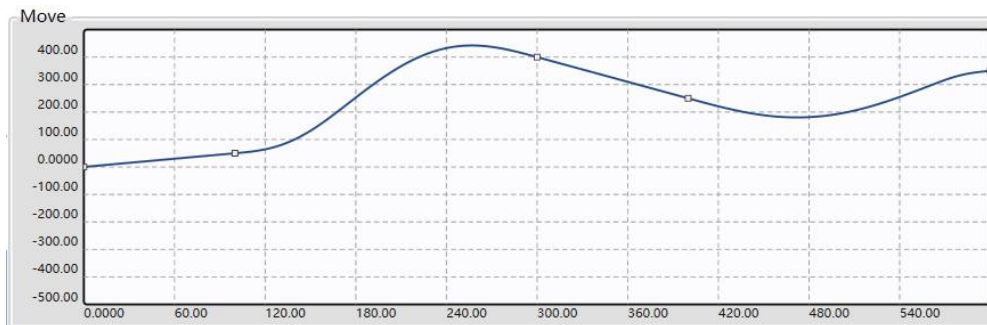


Figure 15.18. 7

Setting data

Parameters of ECAM can be divided into 2 parts, one part is general parameters, and another part is parameters in different modes. ECAM will not run if parameters are set illegal, and the error code register **D8176** will show corresponding error code 46. Possible cases of illegal parameters refer to annotations.

P.S.

- INT32 means 32-bit signed integer, UINT32 means 32-bit unsigned integer, FP32 means 32-bit floating-point number, INT16 means 16-bit signed integer, UINT16 means 16-bit unsigned integer, bool means flag bit.
- R means read only, W means write only, R/W means readable and writable.
- X, Y is input/output register, D is double word register, K is constant, CV is high-speed counter register, M is internal bit register.

Port configuration and general parameter

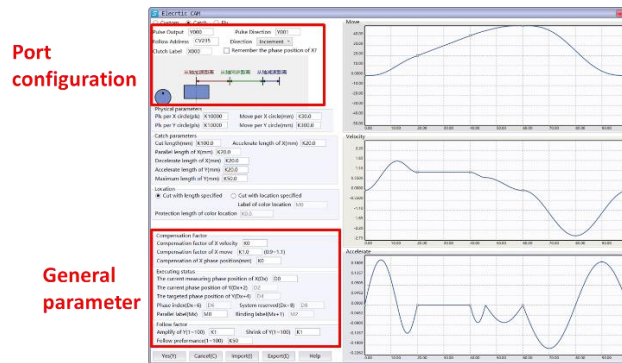


Figure 15.18. 8

Figure 15.18.8 shows the interface of parameter configuration, Table 15.18.1 shows brief information of corresponding parameters. Among parameters, general parameters are divided into 3 types: running status parameter, following factor, and compensation factor.

	Parameter type	Parameter name	Data type	Operand	Default	Range	Unit
Port configuration	Pulse related	Pulse output		Y			
		Pulse direction		Y			
	Following related	Following target		D/CV			
		Count direction			Incremental		
	Clutch related	Clutch flag-bit		X/M			
Else	Remembering phase of input-axis			No			
General parameter	Running status parameter	Measured phase position of input-axis	INT32 W/R (Readable when running, writable when disabled)	D			pls
		Current input-axis phase position	INT32 W/R	D			pls
		Target input-axis phase position	INT32 W/R	D			pls
		Current segment number	UNIT32 W/R	D			
		System reserved	INT32 W/R	D			
		Synchronization flag-bit	bool W/R	M			
	Following factor	Multiplication coefficient	INT32 W	D/K	1	1~100	
		Division coefficient	INT32 W	D/K	1	1~100	
		Following performance	INT32 W	D/K	50	1~100	
	compensation factor	Input-axis velocity compensation factor	FP32 W		0		
		Input-axis displacement compensation factor	FP23 W		1	0.9~1.1	
		Input-axis phase compensation factor	FP32 W		0		mm

Table 15.18. 1

- **Pulse output:**
Y port of PLC that outputs pulse.
- **Pulse direction:**
Y port of PLC that output pulse direction.
- **Address of following target:**
Register that record phase position of input-axis, can be set with D or CV register.
- **Counting direction:**
Direction of counting of following target, can be set incremental or decremental
- **Clutch flag-bit:**
Flag-bit that control clutch, can be set with X or M register (support X port of IO expansion board, not support X port of extended module, additionally the X port need to be interference-proof during certain filtering time).
- **Remembering phase position of input-axis:**
Whether to remember phase position of input-axis, if remember, running will restart from phase position where ECAM disable; If not, running will restart from origin.
- **Measured phase position of input-axis:**
Phase position that input-axis locates in one cycle, recount at begin of new cycle. For flying saw/shear, phase position means feedback pulse count of corresponding encoder during one order length cut; For general CAM, phase position means pulse count of corresponding encoder from first key-joint to last.
- **Current output-axis phase position:**
Phase position that output-axis locates in one cycle, recount at begin of new cycle. For flying saw and general CAM, phase position may be same in different area(segment), it should work with current segment number; For flying shear, phase position means phase position means feedback pulse count of corresponding encoder during one order length cut.
- **Target output-axis phase position:**
Phase position of output-axis that input-axis corresponds to in CAM curve, it is the target that output-axis follow with.
- **Current segment number:**
Current segment number of whole running cycle.
- **System reserved:**
Reserved registers of system that record data, occupy three D registers.
- **Synchronization flag-bit:**
Flag-bit that shows if output-axis is in synchronization area, 1 means in, 0 means not.
- **Binding flag-bit:**
Flag-bit that controls and shows if output-axis is binding with input-output-axis, 1 means binding, 0 means not.
- **Multiplication coefficient:**
Coefficient that amplifies phase position of output-axis proportionally.
- **Division coefficient:**
Coefficient that shrink phase position of output-axis proportionally.
- **Following performance:**
Parameter that adjusts stiffness of following, bigger the parameter is, closer the

following will be.

- **Input-axis velocity compensation factor:**
Compensation factor that compensates velocity error caused by time delay of calculating. System will set this factor automatically.
- **Input-axis displacement compensation factor:**
Compensation factor that compensates accumulative displacement error caused by residual of algorithm or offset of mechanical parameters. It can be set as default in fixed-mark mode.
- **Input-axis phase compensation factor:**
Compensation factor that compensates lead or lag of output-axis phase position. It also can be used for adjusting cut position.

Flying saw parameter

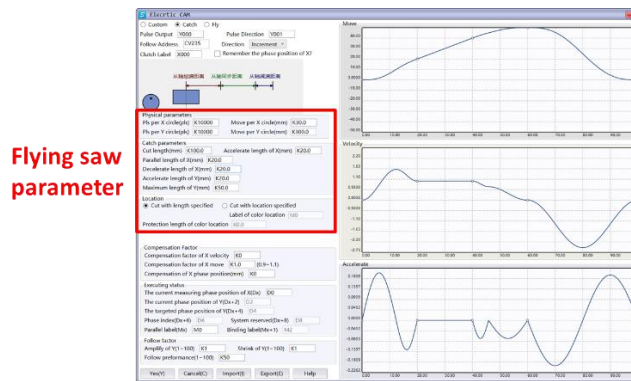


Figure 15.18. 9

Figure 15.18.9 shows the interface of flying saw parameter configuration, Table 15.18.2 shows brief information of corresponding parameters.

	Parameter type	Parameter name	Data type	Operand	Default	Range	Unit	
Flying saw parameter	Mechanical parameter	Input-axis pulse count pre cycle	UINT32 W	D/K		>0	pls	
		Input-axis motion length per cycle	FP32 W	D/K		>0	mm	
		Output-axis pulse count pre cycle	UINT32 W	D/K		>0	pls	
		Output-axis motion length per cycle	FP32 W	D/K		>0	mm	
	CAM mode	Fixed-mark setting						
		Flag-bit of color-mark			X/M			
		Protection distance of color-mark	FP32 W	D/K		0	>=0	mm
	Flying saw curve parameter	Order cutting length	FP32 W	D/K			>0	mm
		Input-axis accelerating distance	FP32 W	D/K			>0	mm
		Input-axis synchronous distance	FP32 W	D/K			>0	mm
		Input-axis decelerating distance	FP32 W	D/K			>0	mm
Output-axis accelerating distance		FP32 W	D/K			>0	mm	
Output-axis limit distance		FP32 W	D/K			>0	mm	

Table 15.18. 2

- **Input-axis pulse count pre cycle:**
Pulse count of input-axis servo outputting or encoder feedback each cycle.

- **Input-axis motion length per cycle:**
Actual motion distance of materials during one cycle
- **Output-axis pulse count pre cycle:**
Pulse count of output-axis servo outputting or encoder feedback each cycle.
- **Output-axis motion length per cycle:**
Actual motion of cutting-module during one cycle
- **Fixed-mark setting:**
Mode select of cutting, fixed-length mode or fixed-mark mode
- **Flag-bit of color-mark:**
Input of mark detection signal (support X port of IO expansion board, not support X port of extended module).
- **Protection distance of color-mark:**
In fixed-mark mode, after one color-mark is detected, system will not detect next until output-axis run a distance combined with order cutting length and protection distance.
- **Order cutting length:**
In fixed-length mode, order cutting length is actual cutting length; In fixed-mark mode, order cutting length is minimum of actual cutting length.
- **Input-axis accelerating distance:**
Length of input-axis accelerating area.
- **Input-axis synchronous distance:**
Length of input-axis synchronous area.
- **Input-axis decelerating distance:**
Length of input-axis synchronous area.
- **Output-axis accelerating distance:**
Length of output-axis accelerating area.
- **Output-axis limit distance:**
Length of output-axis whole motion area.

Flying shear parameter

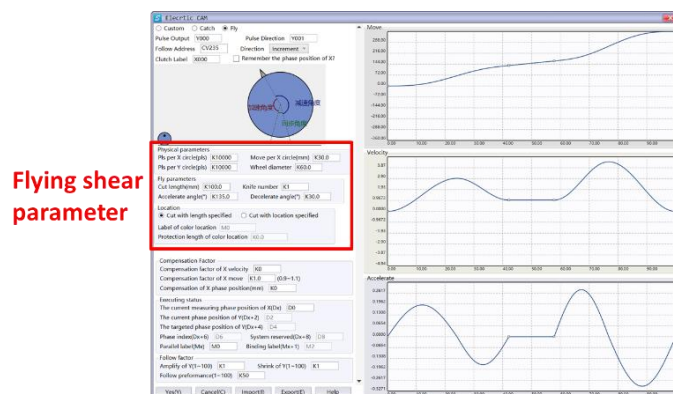


Figure 15.18. 10

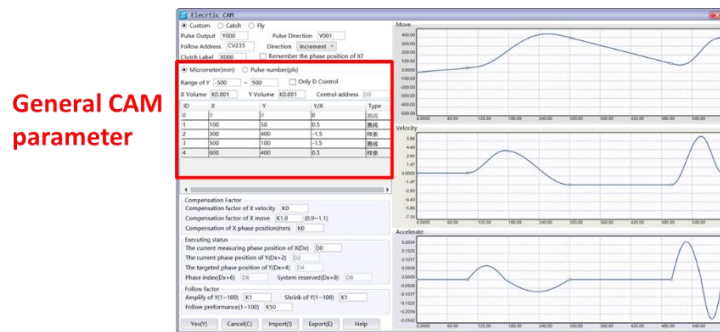
Figure 15.18.10 shows the interface of flying shear parameter configuration, Table 15.18.3 shows brief information of corresponding parameters.

	Parameter type	Parameter name	Data type	Operand	Default	Range	Unit	
Flying shear parameter	Mechanical parameter	Input-axis pulse count pre cycle	UINT32 W	D/K		>0	pls	
		Input-axis motion length per cycle	FP32 W	D/K		>0	mm	
		Output-axis pulse count pre cycle	UINT32 W	D/K		>0	pls	
		Output-axis motion length per cycle	FP32 W	D/K		>0	mm	
	CAM mode	Fixed-mark setting						
		Flag-bit of color-mark		X/M				
		Protection distance of color-mark	FP32 W	D/K	0	>=0	mm	
	Flying shear curve parameter	Order cutting length	FP32 W	D/K		>0	mm	
		Number of cutters (set as i)	FP32 W	D/K		>0		
		Output-axis synchronous angle	FP32 W	D/K		0~360/i	°	
Output-axis accelerating angle		FP32 W	D/K		0~360/i	°		

Table 15.18. 3

- **Output-axis synchronous angle:**
Angle of output-axis accelerating area.
- **Output-axis accelerating angle:**
Angle of output-axis whole motion area.

General CAM parameter



General CAM parameter

Figure 15.18. 11

Figure 15.18.11 shows the interface of general CAM parameter configuration, Table 15.18.4 shows brief information of corresponding parameters.

	Parameter type	Parameter name	Data type	Operand	Default	Range	Unit
General CAM parameter	Mechanical parameter	Unit					
		Range of output-axis					
		Pulse equivalent of input-axis	FP32 W	D/K	0.01	>0	mm/pls
		Pulse equivalent of output-axis	FP32 W	D/K	0.01	>0	mm/pls

Key joint parameter	Input-axis coordinate		D/K			
	Output-axis coordinate		D/K			
	target slope		D/K			
	segment type					
Else	Control mode					
	Start address of control registers					

Table 15.18. 4

- **Unit:**
Unit of key joint coordinate, can be set with mm (millimeter) or pls (pulse count).
- **Ranged of output-axis:**
Display range of output-axis in the CAM curve interface.
- **Pulse equivalent of input-axis:**
Distance that input-axis motions per pulse.
- **Pulse equivalent of output-axis:**
Distance that output-axis motions per pulse.
- **Input-axis coordinate:**
X coordinate of key joint on CAM curve.
- **Output-axis coordinate:**
Y coordinate of key joint on CAM curve.
- **Input-axis coordinate:**
X coordinate of key joint on CAM curve.
- **Target slope:**
slope in key joint on CAM curve.
- **Segment type:**
Curve type of segment, can be set with straight line or spline.
- **Control mode:**
Whether to map curve parameters to D registers.
- **Start address of control registers:**
When control mode is selected, user can set the start address of mapped D-registers.

Curve preview area



Figure 15.18. 12

As shown in [Figure 15.18.12](#), the X axis means displacement of input-axis, the Y axis in there curve, means displacement, velocity, acceleration of output-axis in order. User can input parameters and drag joint in preview area to adjust the curve. Curve should be set smooth,

avoid excessive velocity and acceleration.

Annotation

Fixed-mark mode

Motion cycle starts from origin when color mark is detected (flag-bit of color-mark is ON). If distance between 2 color-marks is longer than order cutting length, the actual cutting length will be distance between marks; If distance between color-marks is equal to order cutting length, the actual cutting length will be order cutting distance; If distance between color marks is shorter than order cutting distance, the detection of second color-mark will take no effect, but take effect until distance between next mark and start mark is longer than order cutting distance, in this case, there may be one or more invalid color-mark signal between 2 valid color-marks.

Rationality of parameter setting

All the parameter setting must not be beyond data range, also they should meet specific relationships, or there will be error report with error code 46 in D8176.

For flying saw, we define length of input-axis accelerating area as A_1 , length of input-axis synchronous area as S , length of input-axis decelerating area as D_1 , order cutting length as L_1 , distance of output-axis accelerating area as A_2 , output-axis limit length as L_2 , and these parameters should meet following relationships (diagrammatic diagram refer to Figure 15.18.1):

$$A_1 + S + D_1 < L_1$$
$$A_2 + S < L_2$$

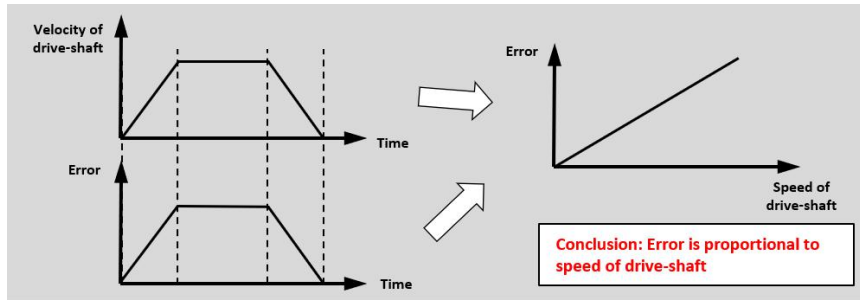
For flying shear, we define angle of output-axis accelerating area as α , angle of output-axis synchronous area as β , number of cutters as K , and these parameters should meet following relationships (diagrammatic diagram refer to Figure 15.18.1):

$$\alpha + \beta < (360^\circ / K)$$

Three kinds of error compensation factor

◆ Input-axis velocity compensation factor

Description: this compensation factor is used for compensating error in proportion to input-axis velocity. During acceleration, if velocity is higher, error is bigger; During deceleration, it is opposite; During uniform motion, the error stays stable. The relationship between error and velocity refer to [Figure 15.18.13](#).



Error symptom shows in finished workpieces

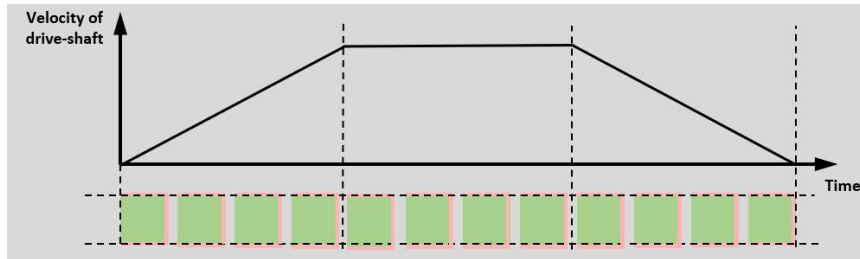


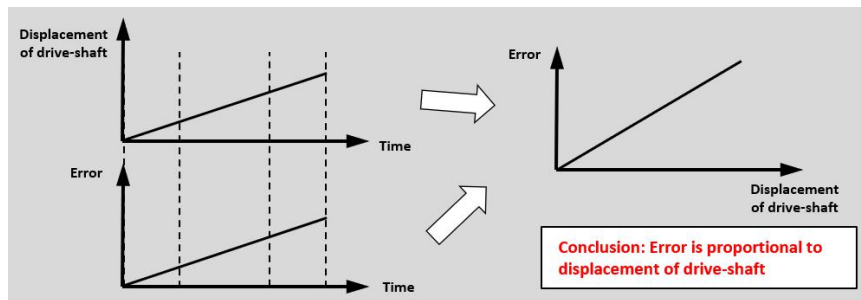
Figure 15.18. 13

Cause: because of feature of ECAM, the error comes from delay caused by operation process.

Solution: the program will give an appropriate algorithm delay time T , along with velocity of input-axis F , the compensation Δ equals to $T \cdot F$ can be calculated. Add the compensation to position of input-axis that measured each time, so that the error can be compensated. What should be noted is, meaning of input-axis velocity compensation factor is offset adjustment of T , it can be set 0 in most cases, and be slightly adjusted when needed.

◆ Input-axis displacement compensation factor

Description: this compensation factor is used for compensating error in proportion to input-axis displacement. the error will increase with input-axis motion. The relationship between error and displacement refer to [Figure 15.18.14](#).



Error symptom shows in finished workpieces

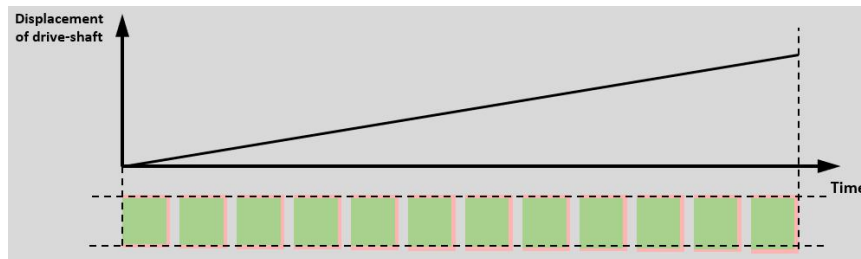


Figure 15.18. 14

Cause: there is differences between actual mechanical structure and theoretical model. For example, advance length L during a round of rolling wheel equals to product of rolling wheel diameter R and π . While π is an irrational number, in actual calculation π will be used as an approximate value. If π is rounded off to 3.14159 and D is 100m, there will be error of 0.000265358979mm in each round, and be accumulated to 2.65368979mm in 10000 cycles.

Solution: As the error is proportional to whole motion distance, we can get the proportional relationship K in actual running process, and compensation($K*L$) in each round to compensate error. What should be noted is, in fixed-length mode the factor is very closed to 1; In fixed-mark mode, the factor is 1 by default, and can be ignored if there is no need.

◆ Input-axis phase compensation factor

Description: this compensation factor is used for compensating error in proportion to input-axis phase. The error has no relationship with velocity and displacement of input-axis, it comes from phase bias in first cut. The error symptom refers to [Figure 15.18.15](#).

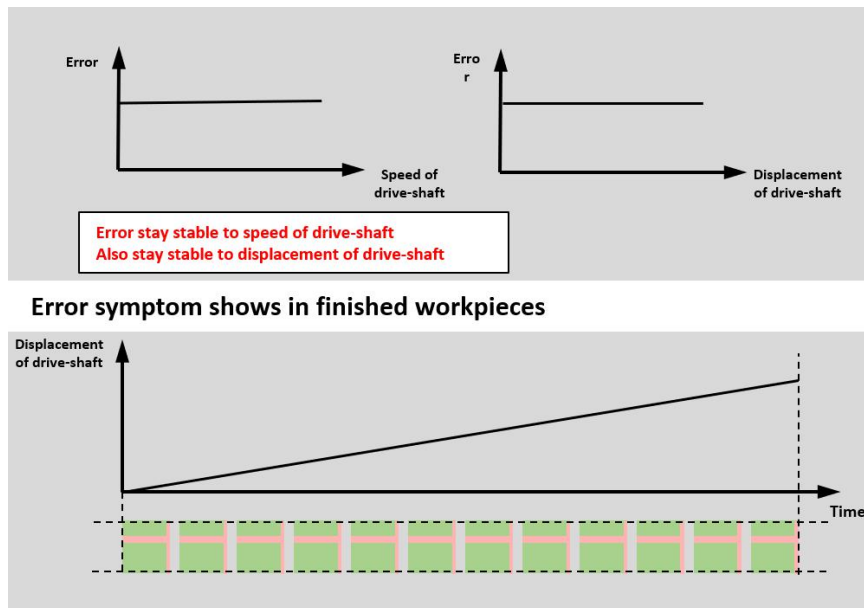


Figure 15.18. 15

Cause: There is phase deviation between input-axis and output-axis, it always happens in first running.

Solution: Add an initial phase compensation in running process of input-axis to compensate the error. Refer to [Figure 15.18.16](#), if the compensation factor is the deviation in first cut, if it is plus the cut will be advanced, if it is minus the cut will be lagging.

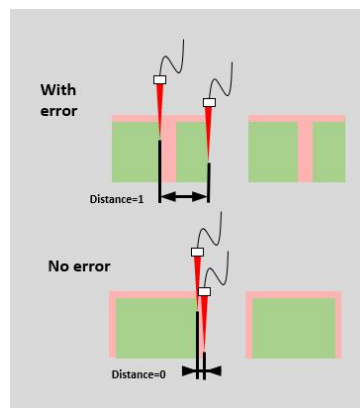


Figure 15.18. 16

Parameter setting of general ECAM

The parameters of general ECAM should meet some specific rules, and may be adjusted automatically if they are illegal:

1. If the target value counting direction of input-axis is incremental, input-axis coordinate of current segment must be bigger than input-axis coordinate last segment.
2. If Y coordinate of two adjacent key joint is same, slope of them will be adjust to 0 automatically.
3. If curve type of current segment is straight line, slope and output-axis coordinate must

be matched, or they will be adjusted automatically.

4. If curve type of current segment is straight line and curve type of last segment is splines, the slope of last segment will be adjusted to be same with current segment to ensure smooth transition.

Clutch

Flag-bit of clutch can be set as X or M register. When flag-bit of clutch is set, binding between input-axis and output-axis will break without disabling instruction, in this way the motion of output-axis can be controlled by other pulsing instructions. When flag-bit of clutch is reset, running will continue from binding break phase (if remember) of input-axis and output-axis.

What should be noted is, when clutch is set, if not especially necessary, it is recommended to keep input-axis motionless during clutch is ON, otherwise the material length of first cut after resetting clutch will have a corresponding offset. If output-axis is being controlled by other pulsing instruction, resetting flag-bit must be after disabling the instruction, otherwise the binding between input-axis and output-axis will not recover, if output-axis position gets deviation from clutch ON to clutch OFF, there will be deviation of whole output-axis travel, for flying saw it may cause overrun. If disable instruction when clutch is ON without remembering input-axis phase, input-axis will restart running from origin, and this may also cause deviation of whole output-axis travel.

Running status parameters

Running parameters are stored in continuous double-word D registers and M bit-registers. If there is need to record the running position when instruction is disabled, user can distribute these registers into retentive zone, so that ECAM can restart running from position recorded in these registers automatically but not the origin. User can also reset these registers in ladder diagram to restart ECAM from origin without remembering the input-axis phase. Additionally, these registers are read only when ECAM instruction enabled, readable and writable when ECAM instruction disabled, so the reset operation can only take effect when ECAM instruction disabled.

Application

This chapter mainly introduces steps of using ECAM instruction:

1. Check if PLC pulsing port is well worked, servo is well worked, and ensure correct wiring.
2. Configure pulsing port, and address of target value. Confirm target value of input-axis forward motion is increment or decrement. Configure flag-bit of clutch by connecting clutch signal to X port or mapping to M register. Confirm if it is needed to remember current phase of input and output axis, if needed the running status parameters should be set into retentive zone, so that ECAM instruction can restart from last position where

it stops after power off.

3. Configure mechanical parameters of ECAM, such as pulsing number and forward distance of each round. These parameters must be as accurate as possible, or it may influence cutting length or synchronous area.
4. Select fixed-mark mode of fixed-length mode according to actual case, and set parameters of flying saw or flying shear:

◆ Flying saw

On the premise that parameter setting is rational, purpose of parameter setting is making cutting module accelerating stable, and more space for synchronous area to complete cutting. [Figure 15.18.17](#) shows the flying saw parameters.

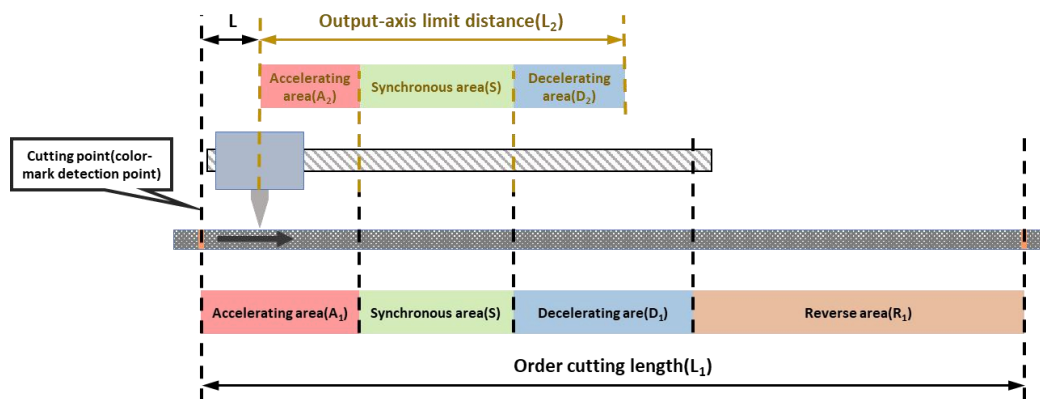


Figure 15.18. 17

Set input-axis motion distance during accelerating area as A_1 , output-axis motion distance in accelerating area as A_2 , distance between position of input-axis and position of output-axis in origin as L , and $L = |A_1 - A_2|$. With L , the cutting point can be certain (it is highly recommended to set A_1 more than A_2 , or output-axis will decelerate in accelerating area, which is not stable and energy-efficient).

If fixed-length mode is selected, according to length L , user should ensure material is long enough to meet first cut.

If fixed-mark mode is selected, when output-axis is in origin, if the color mark is in detection point, the marked point will be cutting point. User can also set some offset on mark to offset cutting point according to actual case.

◆ Flying shear

On the premise that parameter setting is rational, purpose of parameter setting is making rolling-wheel accelerating stable, and more space for synchronous area to complete cutting. [Figure 15.18.18](#) shows the flying shear parameters.

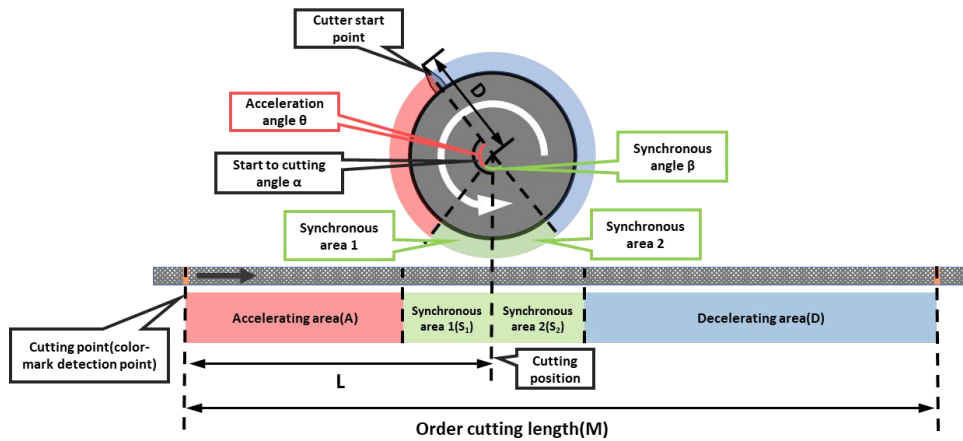


Figure 15.18. 18

In the figure, α is angle between cutter start position to cutting position, β is angle of output-axis synchronous area, θ is angle of output-axis accelerating area. M is order cutting length, A is length of input-axis accelerating area, S_1 is length of input-axis synchronous area before cutting position, L is distance between input-axis origin and cutting position. D is diameter of cutter, i is number of cutters. These parameters should meet following formula:

- a) $A = M\theta / (360^\circ / i)$
- b) $S_1 = ((\alpha - \theta) / 360^\circ) \pi D$, ($\alpha > \theta$)
- c) $L = A + S_1$
- d) $\theta + \beta < (360^\circ / i)$, ($\theta < \alpha - \theta < \beta$)

If fixed-length mode is selected, set appropriate β and θ according to formula (d), to ensure stable acceleration. Get L according to formula (a)(b)(c), so that user can ensure material being long enough to meet first cut.

If fixed-mark mode is selected, mark detection point is always set in cutting point. User can also set some offset on mark to offset cutting point according to actual case.

5. Set appropriate following factor according to actual case and need, the details refer to FOLLOW instruction.
6. Write manual program to drive input/output axis motion, and to reset output axis to origin.
7. Do test run after setting is completed, observe the speed synchronization in the synchronous area. If synchronization error is small, trial cut can start, if synchronization error is big, users need to check if parameter setting is right.
8. After trial cutting, cutting length might be differ from the set length, the error may be caused by mechanical parameter inconsistent with real situation. User can measure the running length during rounds of certain numbers, calculate and reset the mechanical parameters.

Attention

1. Parameter modification does not take effect during instruction running, but after

- instruction restarts (enable after disable).
2. If parameters are set unreasonable, after enabling, the instruction will not run, but report an error. User need to check and modify these parameters and restart the instruction.
 3. User can use flag of synchronous area to trigger the cutting, and use falling-edge of synchronous area flag signal to count.
 4. It is better to keep the acceleration/deceleration stable, otherwise it may cause vibration of mechanical structure and influence the measurement of encoder, even overrun error because of excessive velocity.
 5. Zero returning of flying-saw and flying-shear is different, zero returning of flying-shear need to select returning path (clockwise or anticlockwise) to avoid collision.
 6. Do not clear or modify the target value register when instruction is running, otherwise it is most likely to cause overrun error because of abrupt change of input-axis target value.
 7. Color-mark detection will not take effect when clutch is ON. Before clutch being OFF, all the instruction that drive the output-axis must be disabled.
 8. If there is need to restart the CAM curve and remembering input-axis phase is set, user can disable ECAM instruction, clear all the running status parameters and restart it.

EDRVI/EDRVA

Instruction introduction

1. These instructions outputs pulse with adjustable frequency and target pulse number. For **EDRVI**, adjusting of frequency and target pulse number is relative; For **EDRVA**, adjusting of frequency and target pulse number is absolute.
2. When the instruction is enabled, it will accelerate to set frequency in set acceleration time, decelerate to 0 in set deceleration time, and reach set target pulse number in the outputting process. When the instruction is disabled, pulse outputting will stop instantly. If user modifies frequency and target pulse number during outputting, the outputting will adjust to the modified value with acceleration/deceleration.
4. This instruction can be controlled by special function register **M8068**, when **M8068** is ON, direction port output will reverse.
5. This instruction can be set curve mode by special function register **M8069**, when **M8069** is ON, the frequency curve is S-shape type, when it is OFF, the frequency curve is T-shape type.
6. When target pulse number is set 0x7FFFFFFF, pulse will keep outputting in positive direction; When target pulse number is set 0x80000000, pulse will keep outputting in negative direction. If instruction start with these 2 kinds of target pulse number setting, adjusting of target pulse number during outputting will take no effect.
7. When frequency is set to 0 during outputting, pulse outputting will stop instantly without deceleration, and continue to output with acceleration when frequency is set a new nonzero value.

Operation introduction

1. Set frequency, target pulse number, acceleration time, and deceleration time, then enable the instruction. If do no adjusting, this instruction works like **EDRV**. If adjust frequency or target pulse number parameter stored in corresponding D-registers, in next period when detect the change of parameters system will recalculate the frequency curve, additionally adjusting of target pulse number may cause reversal of motor, details will be introduced in later parts. Adjusting take effects as long as the instruction is enabled, despite pulse outputting being over. Additionally, adjusting of acceleration/deceleration time during pulse outputting takes no effect.
2. As shown in [Figure 15.19.1](#) and [Figure 15.19.2](#), dotted-line is curve without adjusting, solid-line is curve with adjusting. In [Figure 15.19.1](#), frequency is adjusted in time t2 from F1 to F2 when frequency is stable, in [Figure 15.19.2](#), frequency is adjusted to F2 in time t1 when frequency is accelerating. In both case, pulse outputting still reaches the target pulse number as case without adjusting.

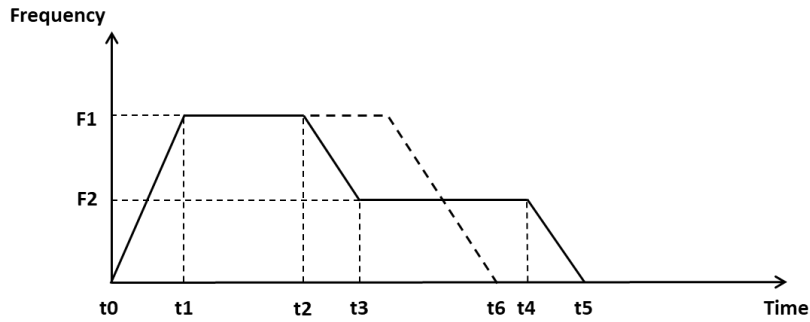


Figure 15.19. 1

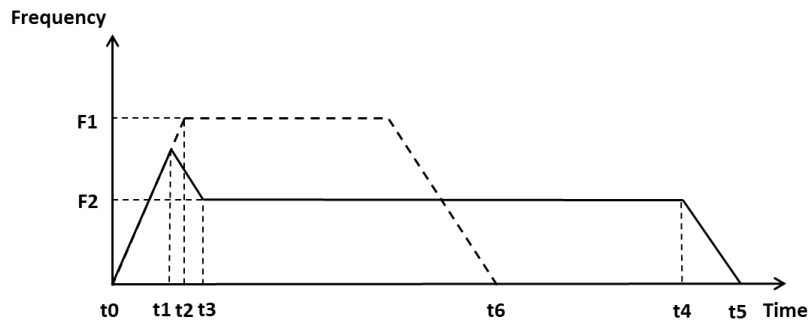


Figure 15.19. 2

- Adjusting of target pulse number is used for position adjusting. As shown in [Figure 15.19.3](#), P0 is origin, P1 is target position without adjusting target pulse number, P2 is target position with adjusting target pulse number, and P2 is after P1. System will adjust frequency to reach the new adjusted target pulse number, [Figure 15.19.4](#) shows 3 kinds of case (black line means adjusting during stable frequency, red line means adjusting during deceleration, blue line means adjusting after pulse outputting is over).

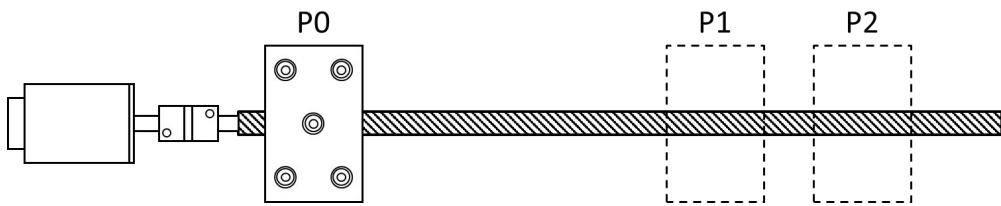


Figure 15.19. 3

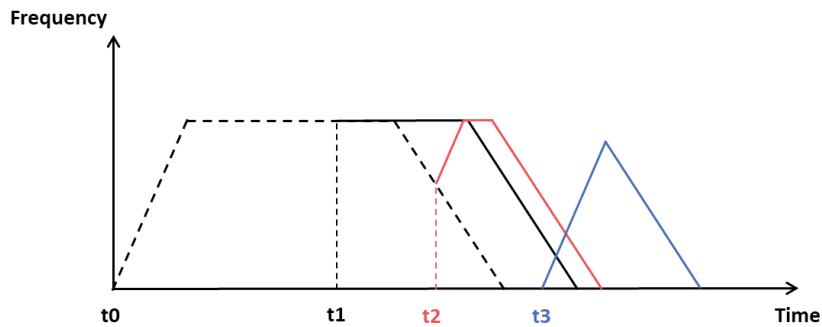


Figure 15.19. 4

[Figure 15.19.5](#) and [Figure 15.19.6](#) shows case that P2 is before P1.

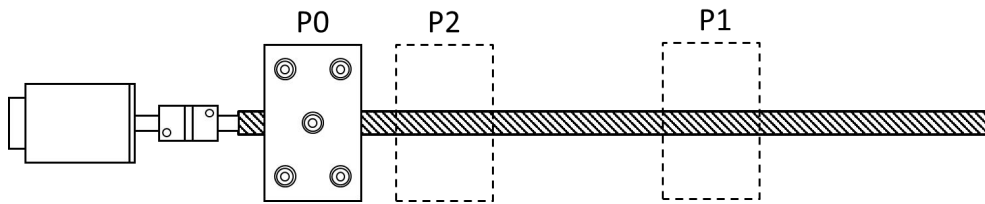


Figure 15.19. 5

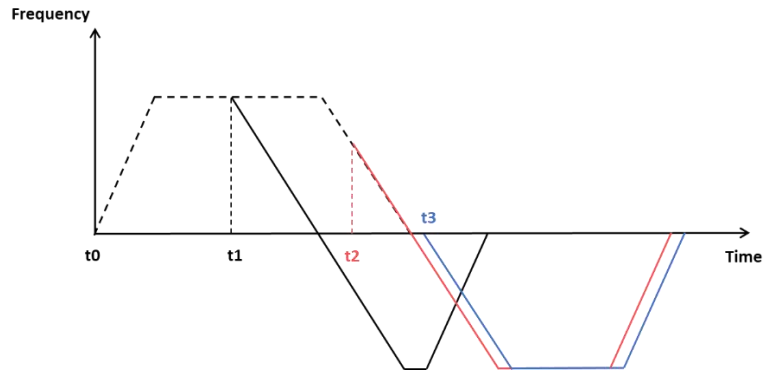


Figure 15.19. 6

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(F)	Frequency of pulse outputting that is adjustable.	K/H/D	■ FGs/FGRB/FGRE/FGRS 0~200000 ■ FGm 0~500000	32-bit unsigned integer
(P)	Target pulse number of pulse outputting that is adjustable	K/H/D	0~2147483647	32-bit unsigned integer
(A)	Acceleration time of pulse outputting	K/H/D	0~65535	16-bit unsigned integer
(D)	Deceleration time of pulse outputting	K/H/D	0~65535	16-bit unsigned integer
(DIR)	Port that outputs direction of pulse	Y	-	bool
(OUT)	Port that outputs the pulse.	Y	-	bool

Table 15.19. 1

Attention

This instruction outputs pulse with direction, so that the pulse accumulation counter (**D8140-D8158**) can count in both directions.

High-speed Count Instructions

Notes

- 1) High-speed count instructions are designed to count high-speed pulse, and there are specific CV registers (**CV235** to **CV255**) are used for high-speed pulse count.
- 2) there are three kind of counting modes for high-speed instructions: single-phase counting, double-phase counting, and AB-phase counting:
 - Single-phase counting counts pulse of one port in one direction (positive or negative).
 - Double-phase counter counts pulse of two ports in two directions (positive and negative). Details refer to [Figure 16.1.1](#).

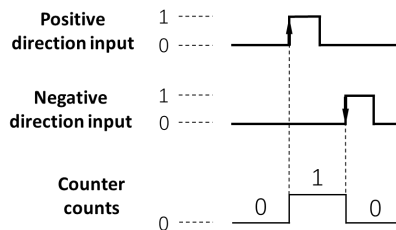


Figure 16.1. 1

- AB-phase counting counts pulse of A phase and B phase of motor, and there are 2 modes for AB-phase count: single frequency multiplication mode and quadruple frequency multiplication mode. Details of 2 modes refer to [Figure 16.1.2](#) and [Figure 16.1.3](#).

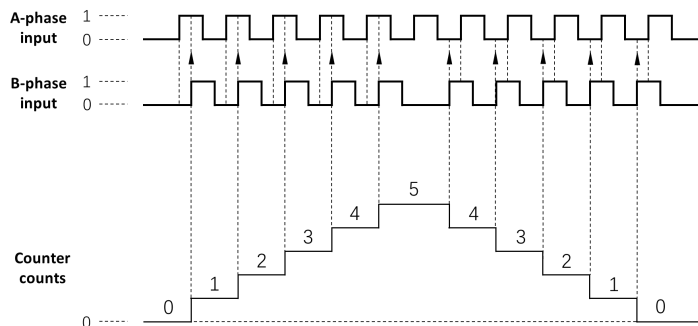


Figure 16.1. 2

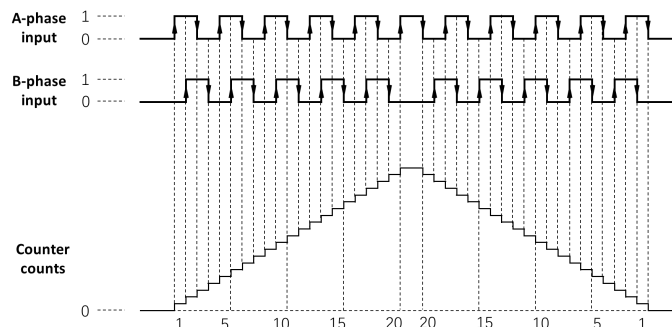


Figure 16.1. 3

HCNT

Instruction introduction

1. This instruction supports most 4 pulse input ports, so that **FGm** series don't support this instruction. For series that only support 2 high-speed pulse ports (**FGs_16MT/FGs_32MT/FGRB_C8X8T**), only **X0** and **X1** can be used for high-speed counting.
2. High speed counters are assigned with specific counting mode in this instruction, details refer to [Table 16.2.1](#) (in the Table, U means counting upward, N means counting downward, A means A-phase, B means B-phase).

	High-speed counter register	High-speed pulse input port			
		X000	X001	X002	X003
Single-phase	CV235	U			
	CV236		U		
	CV237			U	
	CV238				U
Double-phase	CV246	U	D		
	CV248			U	D
AB-phase	CV251	A	B		
	CV252			A	B

Table 16.2. 1

3. Range of count is -2,147,483,648 to 2,147,483,647, when count is out of range, there will be overflow or underflow.
4. Counting mode of AB-phase counting is control by special function registers **M8051** and **M8052**, they control counting mode of **CV251** and **CV252**. When control bit is 0, it is single frequency multiplication mode; When control bit is 1, it is quadruple frequency multiplication mode.

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(C)	Address of selected counter register.	CV	-	32-bit pointer
(SV)	Target number of corresponding counter.	K/H/D	-2147483648~2147483647	32-bit signed integer

Table 16.2. 2

Attention

1. When connect input to port, to improve anti-interference, it is recommended to add shielding lines which on the input line, and connect shielding line to ground.
2. When one high-speed counter is used, the input port it occupies can't be used for other counter, also it can't be applied to common input use.
3. AB-phase counting mode need to be set before instruction running, modify during running takes no effect.

Example

1. Single-phase counting

Instruction table:

Network 000

LD	M0	
HCNT	CV235	K1000 //when M0 is ON, CV235 begin to count pulse of X0 input, when pulse count reach 1000, C235 will be set

Network 001

LD	C235	
OUT	Y000	//when pulse count reach 1000, Y0 will be ON

Network 002

LDP	M1	
RST	C235	K1 //when M1 is ON, reset C235 and CV235

Ladder diagram:

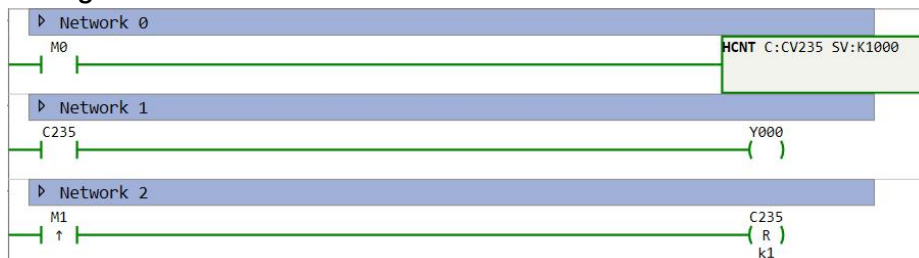


Figure 16.1. 4

2. Double-phase counting

Instruction table:

Network 000

LD	M0	
HCNT	CV246	K1000 //when M0 is ON, CV246 begin to count pulse of X0 input in positive direction, and X1 in negative direction, when pulse count reach 1000, C246 will be set

Network 001

LD	C246	
OUT	Y000	//when pulse count reach 1000, Y0 will be ON

Network 002

LDP	M1	
RST	C246	K1 //when M1 is ON, reset C246 and CV246

Ladder diagram:

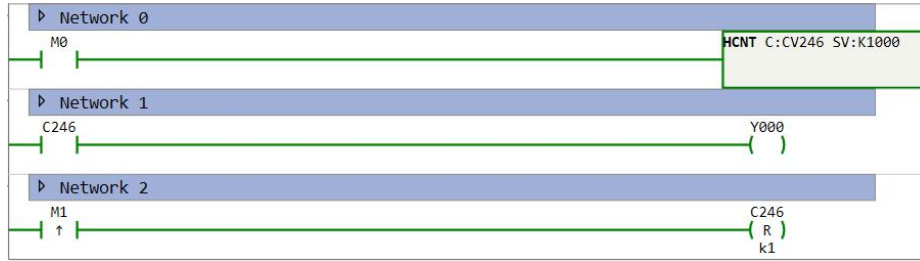


Figure 16.1. 5

3. AB-phase counting

Instruction table:

Network 001

LDP M8150

SET M8051 //set counting mode to single frequency multiplication mode when PLC run

Network 002

LD M0

HCNT CV251 K1000 //when M0 is ON, CV251 begin to count pulse of X0(A-phase) and X1(B-phase) input, when pulse count reach 1000, C251 will be set

Network 003

LD C251

OUT Y000 //when pulse count reach 1000, Y0 will be ON

Network 004

LDP M1

RST C251 K1 //when M1 is ON, reset C251 and CV251

Ladder diagram:

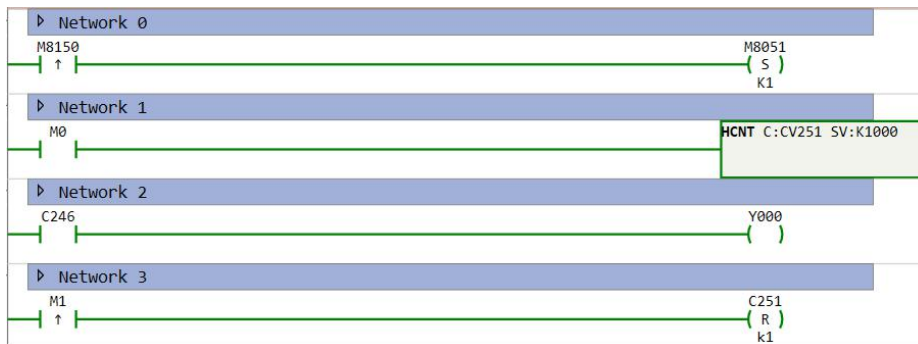


Figure 16.1. 6

EHCNT

Instruction introduction

1. This instruction can only be applied for **FGm** series PLC.
2. There are 4 kinds of counting mode for this instruction:
 - Single-phase counting: use one input port, and can only count in positive direction.
 - Double-phase counting: use two input ports, one is count in upward direction and another in downward direction.
 - AB-phase counting: use two input ports, one for A phase and another for B phase, can be set with single or quadruple frequency multiplication.
 - Pulse and direction counting: use two input ports, one is pulse input and another is pulse direction input. When pulse direction input is ON, counting counts in positive direction, otherwise in negative direction.
3. There are 2 kinds of triggering mode for start and reset counter:
 - Self-triggering: when this mode is selected, counter will start counting once it is enabled, and reset count and counting flag-bit when the count number reaches the target number.
 - External triggering: this mode has two sets of input ports to choose (**X25/X26** and **X27/X32**). When **X25/X26** is chosen, user can use **X25** to start the counter and use **X26** to reset the counter; When **X27/X32** is chosen, user can use **X27** to start the counter and use **X32** to reset the counter.
4. This instruction supports most 12 input ports (**X0** to **X7** and **X10** to **X13**). Shown as [Table 16.3.1](#), in 4 kinds of counting mode, correspondences between input ports and high-speed counters are different.
5. Range of count is -2,147,483,648 to 2,147,483,647, when count is out of range, there will be overflow or underflow.

	High-speed counter register	High-speed pulse input port											
		X00	X01	X02	X03	X04	X05	X06	X07	X10	X11	X12	X13
Single-phase	CV235	U											
	CV236		U										
	CV237			U									
	CV238				U								
	CV239					U							
	CV240						U						
	CV241							U					
	CV242								U				
	CV243									U			
	CV244										U		
	CV245											U	
	CV246												U
Double-phase	CV235	U	D										
	CV237			U	D								
	CV239					U	D						
	CV241							U	D				
	CV243									U	D		
	CV245											U	D
AB-phase	CV235	A	B										
	CV237			A	B								
	CV239					A	B						
	CV241							A	B				
	CV243									A	B		
	CV245											A	B
Pulse and Direction	CV235	PLS	DIR										
	CV237			PLS	DIR								
	CV239					PLS	DIR						
	CV241							PLS	DIR				
	CV243									PLS	DIR		
	CV245											PLS	DIR

Table 16.3. 1

Setting data

Inputs/outputs	Description	Operand	Range	Data type
----------------	-------------	---------	-------	-----------

(EN)	Input that enables or disables instruction.		0/1	bool
(CM)	Counting mode, 0 for single-phase, 1 for double-phase, 2 for AB-phase single frequency multiplication, 3 for AB-phase quadruple frequency multiplication, 4 for pulse and direction.	K	0~4	16-bit unsigned integer
(TM)	Triggering mode of counter, 0 for self-triggering, 2 for external triggering set 1 (X25/X26), 3 for external triggering set 2 (X27/X32).	K	0~3	16-bit unsigned integer
(SV)	Target number of corresponding counter.	K/H/D	-2147483648~2,147483647	32-bit signed integer

Attention

1. When connect input to port, to improve anti-interference, it is recommended to add shielding lines which on the input line, and connect shielding line to ground.
2. When one high-speed counter is used, the input port it occupies can't be used for other counter, also it can't be applied to common input use.
3. Response when counter counts to target number is immediate, it is unaffected from scan period.

HSCS

Instruction introduction

1. This instruction is used to trigger interruption event when counting of high-speed counter meets target number.
2. **FGm** series PLC doesn't support this instruction.
3. This instruction is used together with **HCNT**, it only takes effects when **HCNT** on corresponding high-speed counter is enabled.
4. Interruptions of this instruction must be triggered in order, if one interruption is skipped, the following ones won't be triggered only if user reset the counter.
5. Interruption event will be handled once it is triggered, nothing to do with scan period of PLC.

Setting data

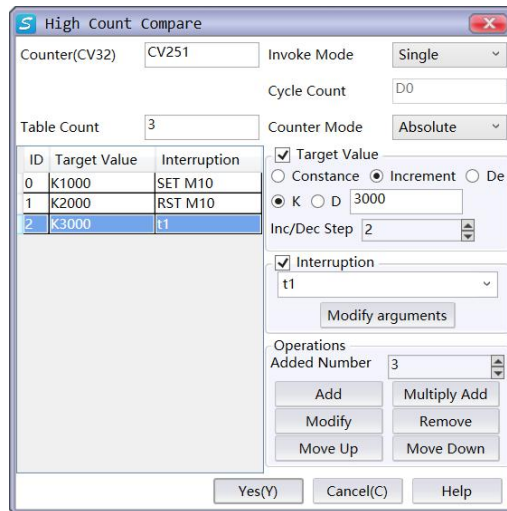


Figure 16.4. 1

1. Counter:
Can be set with single-phase counter (**CV235** to **CV238**) and AB-phase counter (**CV251** and **CV252**) (don't support double-phase counter so far).
2. Triggering mode:
Can be set with single mode or cycle mode. In single mode, when last interruption is triggered, counting of counter will continue, interruptions won't be triggered again only if user reset the counter; In cycle mode, when last interruption is triggered, counter will reset automatically and restart counting with cycle count adding 1, additionally interruptions can be triggered from the first.
3. Cycle count:
D-register that record how many cycles that interruptions are triggered.

4. Counting mode:
Can be set with absolute mode or relative mode. In absolute mode, target number means actual number of pulse outputting; In relative mode, target number means difference value of actual pulse number from last target to current.
5. Target number:
Can be set with K type data or D type data, user can add, move, modify, and remove target numbers by operations on the interface.
6. Interruption:
Can be set with 4 ways: setting coil, resetting coil, subroutine, and null. For setting coil, user can fill "#(SET Yn)" or "#(SET Mn)" to set corresponding Y and M registers; For resetting coil, user can fill "#(RST Yn)" or "#(RST Mn)" to reset corresponding Y and M registers; For subroutine, user can fill name of corresponding subroutine to call it; For null, there is no need to fill, system will do nothing when counting reaching the target.
7. Table length:
Number of target numbers, its maximum limit is 100.

Attention

1. In absolute mode, target numbers sequence of single-phase counting must be incremental, target number of double-phase counting must be different from last one.
2. In relative mode, target numbers of single-phase counting must be positive number, target number of double-phase counting must be nonzero.
3. First of target numbers sequence can't be 0.
4. Interval between two adjacent target numbers is recommended to set more than 50, otherwise the interruption may not be triggered normally.
5. set or reset coil of interruption can be set with coil of extend module.

Example

Instruction table:

```

Network 000
  LD      M0
  HCNT   CV251   K999999 //enable counter when M0 is ON
  HSCS   CV251   K999999  K3 //enable HSCS when M0 is ON
Network 001
  LDP    M1
  RST    C251 K1 //reset counter when M1 is ON, so that interruption can be triggered from start

```

Ladder diagram:

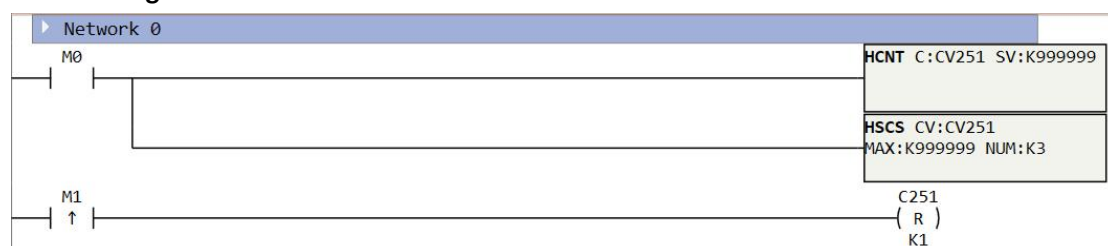


Figure 16.4. 2

String Instructions

Notes

Instructions in this chapter do interconversion between number and string. Type of number can be word, double word, and float, the string can only be ASCII encoded (not support Unicode). Table 17.1.1 shows the commonly used ASCII encoding.

Encoding (Hex)	Character	Encoding (Hex)	Character	Encoding (Hex)	Character
0x20	(space)	0x40	@	0x60	`
0x21	!	0x41	A	0x61	a
0x22	"	0x42	B	0x62	b
0x23	#	0x43	C	0x63	c
0x24	\$	0x44	D	0x64	d
0x25	%	0x45	E	0x65	e
0x26	&	0x46	F	0x66	f
0x27	'	0x47	G	0x67	g
0x28	(0x48	H	0x68	h
0x29)	0x49	I	0x69	i
0x2A	*	0x4A	J	0x6A	j
0x2B	+	0x4B	K	0x6B	k
0x2C	,	0x4C	L	0x6C	l
0x2D	-	0x4D	M	0x6D	m
0x2E	.	0x4E	N	0x6E	n
0x2F	/	0x4F	O	0x6F	o
0x30	0	0x50	P	0x70	p
0x31	1	0x51	Q	0x71	q
0x32	2	0x52	R	0x72	r
0x33	3	0x53	S	0x73	s
0x34	4	0x54	T	0x74	t
0x35	5	0x55	U	0x75	u
0x36	6	0x56	V	0x76	v
0x37	7	0x57	W	0x77	w
0x38	8	0x58	X	0x78	x
0x39	9	0x59	Y	0x79	y
0x3A	:	0x5A	Z	0x7A	z
0x3B	;	0x5B	[0x7B	{
0x3C	<	0x5C	\	0x7C	
0x3D	=	0x5D]	0x7D	}
0x3E	>	0x5E	^	0x7E	~
0x3F	?	0x5F	_	0x7F	DEL (delete)

Table 17.1.1 1

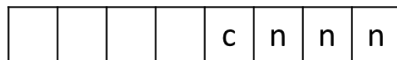
I_S

Instruction introduction

1. This instruction converts WORD type data (IN) into 8-byte string (OUT) with a format mask (FMT) when it is enabled.
2. Only bottom nibble (half-byte) of format mask takes effects, head bit (c) of the nibble decides if the output string is separated with point (".") or comma (","), bottom 3-bit (nnn) of the nibble decides the length of number after the point or comma (maximum is 5). Details refer to Figure 17.2.1.
3. This instruction is time-consuming, it is better to be triggered by edge.

IN	FMT	OUT(ASCII)		OUT+1(ASCII)		OUT+2(ASCII)		OUT+3(ASCII)	
		Low nibble	High nibble	Low nibble	High nibble	Low nibble	High nibble	Low nibble	High nibble
123	0x0004	(space)	(space)	0	,	0	1	2	3
-123	0x0004	(space)	-	0	,	0	1	2	3
123	0x000C	(space)	(space)	0	.	0	1	2	3
-123	0x000C	(space)	-	0	.	0	1	2	3

Low byte of format mask (FMT)



c: 1 for point, 0 for comma

nnn: length of number after point or comma, range is 0~5

Figure 17.2. 1

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	Input WORD data	K/H/D/CV/TV/AI/AO	-32768~32767	16-bit signed integer
(FMT)	Format mask to decide the format of output	K/H/D/CV/TV/AI/AO	-	
(OUT)	Head of registers to store output string	D	-	String

Table 17.2. 1

Example

Instruction table:

Network 000

LDP	X000		
I_S	K12345	H8	D10 //K12345 is converted to "12345" and store in D10~D13 (use point, 0 numbers after point)

Ladder diagram:

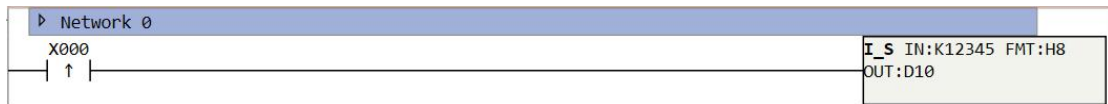


Figure 17.2. 2

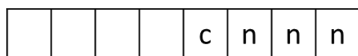
DI_S

Instruction introduction

1. This instruction converts DWORD type data (IN) into 12-byte string (OUT) with a format mask (FMT) when it is enabled.
2. Only bottom nibble (half-byte) of format mask takes effects, head bit (c) of the nibble decides if the output string is separated with point (".") or comma (","), bottom 3-bit (nnn) of the nibble decides the length of number after the point or comma (maximum is 7). Details refer to Figure 17.3.1.
3. This instruction is time-consuming, it is better to be triggered by edge.

IN	FMT	OUT(ASCII)		OUT+1(ASCII)		OUT+2(ASCII)		OUT+3(ASCII)		OUT+4(ASCII)		OUT+5(ASCII)	
		Low nibble	High nibble	Low nibble	High nibble	Low nibble	High nibble	Low nibble	High nibble	Low nibble	High nibble	Low nibble	High nibble
123456	0x0006	(space)	(space)	(space)	(space)	0	,	1	2	3	4	5	6
-123456	0x0006	(space)	(space)	(space)	-	0	,	1	2	3	4	5	6
123456	0x000E	(space)	(space)	(space)	(space)	0	.	1	2	3	4	5	6
-123456	0x000E	(space)	(space)	(space)	-	0	.	1	2	3	4	5	6

Low byte of format mask (FMT)



c: 1 for point, 0 for comma

nnn: length of number after point or comma, range is 0~5

Figure 17.3. 1

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	Input DWORD data	K/H/D/CV/TV/AI/AO	-2147483648~2147483647	32-bit signed integer
(FMT)	Format mask to decide the format of output	K/H/D/CV/TV/AI/AO	-	
(OUT)	Head of registers to store output string	D	-	String

Table 17.3. 1

Example

Instruction table:

Network 000
LDP X000

DI_S D0 HA D10 //D0 is converted to string and stored in D10~D15 (use point, 2 numbers after point)

Ladder diagram:

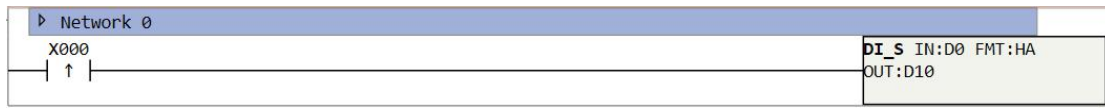


Figure 17.3. 2

R_S

Instruction introduction

1. This instruction rounds float type data (IN) and converts it into specified length string (OUT) with a format mask (FMT) when it is enabled.
2. Only low byte of format mask takes effects, head 4-bit (ssss) of the byte decides the length of output string, 5th bit (c) of the byte decides if the output string is separated with point (".") or comma (","), bottom 3-bit of the nibble decides the length of number after the point or comma (maximum is 7). Details refer to Figure 17.3.1.
3. This instruction is time-consuming, it is better to be triggered by edge.

IN	FMT	OUT(ASCII)		OUT+1(ASCII)		OUT+2(ASCII)	
		Low nibble	High nibble	Low nibble	High nibble	Low nibble	High nibble
1234.5	0x0061	1	2	3	4	,	5
-0.0004	0x0061	(space)	(space)	(space)	0	,	0
-3.67526	0x0069	(space)	(space)	-	3	.	7
1.95	0x0069	(space)	(space)	(space)	2	.	0

Low byte of format mask (FMT)

s	s	s	s	c	n	n	n
---	---	---	---	---	---	---	---

ssss: length of output string, range is 3~15

c: 1 for point, 0 for comma

nnn: length of number after point or comma, range is 0~5

Figure 17.4. 1

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	Input float data	K/D	$\pm 1.17549e-38F \sim \pm 3.40282e+38F$	float
(FMT)	Format mask to decide the format of output	K/H/D/CV/T V/AI/AO	-	
(OUT)	Head of registers to store output string	D	-	String

Table 17.3. 2

Example

Instruction table:

Network 000

LDP X000

R_S K3.141593

HFA

D10 //K3.141593 is converted to "

3.14" and stored in

D10~D17 (15 length string, use point, 2 numbers after point)

Ladder diagram:

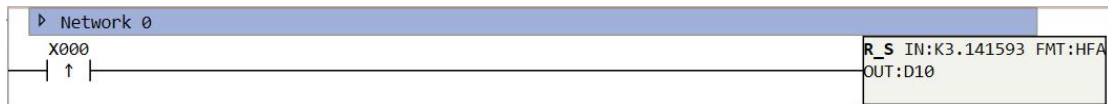


Figure 17.4. 2

S_I

Instruction introduction

1. This instruction converts string into 16-bit signed integer (WORD) when it is enabled.
2. This instruction converts from (IDX)th character of the string, the start character must be number character ("0" to "9"), "+", "-", or any numbers of space; The conversion ends at character which is not number character ("0" to "9") or one of follow characters: "+", "-", ",", and ".". Figure 17.5.1 shows examples of conversion and illegal input.
3. If the number after conversion is too great or too less compared with a 16-bit signed integer, there will be overflow and underflow, **M8169** will turn ON.
4. If the input string is invalid to convert to an integer, there will be no output, and **M8169** will also turn ON. Figure 17.5.1 shows some examples of conversion and illegal input.

Valid input		Invalid input	
Input string	Output integer	Input string	Reason
"123"	123	"A123"	Start with 'A'
"-00456"	-456	" "	Start with ' '
"123.45"	123	"++123"	End at '+'
"+2345"	2345	"+-123"	End at '-'
"000000123ABCD"	123	"+ 123"	End at ' '
		"50000"	Overflow
		"-50000"	Underflow

Figure 17.5. 1

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	head of registers store the input string	D	-	string
(IDX)	Number of index to start the conversion	K/H/D/CV/TV/AI/AO	1~65535	16-bit unsigned integer
(OUT)	Output WORD data	D/CV/TV/AO	-32768~32767	16-bit signed integer

Table 17.5. 1

Example

Instruction table:

Network 000

LDP	X000			
S_I	D0	K1	D100	//convert string in D0 and follow from index 1, and store integer result in D100

Ladder diagram:

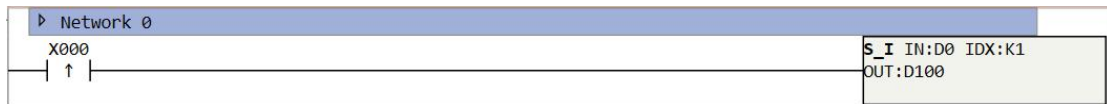


Figure 17.5. 2

S_DI

Instruction introduction

1. This instruction converts string into 32-bit signed integer (DWORD) when it is enabled.
2. This instruction converts from (IDX)th character of the string, the start character must be number character ("0" to "9"), "+", "-", or any numbers of space; The conversion ends at character which is not number character ("0" to "9") or is one of follow characters: "+", "-", ",", and ".". Figure 17.5.1 shows examples of conversion and illegal input.
3. If the number after conversion is too great or too less compared with a 32-bit signed integer, there will be overflow and underflow, **M8169** will turn ON.
4. If the input string is invalid to convert to an integer, there will be no output, and **M8169** will also turn ON. Figure 17.5.1 shows some examples of conversion and illegal input.

Valid input		Invalid input	
Input string	Output integer	Input string	Reason
"123"	123	"A98765"	Start with 'A'
"-00654"	-654	" "	Start with ' '
"543.21"	543	"++98765"	End at '+'
"+54321"	54321	"+-98765"	End at '-'
"000054321ABCD"	54321	"+ 98765"	End at ' '
		"3000000000"	Overflow
		"-3000000000"	Underflow

Figure 17.6. 1

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	head of registers store the input string	D	-	string
(IDX)	Number of index to start the conversion	K/H/D/CV/TV/ AI/AO	1~65535	16-bit unsigned integer
(OUT)	Output DWORD data	D	-2147483648~2147483647	32-bit signed integer

Table 17.6. 1

Example

Instruction table:

Network 000

LDP	X000			
S_DI	D0	K1	D100	//convert string in D0 and follow from index 1, and store integer result in D100D101

Ladder diagram:

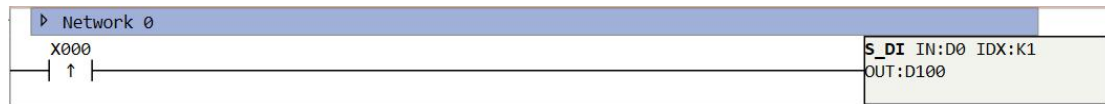


Figure 17.6. 2

S_R

Instruction introduction

1. This instruction converts string into float (DWORD) when it is enabled.
2. This instruction converts from (IDX)th character of the string, the start character must be number character ("0" to "9"), "+", "-", or any numbers of space; The conversion ends at character which is not number character ("0" to "9") or is one of follow characters: "+", "-", ",", and ".". Figure 17.5.1 shows examples of conversion and illegal input.
3. Conversion to float will not cause overflow or underflow, but if the significant digits of the conversion number is more than 7, the result will be rounded.
4. If the input string is invalid to convert to a float, there will be no output, and **M8169** will also turn ON. Figure 17.5.1 shows some examples of conversion and illegal input.

Valid input		Invalid input	
Input string	Output float	Input string	Reason
"123"	123.0	"A123"	Start with 'A'
"-00456"	-456.0	" "	Start with ' '
"123.45"	123.45	"++123"	End at '+'
"+2345"	2345.0	"+-123"	End at '-'
"00.000000123"	0.000000123	"+ 123"	End at ' '

Figure 17.7. 1

Setting data

Inputs/outputs	Description	Operand	Range	Data type
(EN)	Input that enables or disables instruction.		0/1	bool
(IN)	head of registers store the input string	D	-	string
(IDX)	Number of index to start the conversion	K/H/D/CV/TV/AI/AO	1~65535	16-bit unsigned integer
(OUT)	Output float data	D	$\pm 1.17549\text{e-}38\text{F} \sim \pm 3.40282\text{e+}38\text{F}$	float

Table 17.7. 1

Example

Instruction table:

Network 000

LDP X000

S_R D0 K1 D100 //convert string in D0 and follow from index 1, and store float result in D100D101

Ladder diagram:

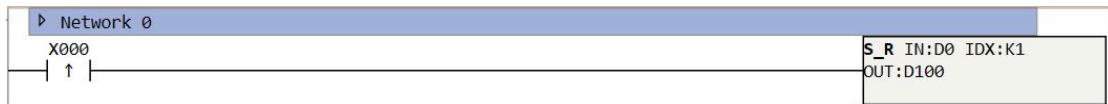


Figure 17.7. 2

PID Control Instructions

EPID

PID control is a kind of closed-loop control, accords to error ($e(t)$) between measured value ($y(t)$) of control target and set target value ($r(t)$), use proportional (K_p), integral (K_i), differential (K_d) coefficient to process the error and use it as input of control target ($u(t)$). Details refer to Figure 18.1.1.

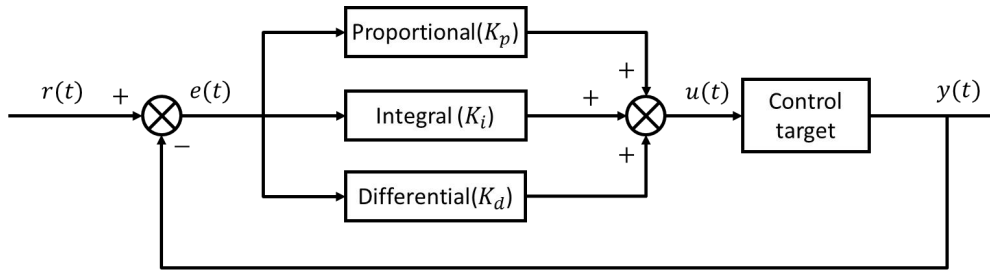


Figure 18.1. 1

The mathematic relations of $r(t)$, $e(t)$, $u(t)$, and $y(t)$ refer to following formula:

$$e(t) = r(t) - y(t)$$

$$u(t) = K_p \left[e(t) + \frac{1}{T_i} \int_0^t e(t) dt + T_d \frac{de(t)}{dt} \right]$$

In formula above K_p is proportional coefficient, T_i is integral time. T_d is differential time. While in actual case, there is minimum interval time for the sampling (T), so that the formula can be written as:

$$u_k = K_p \left[e_k + \frac{T}{T_i} \sum_{j=0}^k e_j + T_d \frac{e_k - e_{k-1}}{T} \right]$$

Integral time T_i and differential time T_d can also be transformed into integral coefficient K_i and differential coefficient K_d :

$$u_k = K_p e_k + K_i \sum_{j=0}^k e_j + K_d (e_k - e_{k-1}) \quad \left(K_i = \frac{K_p T}{T_i}, K_d = \frac{K_p T_d}{T} \right)$$

Instruction introduction

1. This instruction does PID control to output according to measured value and target value. It supports switching value output and analog quantity output, and can control 2 loops at most.
2. User can adjust the PID parameters manually, and can also use self-tuning to set the initial PID parameters. The self-tuning uses critical oscillations method, and it is recommended to adjust output of device (measured value) close to target value to lower the time cost of self-tuning.
3. This instruction has three modes to select and user can switch them during running.

Setting data

Address offset	Parameter	Data type	Permission	Description
0	Sampling cycle	dword	R/W	Time interval to sample the error, unit is ms (millisecond). For switching value mode, it is 1000ms by default; For analog quantity mode, it is 0 by default and must be assigned by user.
2	Proportional coefficient	float	R/W	K_p (proportional coefficient) of PID control, takes no effect when is set 0.
4	Integral time	float	R/W	T_i (Integral time) of PID control, of which unit is ms, takes no effect when is set 0.
6	Integral coefficient	float	R	K_i (Integral coefficient) of PID control, $K_i = K_p T / T_i$.
8	Differential time	float	R/W	T_d (Integral time) of PID control, of which unit is ms, takes no effect when is set 0.。
10	Differential coefficient	float	R	K_d (Differential coefficient) of PID control, $K_d = K_p T_d / T$.
12	Dead band range	float	R/W	When error between target value and measured value is less than dead band range, PID control is ineffective.
14	Output high limit	float	R/W	High limit of the output, for switching value mode, it is limit of duty ratio; For analog quantity mode, it is limit of the analog output.
16	Output low limit	float	R/W	Low limit of the output, for switching value mode, it is limit of duty ratio; For analog quantity mode, it is limit of the analog output.
18	PID configuration	hex	R/W	Configuration for PID control processing, details refer to following introduction.
20	PID calculated value	dword	R	Only for switching value mode, store close time during a sampling cycle, of which unit is ms. It is effective under analog quantity mode.
22	PID valid range	float	R/W	When error between target value and measured value is exceeds PID valid range, PID will output high/low limit according to sign of error and action direction.
24	Error code	hex	R	Error code record legality of parameters setting when instruction is enabled, details refer to following introduction.
25	PID mode	word	R/W	Data register records and switched the PID processing mode, value 0 for routine control mode, value 1 for manual control mode, value 2 for fuzzy self-adjusting control mode, else value for pausing control (output remains same).
26	Fuzzy self-adjusting status	word	R/W	Data register records and switches the status of fuzzy self-adjusting control, value 1 for in self-adjusting status, else value for not in self-adjusting status.
27-33	Reserved	-	-	System reserved area
34	Self-tuning status	word	R/W	Data register records and switches the status of self-tuning, value 0 for self-tuning completed, value 1 for self-tuning processing, value 2 for self-tuning failed. User can write 1 into this register to restart the self-tuning
36	Self-tuning timeout	dword	R/W	Timeout for self-tuning, if time of self-tuning exceed timeout, the self-tuning will fail.
40-49	Reserved	-	-	System reserved area

Table 18.1. 1

In EPID instruction guide, user can set head D-register to map parameters to D-registers, so

that parameters could be monitored and adjusted during running, the mapping rule of parameters refer to [Table 18.1.1](#). Parameters will be initialized by setting in guide, user can also initialize these parameters by transfer instructions. The following introduces details of the parameters:

1. Sampling cycle
Sampling cycle (T) is time interval that system samples the measured value and target value, its unit is ms (millisecond). Additionally, it is also the cycle of switching output. Sampling cycle must be longer than scan period of PLC.
2. Proportional coefficient
Proportional coefficient (K_p) scales the error to control the device, when it is big, the response of control will be quick, but there will be overshoot and fluctuation in device output; When it is small, the device output will be stable, but the response will be slow.
3. Integral time
Integral time (T_i) divides the accumulated error from start to current, when it is bigger, integral takes stronger effect; When it is smaller, integral takes weaker effect.
4. Integral coefficient
Integral coefficient (K_p) scales the accumulation of error to eliminate the difference between measured value and target value, it equals to $K_p T / T_i$.
5. Differential time
Differential time (T_d) multiplies the error difference from current one to last one, when it is bigger, differential takes stronger effect; When differential takes weaker effect.
6. Differential coefficient
Integral coefficient (K_d) scales the gradient (rate of change) of error to follow the change of measured value, it equals to $K_p T_d / T$.

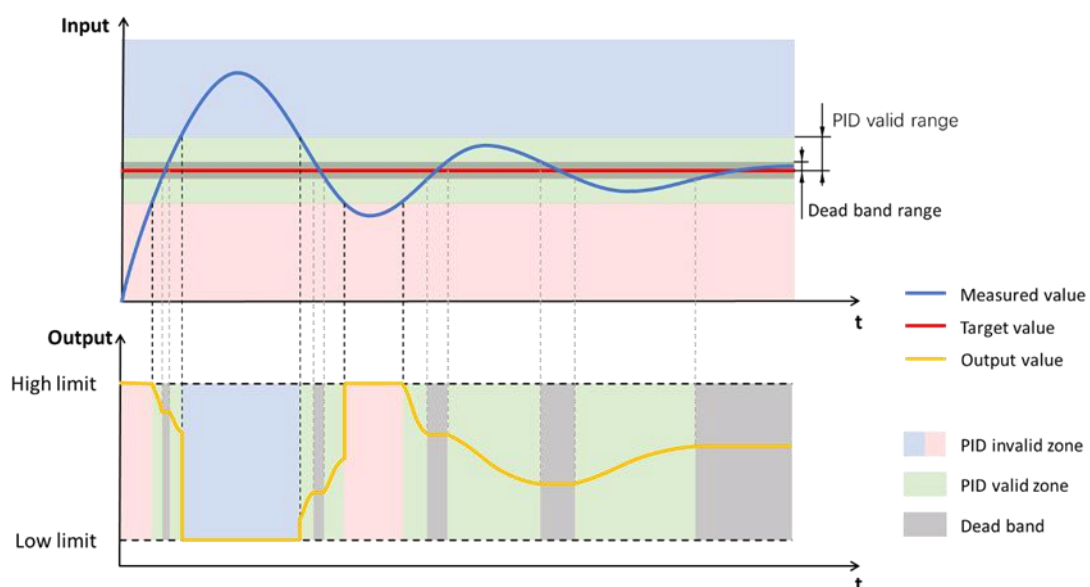


Figure 18.1. 2

7. Dead band range
Refer to Figure 18.1.2, when error between target value and measured value is less than dead band range, PID control is ineffective, the output value will unchanged still the error is out of the dead band. Function of dead band is to avoid unnecessary calculation

during stable state (error is small and stable).

8. Output high limit

Refer to Figure 18.1.2, this parameter limits the highest value of output, it must be greater than output low limit. Under switching value mode, output limit must be no more than sampling cycle (T).

9. Output low limit

Refer to Figure 18.1.2, this parameter limits the lowest value of output, it must be less than output high limit. Under switching value mode, output low limit must be nonnegative.

Proper output high limit and low limit can shorten time cost of self-tuning. If high limit is too high and low limit is too low, self-tuning will cause bigger and longer output fluctuation, so that it will cost more time; If high limit is too low and low limit is too high, measured value may not reach the target value and the self-tuning will fail.

10. PID configuration

Bit	15:4	3	2:1	0
Value	Reserved	SD	TYPE	Dir

Table 18.1. 2

This data-register records and controls following configuration of PID control:

- ◆ Dir: action direction of the target device, should be set by actual case. When this bit is ON, the action direction is forward, the error is equal to measured value minus target value, the PID output should increase/decrease when measured value increases/decreases; When this bit is OFF, the action direction is backward, the error is equal to target value minus measured value, the PID output should decrease/increases when measured value increases/decrease.
- ◆ TYPE: control type of PID, value 00 for PID, value 01 for PI, value 10 for P. PID is applicable to actual case with large lag, such as temperature control; PI is applicable actual case with quick response, such as pressure control and flow control.
- ◆ SD: this bit enables or disables automatic detection of stable state, value 0 for enable, value 1 for disable, it is 0 by default. Automatic detection of stable state is used for fuzzy self-adjusting control, when it is enabled, system will enter self-adjusting automatically when error is stable; When it is disabled, user need to set self-adjusting state manually.

11. PID calculated value

This value is only readable and if only valid under switching value mode. It records the calculated output ($u(t)$) of PID control, it's also the close duration of switching value in a sampling cycle.

12. PID valid range

Refer to Figure 18.1.2, when error between target value and measured value is within the dead band range, the PID control will be valid; Otherwise PID control will be invalid, in this case, if the action direction is forward, output will jump to high/low limit when error is plus/minus; If the action direction is backward, output will jump to low/high limit when error is plus/minus.

13. Error code

Bit	15:3	2	1	0
Value	Reserved	SC	HL	ST

Table 18.1. 3

When parameters are set illegal, PID won't calculate and output will maintain same, this data register record error of parameter setting:

- ◆ SC: when sampling cycle (T) is set illegal/legal, this bit turns ON/OFF.
- ◆ HL: when output high limit is set illegal/legal, this bit turns ON/OFF.
- ◆ ST: when PID valid range is set illegal/legal, this bit turns ON/OFF.

14. PID mode

User can switch PID modes by set different values in this data-register: value 0 for routine control mode, value 1 for manual control mode, value 2 for fuzzy self-adjusting mode, else value for stopping control.

- ◆ Routine control mode: under this mode, PID control will running with user set PID parameters, user can also modify self-tuning status to enable self-tuning.
- ◆ Manual control mode: under this mode, PID control will running as output that set by users. When this mode is switched to routine control mode or fuzzy self-adjusting mode, output will transition smoothly.
- ◆ Fuzzy self-adjusting mode: under this mode, PID control will detect if error is stable. If error is stable, PID control will enter self-adjusting mode and adjust the PID parameters automatically.
- ◆ Stopping control: output will maintain the same when control is stopped.

15. Fuzzy self-adjusting status

This data register records if control enters self-adjusting status under fuzzy self-adjusting mode, value 1 for in self-adjusting status, else value for not. Control will enter self-adjusting mode automatically when error is stable, and adjust PID parameters (K_p , K_i , K_d) automatically, user can also set the value with 1 to force control to enter self-adjusting mode, but this operation will cost more time to stabilize the error and even fail if enter the mode when error is not stable.

16. Self-tuning status

This data register record and control status of self-tuning, value 0 for self-tuning completed, value 1 for self-tuning processing, value 2 for self-tuning failed, else value for pausing control (output maintains the same).

Self-tuning is used to set initial PID parameters when PID control starts. When self-tuning is enabled, output will jump between high limit and low limit alternately to make fluctuation of measured value to reach the target value.

It is recommended to adjust the output to make measured value being close to target value under manual control mode, so that the self-tuning will be quick and the initial PID parameter will be set reasonable.

17. Self-tuning timeout

This data register set timeout of self-tuning, timeout of self-tuning is no less than 60 minutes. if time cost of self-tuning exceeds the timeout, self-tuning status will be self-tuning failed.

Appendix

System special function registers

System special function registers are D-registers (data registers) and M-registers (bit registers) that have special function. Details refer to following table:

Register	Description	read-write access
D-registers (data registers)		
D8040~D8047	Data-registers monitor the last 8 active state bits.	R
D8048	Current mode of IST (0: none, 1: manual mode, 2: zero-return mode, 3: single-step mode, 4: single-cycle mode, 5: automatic mode).	R
D8066	Station number of COM0, range from 1 to 127.	R
D8067	Station number of COM1, range from 1 to 127.	R
D8068	Baud rate mode of COM0, range from 0 to 7, corresponding to 1200~115200 in configuration.	R
D8069	Baud rate mode of COM0, range from 0 to 7, corresponding to 1200~115200 in configuration.	R
D8070	Data bit of COM0, be in constant 0, corresponding to 8 data bits.	R
D8071	Data bit of COM1, be in constant 0, corresponding to 8 data bits.	R
D8072	Stop bit of COM0, range from 0 to 1, corresponding to 1 or 2 stop bits.	R
D8073	Stop bit of COM1, range from 0 to 1, corresponding to 1 or 2 stop bits.	R
D8074	Check bit mode of COM0, range from 0 to 2, corresponding to NONE, EVEN, and ODD.	R
D8075	Check bit mode of COM0, range from 0 to 2, corresponding to NONE, EVEN, and ODD.	R
D8076~D8079	System reserved.	R/W
D8080~D8085	Undefined	
D8086	Station number of COM2, range from 1 to 127.	R
D8087	Baud rate mode of COM2, range from 0 to 7, corresponding to 1200~115200 in configuration.	R
D8088	Data bit of COM2, be in constant 0, corresponding to 8 data bits.	R
D8089	Stop bit of COM2, range from 0 to 1, corresponding to 1 or 2 stop bits.	R
D8090	Check bit mode of COM2, range from 0 to 2, corresponding to NONE, EVEN, and ODD.	R
D8092	Y000 pulse port acceleration/deceleration time of multistage pulse output.	R/W
D8093	Y001 pulse port acceleration/deceleration time of multistage pulse output.	R/W
D8094	Y002 pulse port acceleration/deceleration time of multistage pulse output.	R/W
D8095	Y003 pulse port acceleration/deceleration time of multistage pulse output.	R/W
D8108	Y000 pulse port error segment number (For multistage pulse output)	R
D8109	Y001 pulse port error segment number (For multistage pulse output).	R
D8110	Y002 pulse port error segment number (For multistage pulse output).	R

D8111	Y003 pulse port error segment number (For multistage pulse output).	R
D8124	Y000 pulse port current segment number (For multistage pulse output).	R
D8125	Y001 pulse port current segment number (For multistage pulse output).	R
D8126	Y002 pulse port current segment number (For multistage pulse output).	R
D8127	Y003 pulse port current segment number (For multistage pulse output).	R
D8140(D8141)	Y000 pulse port accumulative count of pulse number.	R/W
D8142(D8143)	Y001 pulse port accumulative count of pulse number.	R/W
D8144(D8145)	Y002 pulse port accumulative count of pulse number.	R/W
D8146(D8147)	Y003 pulse port accumulative count of pulse number.	R/W
D8173	Target number of interrupt timer 0, range from 1 to 32767ms.	R/W
D8174	Target number of interrupt timer 0, range from 1 to 32767ms.	R/W
D8175	Current scan period (unit is 0.1ms).	R
D8176	<p>error code</p> <p>0x00: no error</p> <p>0x09: real-time clock error</p> <p>0x0A: watchdog timer timeout</p> <p>0x0C: serial port communication timeout</p> <p>0x0D: USB port communication timeout</p> <p>0x0E: XML region data exception,</p> <p>0x0F: configuration region data exception</p> <p>0x10: HashCode region data exception</p> <p>0x11: Modbus table data exception</p> <p>0x12: PLSTABLE table data exception</p> <p>0x13 PLSBLOCK region data exception</p> <p>0x17: segments number of multistage pulse output out of range</p> <p>0x18: first segment frequency of multistage pulse lower than default start frequency</p> <p>0x19: frequency lower than low limit</p> <p>0x1A: target number of single-stage pulse lower than low limit.</p> <p>0x1B: acceleration/deceleration time of single-stage pulse lower than low limit</p> <p>0x1C: parameters don't meet legal condition of acceleration/deceleration</p> <p>0x1D: last segment of multistage pulse doesn't meet legal condition of acceleration/deceleration</p> <p>0x1E: FPGA loads firmware timeout</p> <p>0x1F: EPCS4 ID check error</p> <p>0x20: EPCS4 erases data error</p> <p>0x21: EPCS4 update firmware timeout</p> <p>0x22: EPCS4 update data package error</p> <p>0x23: EPCS4 response check error</p> <p>0x24: interpolation parameters error</p> <p>0x25: interpolation prospective error</p> <p>0x26: interpolation velocity error (be 0 or too big)</p> <p>0x27: end-point coordinates error (movement can't be 0)</p> <p>0x28: arc interpolation radius error</p>	R/W

	<p>0x29: be about to cross the border</p> <p>0x2A: interpolation parameters don't meet legal condition of acceleration/deceleration</p> <p>0x2B: chord error too big</p> <p>0x2C: exceed maximum of pulse frequency</p> <p>0x2D: ECAM data error</p> <p>0x2E: ECAM parameters error</p> <p>0x34: low extension module version, new function not applicable</p> <p>0x35: low PLC version, new function not applicable</p> <p>0x36: extension module communication error</p> <p>0x37: extension module retransmission timeout</p> <p>0x38: extension module type query error</p> <p>0x39: extension module parameters passing error</p> <p>0x3A: Ethernet port module configuration error</p> <p>0x3B: CAD graph import to FLASH handshake error</p> <p>0x3F: HCNT parameter setting unreasonable</p> <p>0x40: CADflash read/write error</p> <p>0x41: CADflash download data package error</p> <p>0x42: graph data read error</p> <p>0x43: spline interpolation error</p> <p>0x44: zero signal of zero-return disabled</p> <p>0x45: displacement of acceleration/deceleration too long</p> <p>0x46: high-speed counter ports conflict</p> <p>0x48: ECAM FIFO cross border</p> <p>0x49: platform-system pulse output port error</p> <p>0x4A: TBL segments jump error</p> <p>0x4B: pulse peak flirting judgement timeout</p> <p>0x4C: ZRNR frequency missing</p> <p>0x7A: stack overflow error</p> <p>0x7B: access inner registers out of range</p>	
D8177	Maximum of scan period (unit is millisecond).	R/W
D8178	Potentiometer VR0 reading.	R/W
D8179	Potentiometer VR1 reading.	R/W
D8180~D8192	System reserved	
D8194(D8195)	Y000 pulse port basal frequency (5Hz~ Y000 maximum frequency)	R/W
D8196(D8197)	Y001 pulse port basal frequency (5Hz~ Y001 maximum frequency)	R/W
D8198(D8199)	Y002 pulse port basal frequency (5Hz~ Y002 maximum frequency)	R/W
D8200(D8201)	Y003 pulse port basal frequency (5Hz~ Y000 maximum frequency)	R/W
D8202(D8203)~D8212(D8213)	Pulse port basal frequency of pulse port Y004~Y007 and Y010~Y011 (5Hz~ Y000 maximum frequency)	R/W
D8214(D8215)	Y000 pulse port maximum frequency	R/W
D8216(D8217)	Y001 pulse port maximum frequency	R/W
D8218(D8219)	Y002 pulse port maximum frequency	R/W
D8220(D8221)	Y003 pulse port maximum frequency	R/W

D8222(D8223) ~ D8232(D8233)	Pulse port maximum frequency of pulse port Y004~Y007 and Y010~Y011.	R/W
D8234	Y0 port deceleration time under FOLLOW limit position mode (address increases with 6 for Y1~Y5)	R/W
D8236(D8237)	Y0 port FOLLOW positive limit position (address increases with 6 for Y1~Y5)	R/W
D8238(D8239)	Y0 port FOLLOW negative limit position (address increases with 6 for Y1~Y5)	R/W
D8294	Y0 port deceleration time under platform-system limit position mode (address increases with 6 for Y1~Y5)	R/W
M-registers (bit-registers)		
M8035	High-speed counter CV235 counting direction.	R
M8036	High-speed counter CV236 counting direction.	R
M8037	High-speed counter CV237 counting direction.	R
M8038	High-speed counter CV238 counting direction.	R
M8039	High-speed counter CV239 counting direction.	R
M8040	High-speed counter CV240 counting direction.	R
M8041	High-speed counter CV241 counting direction.	R
M8042	High-speed counter CV242 counting direction.	R
M8043	High-speed counter CV243 counting direction.	R
M8044	High-speed counter CV244 counting direction.	R
M8045	High-speed counter CV245 counting direction.	R
M8046	Error flag bit of extension module initialization.	R
M8050	High-speed counter CV250 frequency multiplication (0: single frequency multiplication, 1: quadruple frequency multiplication)	R/W
M8051	High-speed counter CV251 frequency multiplication (0: single frequency multiplication, 1: quadruple frequency multiplication)	R/W
M8052	High-speed counter CV252 frequency multiplication (0: single frequency multiplication, 1: quadruple frequency multiplication)	R/W
M8053	High-speed counter CV253 frequency multiplication (0: single frequency multiplication, 1: quadruple frequency multiplication)	R/W
M8054	High-speed counter CV254 frequency multiplication (0: single frequency multiplication, 1: quadruple frequency multiplication)	R/W
M8055	High-speed counter CV255 frequency multiplication (0: single frequency multiplication, 1: quadruple frequency multiplication)	R/W
M8056	High-speed counter CV250 counting direction.	R
M8057	High-speed counter CV251 counting direction.	R
M8058	High-speed counter CV252 counting direction.	R
M8059	High-speed counter CV253 counting direction.	R
M8060	High-speed counter CV254 counting direction.	R
M8061	High-speed counter CV255 counting direction.	R
M8062	PWMS high low-level mode (0: high after low, 1: high before low).	R/W
M8064	COM2 is transmitting data.	R
M8065	COM2 is receiving data.	R
M8066	COM2 transmission completes	R/W

M8067	COM2 reception completes	R/W
M8068	Global polarity of all pulse outputting ports (pulse direction reversals when it is 1).	R/W
M8069	Frequency curve type of acceleration/deceleration (0: T-shape type, 1: S-shape type).	R/W
M8070	If to ignore the pulse output error of Y0 port, set 0 by default (0: ignore and continue outputting pulse, 1: stop outputting pulse).	R/W
M8071	If to ignore the pulse output error of Y1 port, set 0 by default (0: ignore and continue outputting pulse, 1: stop outputting pulse).	R/W
M8072	If to ignore the pulse output error of Y2 port, set 0 by default (0: ignore and continue outputting pulse, 1: stop outputting pulse).	R/W
M8073	If to ignore the pulse output error of Y3 port, set 0 by default (0: ignore and continue outputting pulse, 1: stop outputting pulse).	R/W
M8086	Illegal parameter setting of Y0 pulse outputting port.	R
M8087	Illegal parameter setting of Y1 pulse outputting port	R
M8088	Illegal parameter setting of Y2 pulse outputting port.	R
M8089	Illegal parameter setting of Y3 pulse outputting port.	R
M8102	Pulse direction of Y0 pulse outputting (0: negative, 1: positive)	R
M8103	Pulse direction of Y1 pulse outputting (0: negative, 1: positive)	R
M8104	Pulse direction of Y2 pulse outputting (0: negative, 1: positive)	R
M8105	Pulse direction of Y3 pulse outputting (0: negative, 1: positive)	R
M8134	PLSY Y0 port is outputting pulse.	R
M8135	PLSY Y1 port is outputting pulse.	R
M8136	PLSY Y2 port is outputting pulse.	R
M8137	PLSY Y3 port is outputting pulse.	R
M8144	Platform system 1 is outputting pulse.	R
M8145	Platform system 2 is outputting pulse.	R
M8146	Platform system 3 is outputting pulse.	R
M8147	Platform system 4 is outputting pulse.	R
M8148	Platform system 5 is outputting pulse.	R
M8150	Only be closed during first scan period.	R
M8151	Close when program starts running.	R
M8152	Open when program starts running.	R
M8153	RTC battery low voltage (0: OK, 1: ERROR).	R
M8154	DC 24V voltage error (0: OK, 1: ERROR).	R
M8155	IO configuration error (0: OK, 1: ERROR).	R
M8156	PLC hardware error (0: OK, 1: ERROR).	R
M8157	Output refresh error (0: OK, 1: ERROR).	R
M8158	10 milliseconds interval pulse output (0: LOW, 1: HIGH).	R
M8159	100 milliseconds interval pulse output (0: LOW, 1: HIGH).	R
M8160	1 second interval pulse output (0: LOW, 1: HIGH).	R
M8161	1 minute interval pulse output (0: LOW, 1: HIGH).	R
M8162	Clear all data out of retentive zone (0: NONE, 1: CLEAR).	R/W
M8163	Clear all data in retentive zone (0: NONE, 1: CLEAR).	R/W

M8164	Disable all outputs (0: ENABLE, 1: DISABLE).	R/W
M8165	Direction of shift instruction (0: RIGHT, 1: LEFT).	R
M8166	Overflow flag-bit of shift instructions (0: OK, 1: ERROR).	R
M8167	Zero flag-bit of shift instruction (0: OK, 1: ERROR).	R
M8168	BCD conversion error (0: OK, 1: ERROR).	R
M8169	Calculation result overflow (0: OK, 1: ERROR).	R
M8170	Calculation result is minus (0: OK, 1: ERROR).	R
M8171	Calculation result is zero (0: OK, 1: ERROR).	R
M8172	Divisor is zero (0: OK, 1: ERROR).	R
M8173	Offset address out of bound ().	R
M8174	PLC in error status (0: OK, 1: ERROR).	R
M8175	Running state of PLC (0: STOP, 1: RUN).	R
M8176	COM0 is transmitting data.	R
M8177	COM1 is transmitting data.	R
M8178	COM0 is receiving data	R
M8179	COM1 is receiving data	R
M8180	COM0 transmission completed.	R/W
M8181	COM1 transmission completed.	R/W
M8182	COM0 reception completed.	R/W
M8183	COM1 reception completed.	R/W
M8184	Extension module 1 communication error.	R
M8185	Extension module 2 communication error.	R
M8186	Extension module 3 communication error.	R
M8187	Extension module 4 communication error.	R
M8188	Extension module 5 communication error.	R
M8189	Extension module 6 communication error.	R
M8190	Extension module 7 communication error.	R
M8191	Extension module 8 communication error.	R
M8192	Local polarity of pulse outputting port Y0 (pulse direction reversals when it is 1).	R/W
M8193	Local polarity of pulse outputting port Y1 (pulse direction reversals when it is 1).	R/W
M8194	Local polarity of pulse outputting port Y2 (pulse direction reversals when it is 1).	R/W
M8195	Local polarity of pulse outputting port Y3 (pulse direction reversals when it is 1).	R/W
M8202	Stop mode of EDRVI/EDRVA (0: instantly stop, 1: slower to stop).	R/W
M8203	Stop mode of FOLLOW in limit position (0: instantly stop, 1: slower to stop).	R/W
M8204	Stop mode of FOLLOW under limit signal (0: instantly stop, 1: slower to stop).	R/W
M8205	Y0 port positive limit signal.	R/W
M8206	Y0 port negative limit signal.	R/W
M8207	Y1 port positive limit signal.	R/W
M8208	Y1 port negative limit signal.	R/W
M8209	Y2 port positive limit signal.	R/W
M8210	Y2 port negative limit signal.	R/W
M8211	Y3 port positive limit signal.	R/W
M8212	Y3 port negative limit signal.	R/W

M8225	If to maintain direction when pulse outputting stop (0: not maintain, 1: maintain).	R/W
M8234	PLSY Y0 port pulse outputting completed.	R
M8234	PLSY Y1 port pulse outputting completed.	R
M8236	PLSY Y2 port pulse outputting completed.	R
M8237	PLSY Y3 port pulse outputting completed.	R