Index

Applied instructions allow the user to perform complex data manipulations, mathematical operations. Each applied instruction has unique mnemonics and special function numbers. Each applied instruction will be expressed using a table similar to that show below. And will be found at the beginning of the description of each new instruction.

COMPARE

FNC(10)			16 bits:CMP & CMP(P) - - - - - - - - - - - - - - - - - - - 7 Steps
D	CMP	P	32 bits:(D)CMP&(D)CMP(P) - - - - - - - - - - - - - - -13 Steps

Operands:	← [S1.][S2.] →								
	K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z
Operands:	← [D.] →								
	X	Y	M	S					

No modification of the instruction mnemonic is required for 16 bit operation, and it will operate continuously, i.e. on every scan cycle of the user program, the instruction will operation and provide a new result.
However, pulse operation requires a ‘P’ to be added directly after the mnemonic, while 32 bit operation requires a “D” to be added before the mnemonic. This means that if an instruction was being used with both pulse and 32 bit applied operation it would look like D***P, where *** was the basic mnemonic.

The ‘pulse’ function allows the associated instruction to be Activated on the rising edge of the control input. The Instruction is driven ON for the duration of one program Scan cycle. Thereafter, even if the control input remains on the associated instruction will not be active.
Following is Symbols list:
[D.]: Destination device
[S.]: Source device

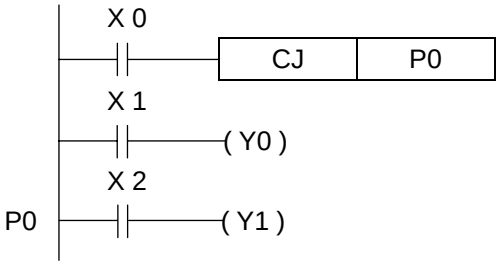
[m,n]: Number of active devices, bits or an operational constant.
Following is instruction modifications:
*** - An instruction operation in 16 bit mode, where *** identifies the instruction mnemonic.
***P - An instruction modified to use 16 bits pulse operation.
D*** - An instruction modified to use 32 bits operation.

Condition Jump

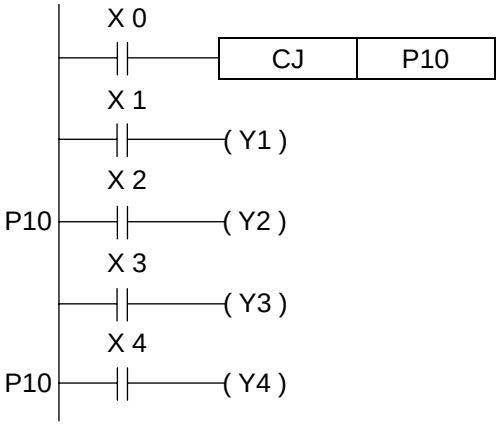
FNC(00)			16 bits: CJ & CJ(P) - 3 Steps
	CJ	P	

		J1n	J2n--

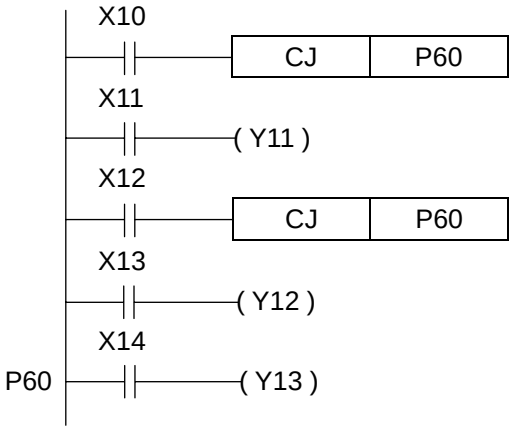
Operand: P00 ~ P63



Example (A)



Example (B)



Example (C)

- Example (A): If X0 ON forces the program to jump to LAB P0, any program area which is skipped will not update. Output statuses will not change even input the devices.
- Example (A): If miss LAB P0 pointer, then X0 ON will jump directly to END.
- If a backwards jump is used, then need to care the watchdog timer overrun.
- If LAB pointer is duplicated to use, only the last pointer is effective.
- Example (B): X0 ON forces the program to jump to the second LAB pointer.
- Example (C): Many CJ statements can be assigned to jump to the same pointer.

Subroutine Call

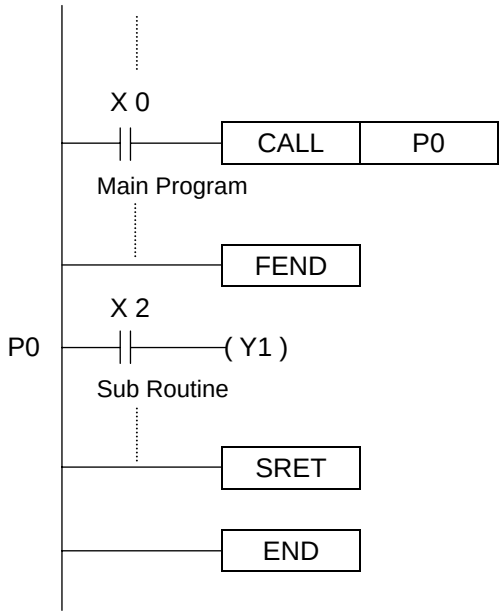
FNC(01)			16 bits: CALL & CALL(P) - - - - - - - - - - - - - - - - - - 3 Steps			J1n	J2n--
	CALL	P					

Operand: P00~P63

Subroutine Return

FNC(02)			16 bits: SRET - 1 Steps			J1n	J2n--
	SRET						

Operand: None



- When X0 ON, program will jump to subroutine pointer LAB P0 and execute Subroutine until SRET instruction is executed, then program return to original step and continue processing.
- The LAB assigns beginning of subroutine must be programmed after an FEND.
- The same LAB can only be used once, but many CALL statements can be assigned to a single LAB subroutine.
- Subroutines can be nested for 5 levels including one CALL instruction.

Interrupt Return

FNC(03)	16 bits: IRET - 1 Steps			J1n	J2n--
IRET					

Operand: None

Enable Interrupt

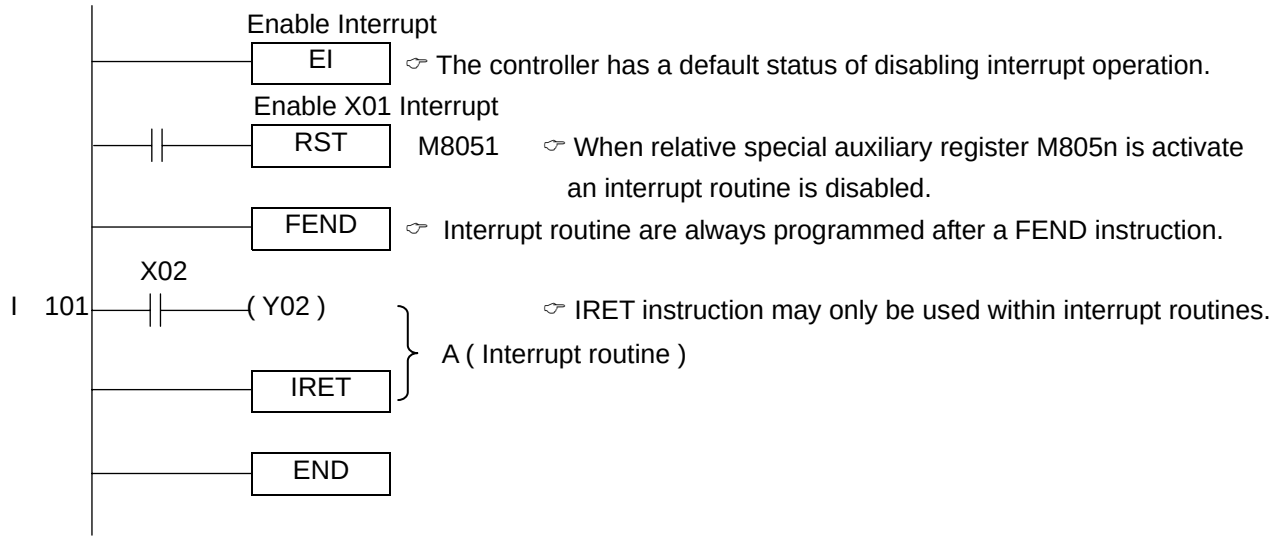
FNC(04)	16 bits: EI - 1 Steps			J1n	J2n--
EI					

Operand: None

Disable Interrupt

FNC(05)	16 bits: DI - 1 Steps			J1n	J2n--
DI					

Operand: None



Number of Interrupt pointer

I □ 0 □ Interrupt Input Relay 6 points

0 : Falling Edge Trigger, 1 : Rising Edge Trigger.

No. of Input Relay X0 ~ X5 (0 ~ 5) (Can't duplicate)

I □ □ □ Timer interrupt 1 points

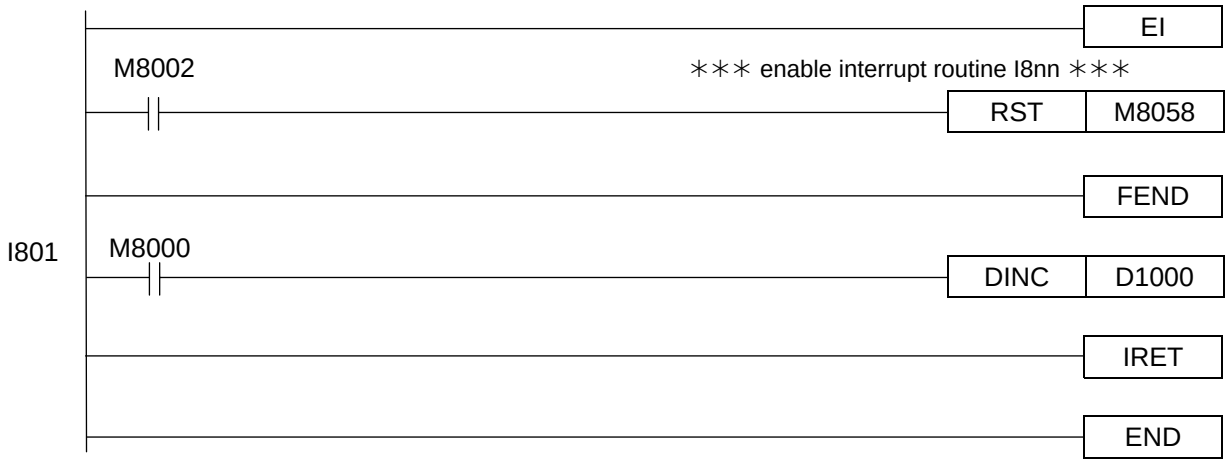
01 ~ 99 ms

8

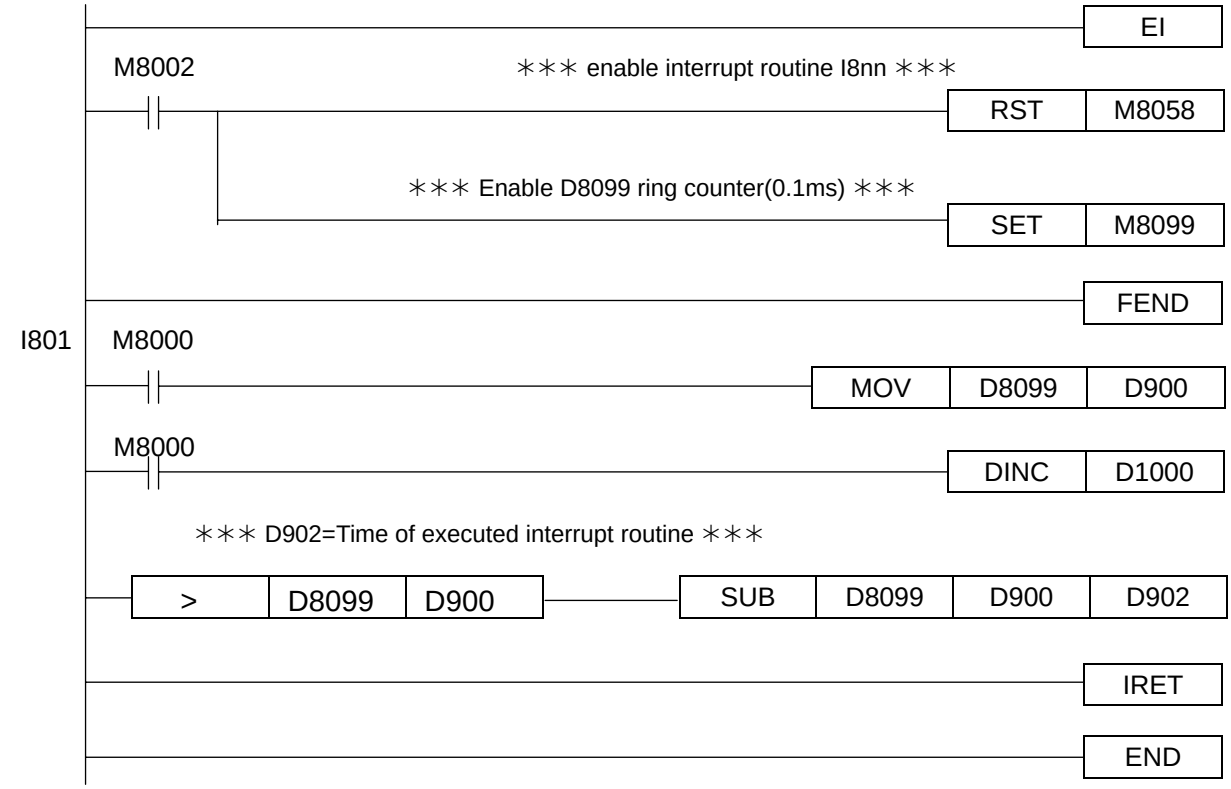
<< Note >>

- ♦ When an interrupt program execute, other Interrupt Call is ineffective.
- ♦ If Interrupt occur within the range of Disable Interrupt (DI~EI), this interrupt request signal is stored temporarily, and execute until within the range of Enable Interrupt (EI~DI).
- ♦ When Disable Interrupt flag M805Δ act, the corresponding Interrupt input will not be executed.
- ♦ In interruption program, FNC(50) REF command can not be used. (Ex: section A in above sample program)

Timer Interrupt program



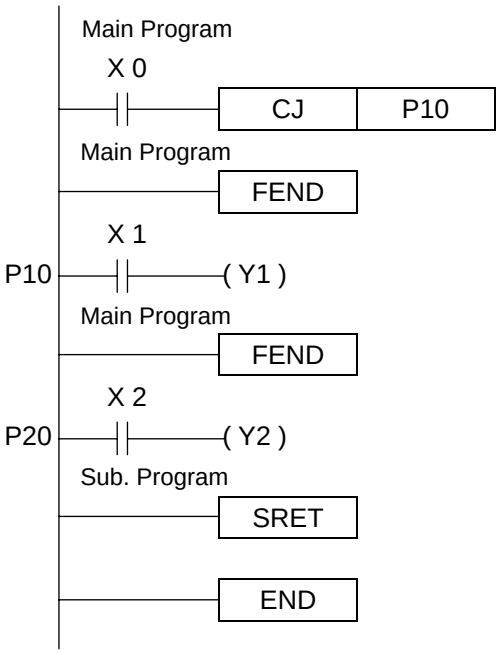
Caculated Interrupt routine executed time



First End

FNC(06)	16 bits: FEND - 1 Steps			J1n	J2n--
FEND					

Operand: None

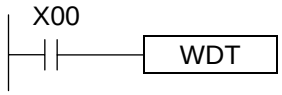


- ♦ A FEND instruction indicates the first end of a main program and the start of the subroutine program area to be used.
- ♦ Multiple FEND instruction can be use to separate different subroutines.
- ♦ When FEND is executed, the program return to Step 0.
- ♦ FEND can't be used after an END instruction.

Watch Dog Timer

FNC(07)	16 bits: WDT - 1 Steps			J1n	J2n--
WDT	P				

Operand: None



- ♦ This instruction will compare the cycle time with the content of special data register D8000.
- ♦ If the watch dog timer > the content of D8000, then error occurred and error code is 6309.
- ♦ Can use MOV instruction to change content of special data register D8000.
- ♦ If do not write WDT instruction in program, then the watch dog timer is ineffective.

FOR

FNC(08)			16 bits: FOR - 7 Steps			
	FOR					

			J1n	**J2n--**

Operands: ⟵ [S.] ⟶

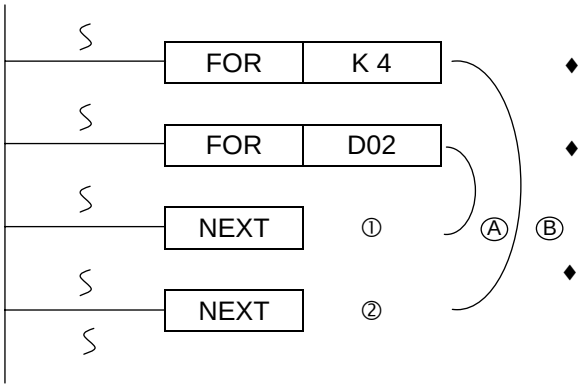
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

NEXT

FNC(09)			16 bits: NEXT - 7 Steps			
	NEXT					

			J1n	**J2n--**

Operand: None



- ♦ After program B execute 4 times, then execute the program below ② NEXT.
- ♦ If the content of D0Z is 5, then program B is executed 4 times, and program A will be executed 20 times.
- ♦ The maximum nest level of FOR –NEXT is 5 levels.

Compare

FNC(10)			16 bits: CMP & CMP(P) - - - - - - - - - - - - - - - - - - - 7 Steps
D	CMP	P	32 bits: (D)CMP&(D)CMP(P) - - - - - - - - - - - - - - -13 Steps

		J1n	**J2n--**

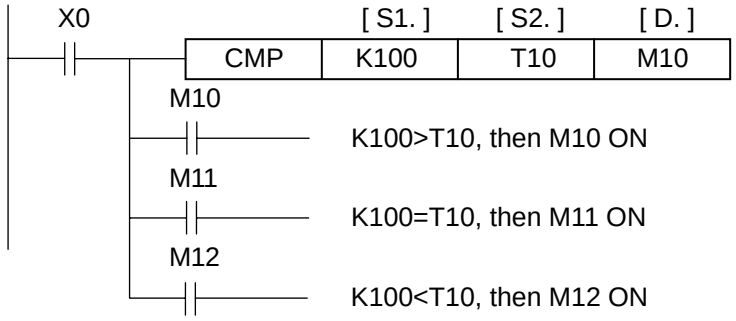
Operands: [S1.][S2.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands: [D.]

X	Y	M	S

Flag:



- ♦ Data of [S1.] is compared with data of [S2.] and [D.] will be changed according to the result. This will automatic occupy 3 bit destination devices from head address of designation M10 ~ M12.
- ♦ Full algebraic comparisons are used, i.e. -10 smaller than +2.
- ♦ When X0 OFF, then [D.] bit devices status will not be changed.

Zone Compare

FNC(11)			16 bits: ZCP & ZCP(P) - 9 Steps
D	ZCP	P	32 bits: (D)ZCP&(D)ZCP(P) - - - - - - - - - - - - - - - -17 Steps

		J1n	**J2n--**

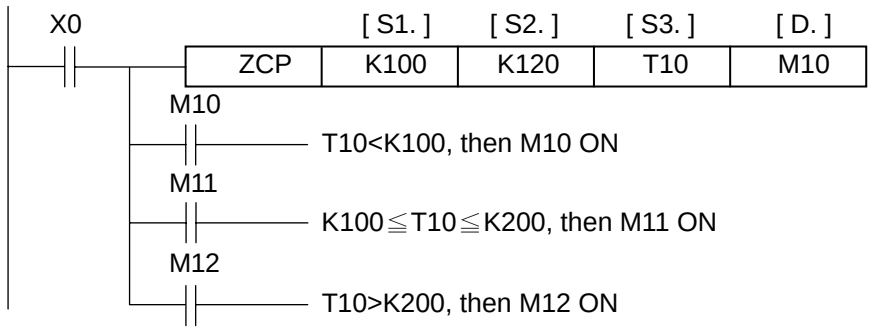
Operands: [S1.] [S2.][S3.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands: [D.]

X	Y	M	S

Flag:



- ♦ Content of [S3.] is compared with data range of [S1.] and [S2.] and [D.] will be changed according to the result. This will automatic occupy 3 bit destination devices from head address of designation M10 ~ M12.
- ♦ Set [S1.] ≦ [S2.], if [S1.] > [S2.], then data of [S2.] is as same as data of [S1.].
- ♦ Full algebraic comparisons are used, i.e. -10 smaller than +2.
- ♦ When X0 OFF, then [D.] bit devices status will not be changed.

Move

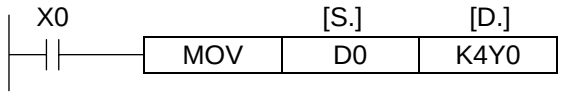
FNC(12)			16 bits: MOV & MOV(P) - - - - - - - - - - - - - - - - - - - 5 Steps
D	MOV	P	32 bits: (D)MOV&(D)MOV(P) - - - - - - - - - - - - - - -9 Steps

			J1n	**J2n--**

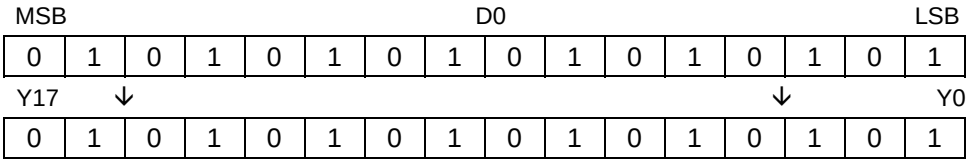
Operands: [S.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

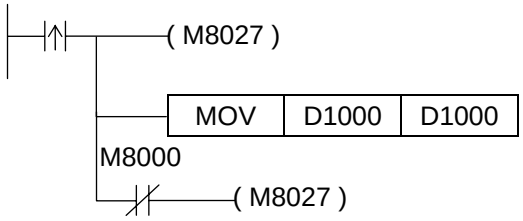
[D.]



♦ When X0 ON, contents of source device [S.] copied to destination device [D.].



♦ When M8027 ON, CPU will write content of [S.] into EEPROM, [D.] only D register can be used.



Note: When M8027 ON, for avoid to damage EEPROM, must be used Pulse Instruction MOV(P).

Shift Move

FNC(13)			16 bits: SMOV & SMOV(P) - - - - - - - - - - - - - - - - - 7 Steps
	SMOV	P	

Operands: [S.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.]

♦ Reserved

Complement

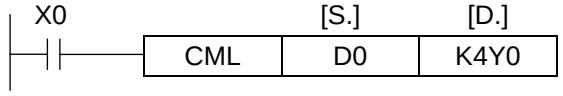
FNC(14)			16 bits: CML & CML(P) - - - - - - - - - - - - - - - - - - - 5 Steps
D	CML	P	32 bits: (D)CML & (D)CML(P) - - - - - - - - - - - - - - -9 Steps

			J1n	**J2n--**

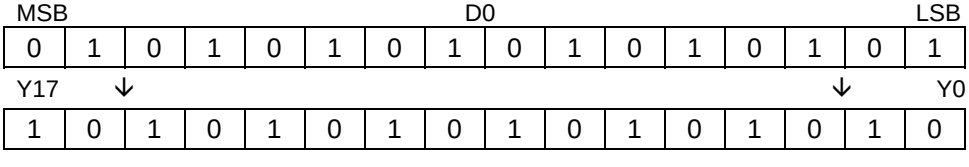
Operands: [S.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.]



♦ Each data bit within the source device [S.] is inverted and then copied to the designated destination [D.].



Block Move

FNC(15)			16 bits: BMOV & BMOV(P) - - - - - - - - - - - - - - - - - 7 Steps
	BMOV	P	

		J1n	J2n--

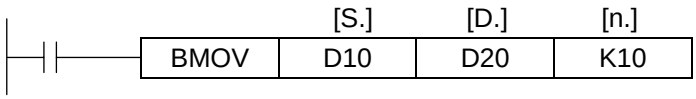
Operands: ←— [S.] —→

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

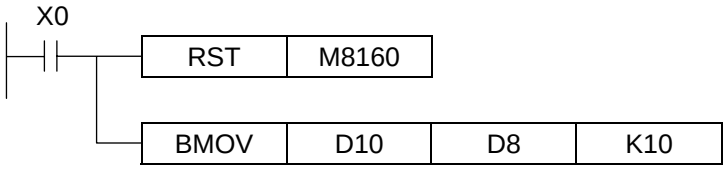
←n→ ←— [D.] —→

n ≤ 128

Flag: None

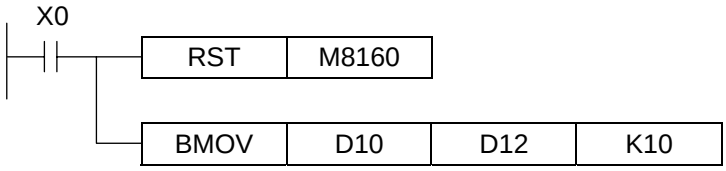


♦ When X0 ON, the move as follows,



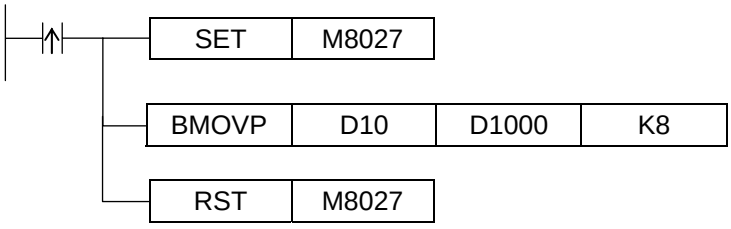
D19	D18	D17	D16	D15	D14	D13	D12	D11	D10
↓⑩	↓⑨	↓⑧	↓⑦	↓⑥	↓⑤	↓④	↓③	↓②	↓①
D17	D16	D15	D14	D13	D12	D11	D10	D9	D8

♦ When transmitting number is repeat, the move as follows,



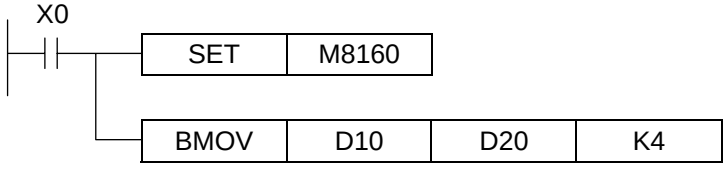
D10	D11	D12	D13	D14	D15	D16	D17	D18	D19
↓⑩	↓⑨	↓⑧	↓⑦	↓⑥	↓⑤	↓④	↓③	↓②	↓①
D12	D13	D14	D15	D16	D17	D18	D19	D20	D21

♦ When M8027 ON, CPU will write the content of [S.] into EEPROM, [D.] only D register can be used.



Note: When M8027 ON, for avoid to damage EEPROM, must be used Pulse Instruction MOV(P).

♦When M8160 ON, the move as follows (M8027 can not be ON) (V2.85 or more is effective),



D14 Lower	D13 Upper	D13 Lower	D12 Upper	D12 Lower	D11Upper	D11 Lower	D10 Upper	D10 Lower
↓	↓	↓	↓	↓	↓	↓	↓	↓
D24 Upper	D24 Lower	D23 Upper	D23 Lower	D22 Upper	D22 Lower	D21 Upper	D21 Lower	D20 Upper	D20 Lower

Fill Move

FNC(16)			16 bits: FMOV & FMOV(P) - - - - - - - - - - - - - - - - - 7 Steps
D	FMOV	P	32 bits: (D)FMOV & (D)FMOV(P) - - - - - - - - - - - -13 Steps

		J1n	**J2n--**

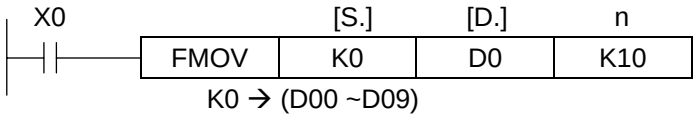
Operands: [S.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.]

n

n ≤ 128



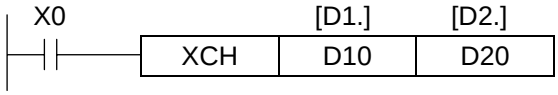
Exchange

FNC(17)			16 bits: XCH & XCH(P) - - - - - - - - - - - - - - - - - - - 5 Steps
D	XCH	P	32 bits: (D)XCH & (D)XCH(P) - - - - - - - - - - - - - - -9 Steps

		J1n	**J2n--**

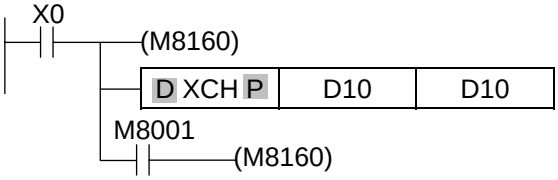
Operands: [D1.] [D2.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z



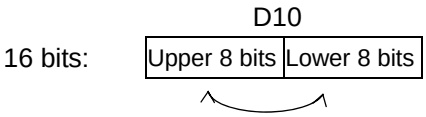
Before : (D10)=100 After : (D10)=200
(D20)=200 (D20)=100

<< Function of Expanded >> SWAP

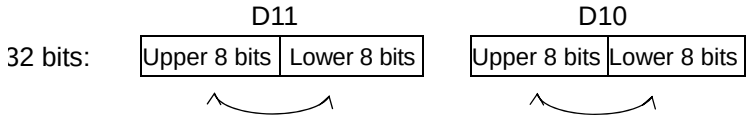


♦ If M8160 ON, [D1.] and [D2.] are the same word device, then the upper 8 bits and the lower 8bits will exchange.

♦ If [D1.] and [D2.] are not the same device, error flag M8067 ON, error code 6705. Error step number is stored to D8069 and not be executed.



Before executing (D10)=0050H=80，After executing (D10)=5000H=20480



Before executing (D11,D10)=87654321H=80，After executing 65872143H

BCD (BINARY CODE TO DECIMAL)

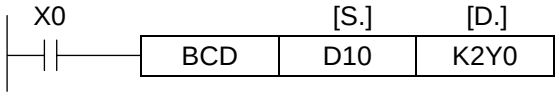
FNC(18)			16 bits: BCD & BCD(P) - - - - - - - - - - - - - - - - - - - 5 Steps
D	BCD	P	32 bits: (D)BCD & (D)BCD(P) - - - - - - - - - - - - - - -9 Steps

			J1n	**J2n--**

Operands: [S.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.]



- The binary source data [S.] is converted into an equivalent BCD number and stored to the destination device [D.].
- If the converted BCD number exceeds the operational ranges of 0 to 9999 (16 bit operation) or 0 to 99999999 (32 bit operation), an error will occur. Error flag M8067 ON, error code 6705 and error step number stored to D8069. Program will be executed continuously, but result will not be stored to [D.]
- This instruction can be used to output data to a seven segment display directly.

BIN (DECIMAL CODE TO BINARY)

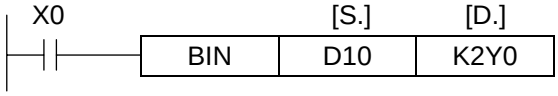
FNC(19)			16 bits: BIN & BIN(P) - 5 Steps
D	BIN	P	32 bits: (D)BIN & (D)BIN(P) - - - - - - - - - - - - - - - - -9 Steps

			J1n	**J2n--**

Operands: [S.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.]



- The BCD source data [S.] is converted into an equivalent binary number and stored at the destination device [D.].
- If the source data is not provided in a BCD format, an error will occur. Error flag M8067 ON, error code 6705 and error step number stored to D8069.
- The device [S.] can't be used constant K/H.

Addition

FNC(20)			16 bits: ADD & ADD(P) - - - - - - - - - - - - - - - - - - - 7 Steps
D	ADD	P	32 bits: (D)ADD &(D)ADD(P) - - - - - - - - - - - - - - 13 Steps

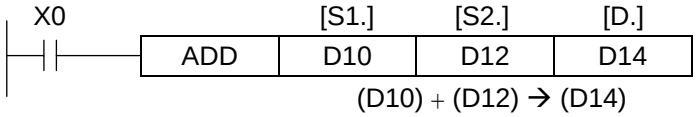
		J1n	J2n--

Operands: [S1.][S2.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.]

Flag: M8020, M8021, M8022



(D10) + (D12) → (D14)

- The data contained within the source devices [S1.], [S2.] is added and the result stored to specified destination devices [D.].
- All calculations are algebraically processed, i.e. 5+(-8) = -3.
- If the result of a calculation is “0”, then zero flag M8020 ON.
- If the result exceeds 32,767 (16 bit limit) or 3,147,483,647 (32 bit operation), the carry flag M8022 ON.
- If the result exceeds -32,767 (16 bit limit) or -2,147,483,647 (32 bit limit), the borrow flag M8021 ON.

Subtraction

FNC(21)			16 bits: SUB & SUB(P) - - - - - - - - - - - - - - - - - - - 7 Steps
D	SUB	P	32 bits: (D)SUB &(D)SUB(P) - - - - - - - - - - - - - - 13 Steps

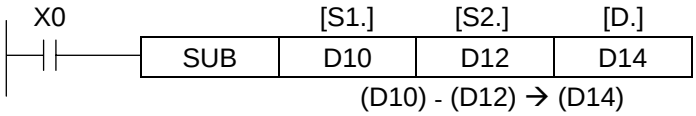
		J1n	J2n--

Operands: [S1.][S2.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

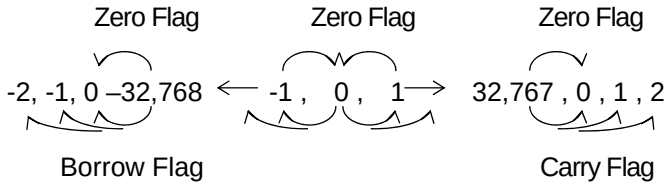
[D.]

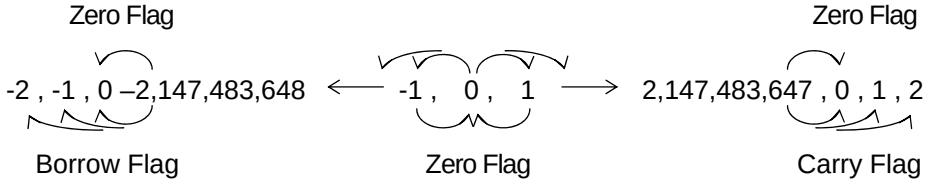
Flag: M8020, M8021, M8022



(D10) - (D12) → (D14)

- Content of [S1.] subtract content of [S2.], and the result stored to specified destination devices [D.].
- All calculations are algebraically processed, i.e. 5 - 8 = -3.
- The MSB of devices is sign (0:Positive, 1:Negative).





Multiplication

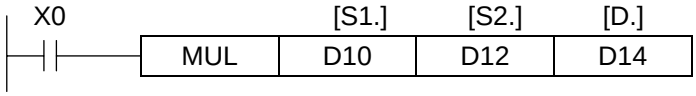
FNC(22)			16 bits: MUL & MUL(P) - - - - - - - - - - - - - - - - - - - 7 Steps
D	MUL	P	32 bits: (D)MUL &(D)MUL(P) - - - - - - - - - - - - - - 13 Steps

		J1n	J2n--

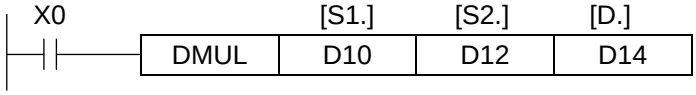
Operands: ← [S1.][S2.] →

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

← [D.] →



16 bit: (D10) × (D12) → (D15, D14)



32 bit: (D11,D10) × (D13,D12) → (D17,D16,D15,D14)

♦ The primary source [S1.] is multiplied by the secondary source [S2.]. The result is stored to destination [D.].

Division

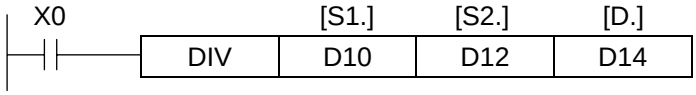
FNC(23)			16 bits: DIV & DIV(P) - - - - - - - - - - - - - - - - - - - 7 Steps
D	DIV	P	32 bits: (D)DIV & (D)DIV(P) - - - - - - - - - - - - - - -13 Steps

		J1n	J2n--

Operands: ← [S1.][S2.] →

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

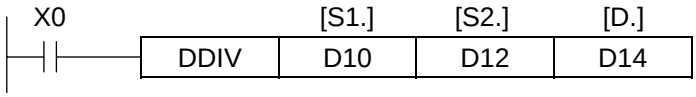
← [D.] →



Dividend divisor quotient remainder

(D10) ÷ (D12) → (D14) . . . . . (D15)

16 bits 16 bits 16 bits 16 bits



Dividend divisor quotient remainder

(D11,D10) ÷ (D13,D12) → (D15,D14).……(D17,D16)

32 bits 32 bits 32 bits 32 bits

♦ The primary source [S1.] is divided by the secondary source [S2.]. The result is stored to destination [D.].

♦ If value of source device [S2.] is “0” (zero), then an operation error is executed. Error code 6706 and error step number stored to D8069, the program operation is cancelled.

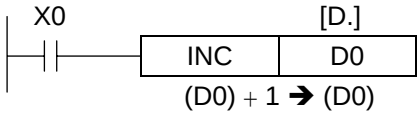
♦ V1.17 edition：If value of source device [S2.] is “0” (zero), then will not execute and directly jump to next instruction.

Increment

FNC(24)			16 bits: INC & INC(P) - 3 Steps
D	INC	P	32 bits: (D)INC & (D)INC(P) - - - - - - - - - - - - - - - -5 Steps

		J1n	**J2n--**

	K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z
Operands:	⟵					[D.]			⟶



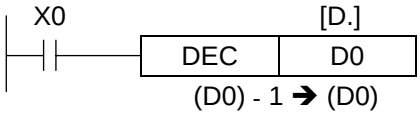
- ♦ On every execution of the instruction, the device specified as the destination [D.] and its current value increased 1.
- ♦ In 16 bit operation, when +32,767 is reached, the next execution will write a value of -32,768 to destination device.
- ♦ In 32 bit operation, when +2,147,483,647 is reached, the next execution will write -2,147,483,648 to destination device.
- ♦ The carry, zero and borrow flag are unaffected in the operation.

Decrement

FNC(25)			16 bits: DEC & DEC(P) - 3 Steps
D	DEC	P	32 bits: (D)DEC & (D)DEC(P) - - - - - - - - - - - - - - - -5 Steps

		J1n	**J2n--**

	K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z
Operands:	⟵					[D.]			⟶



- ♦ On every execution of the instruction, the device specified as the destination [D.] and its current value decreased 1.
- ♦ In 16 bit operation, when -32,768 is reached, the next execution will write a value of +32,767 to destination device.
- ♦ In 32 bit operation, when -2,147,483,648 is reached, the next execution will write +2,147,483,647 to destination device.
- ♦ The carry, zero and borrow flag are unaffected in the operation.

Logical AND

FNC(26)			16 bits: WAND & WAND(P) - - - - - - - - - - - - - - - - 7 Steps
D	WAND	P	32 bits: (D)WAND &(D)WAND(P) - - - - - - - - - - - 13 Steps

		J1n	J2n--

Logical OR

FNC(27)			16 bits: WOR & WOR(P) - - - - - - - - - - - - - - - - - - 7 Steps
D	WOR	P	32 bits: (D)WOR & (D)WOR(P) - - - - - - - - - - - - -13 Steps

		J1n	J2n--

Logical XOR

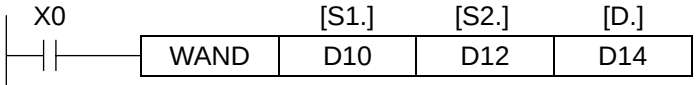
FNC(28)			16 bits: WXOR & WXOR(P) - - - - - - - - - - - - - - - - 7 Steps
D	WXOR	P	32 bits: (D)WXOR &(D)WXOR(P) - - - - - - - - - - - 13 Steps

		J1n	J2n--

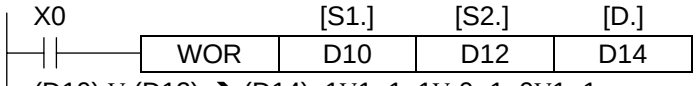
Operands: ← [S1.][S2.] →

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

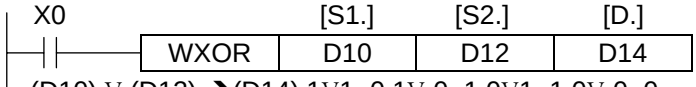
← [D.] →



(D10) Λ (D12) ➔ (D14), 1Λ1=1, 1Λ 0=0, 0Λ1=0,



(D10) V (D12) ➔ (D14), 1V1=1, 1V 0=1, 0V1=1,



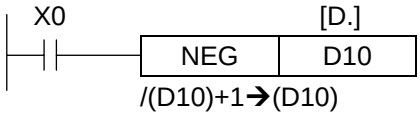
(D10) V (D12) ➔(D14),1V1=0,1V 0=1,0V1=1,0V 0=0.

Negation

FNC(29)			16 bits: NEG & NEG(P) - - - - - - - - - - - - - - - - - - - 3 Steps
D	NEG	P	32 bits: (D)NEG & (D)NEG(P) - - - - - - - - - - - - - - -5 Steps

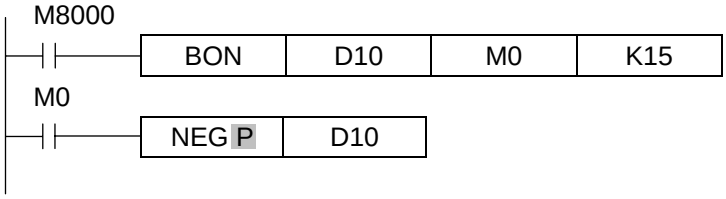
		J1n	J2n--

Operands:	K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z
	←				[D.]				→



- When X0 ON, the selected device [D.] is inverted. (“1”→”0”,“0”→”1”)
- When this is complete, a further binary 1 is added to the bit pattern. The result is become a negative number or a negative number will become a positive.

< Example >> Absolute Value of Negative



<< Note of Negation >>

(D 10)=2

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

(D 10)=1

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

(D 10)=0

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(D 10)= -1 → $(\overline{D\ 10})+1=1$

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	→	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

(D 10)= -2 → $(\overline{D\ 10})+1=2$

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	→	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

(D 10)= -32,765 → $(\overline{D\ 10})+1=32,765$

1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	→	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1

(D 10)= -32,766 → $(\overline{D\ 10})+1=32,766$

1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	→	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

(D 10)= -32,767 → $(\overline{D\ 10})+1=32,767$

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	→	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

(D 10)= -32,768 → $(\overline{D\ 10})+1=-32,768$

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	→	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Rotation Right

FNC(30)			16 bits: ROR & ROR(P) - - - - - - - - - - - - - - - - - - - 5 Steps
D	ROR	P	32 bits: (D)ROR & (D)ROR(P) - - - - - - - - - - - - - - -9 Steps

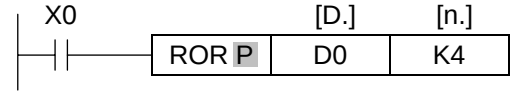
		J1n	J2n--

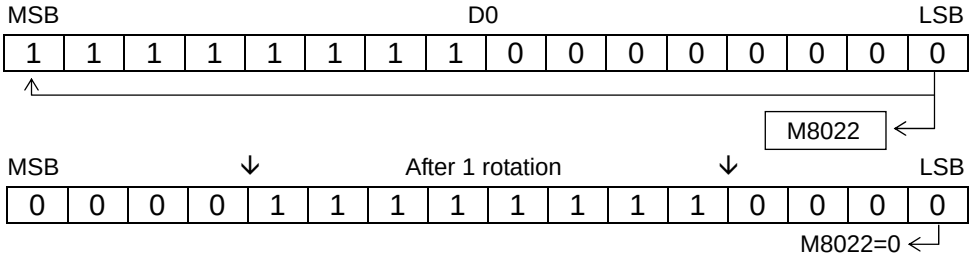
	K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z
Operands:	← n →		←			[D.]			→

16bit : n ≤ 16

32bit : n ≤ 32

Flag: M8022





♦ After rotation right, the LSB of specified devices is shifted into carry flag M8022.

Rotation Left

FNC(31)			16 bits: ROL & ROL(P) - - - - - - - - - - - - - - - - - - - 5 Steps
D	ROL	P	32 bits: (D)ROL & (D)ROL(P) - - - - - - - - - - - - - - -9 Steps

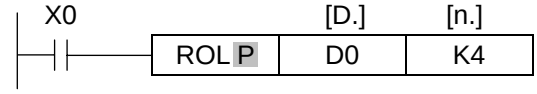
		J1n	J2n--

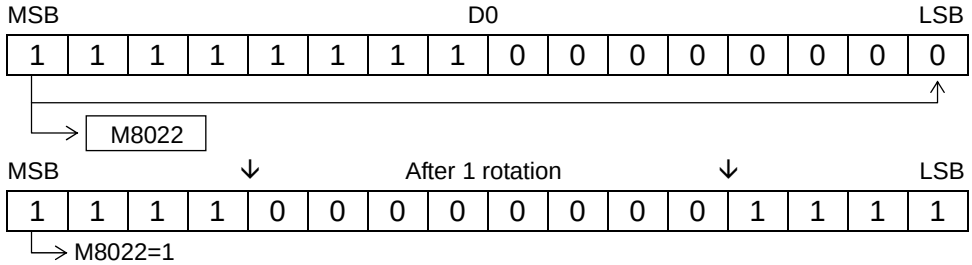
	K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z
Operands:	← n →		←			[D.]			→

16bit : n ≤ 16

32bit : n ≤ 32

Flag:





♦ After rotation left, the MSB of specified devices is shifted into carry flag M8022.

Rotation Right with Carry

FNC(32)			16 bits: RCR & RCR(P) - - - - - - - - - - - - - - - - - - - 5 Steps
D	RCR	P	32 bits: (D)RCR & (D)RCR(P) - - - - - - - - - - - - - - -9 Steps

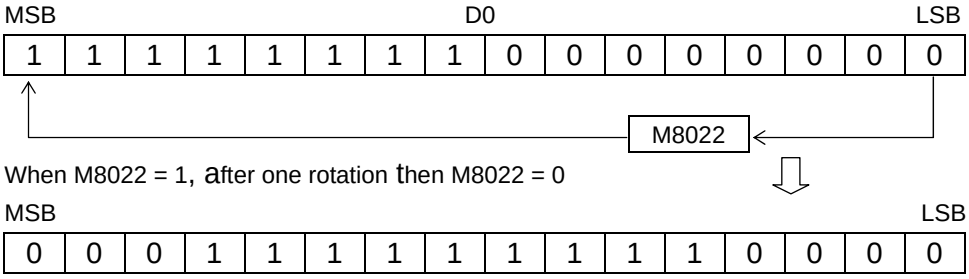
		J1n	J2n--

	K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z
Operands:	← n →		←			[D.]			→

16bit : n ≤ 16

32bit : n ≤ 32

Flag:



When M8022 = 1, after one rotation then M8022 = 0

Rotation Left with Carry

FNC(33)			16 bits: RCL & RCL(P) - - - - - - - - - - - - - - - - - - - 5 Steps
D	RCL	P	32 bits: (D)RCL & (D)RCL(P) - - - - - - - - - - - - - - -9 Steps

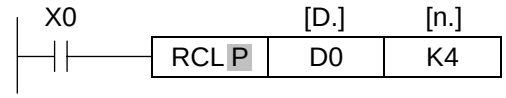
		J1n	J2n--

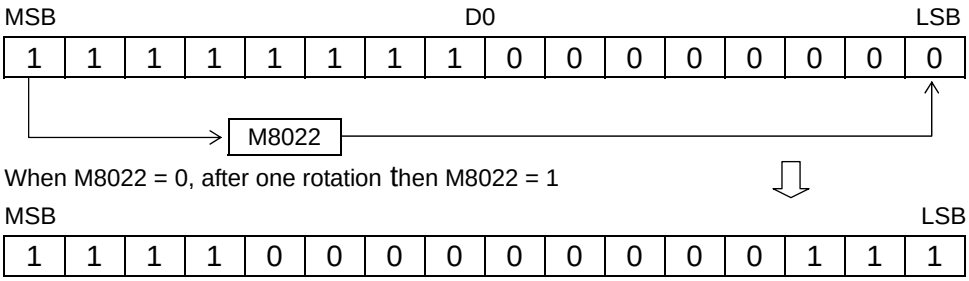
	K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z
Operands:	← n →		←			[D.]			→

16bit : n ≤ 16

32bit : n ≤ 32

Flag:





When M8022 = 0, after one rotation then M8022 = 1

Shift Right

FNC(34)			16 bits: SFTR & SFTR(P) - - - - - - - - - - - - - - - - - - 9 steps				
	SFTR	P				**J1n**	**J2n--**

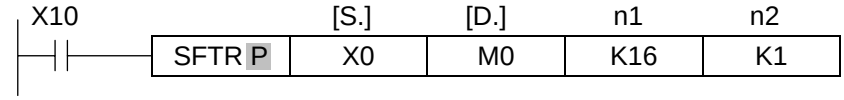
Shift Left

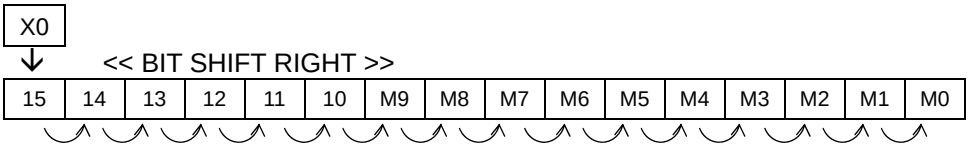
FNC(35)			16 bits: SFTL & SFTL(P) - - - - - - - - - - - - - - - - - - 9 steps				
	SFTL	P				**J1n**	**J2n--**

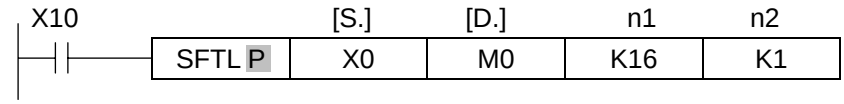
Operands:

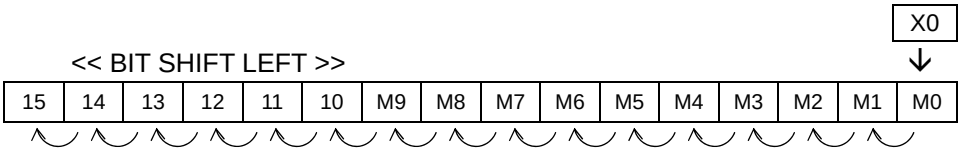
	← [S.] →				
K.H.		X	Y	M	S
← n →	n2 ≤ n1 ≤ 256	← [D.] →			

Flag:

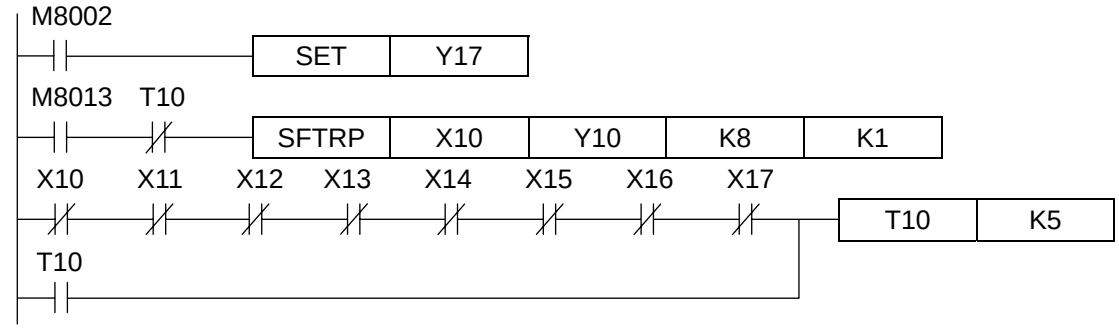








Example I/O Test: Wiring X10 ↔Y10 … X17 ↔Y17



Word Shift Right

FNC(36)			16 bits: WSFR & WSFR(P) - - - - - - - - - - - - - - - - - 9 steps
	WSFR	P	

		J1n	J2n--

Word Shift Left

FNC(37)			16 bits: WSFL & WSFL(P) - - - - - - - - - - - - - - - - - 9 steps
	WSFL	P	

		J1n	J2n--

Operands:

[S.]

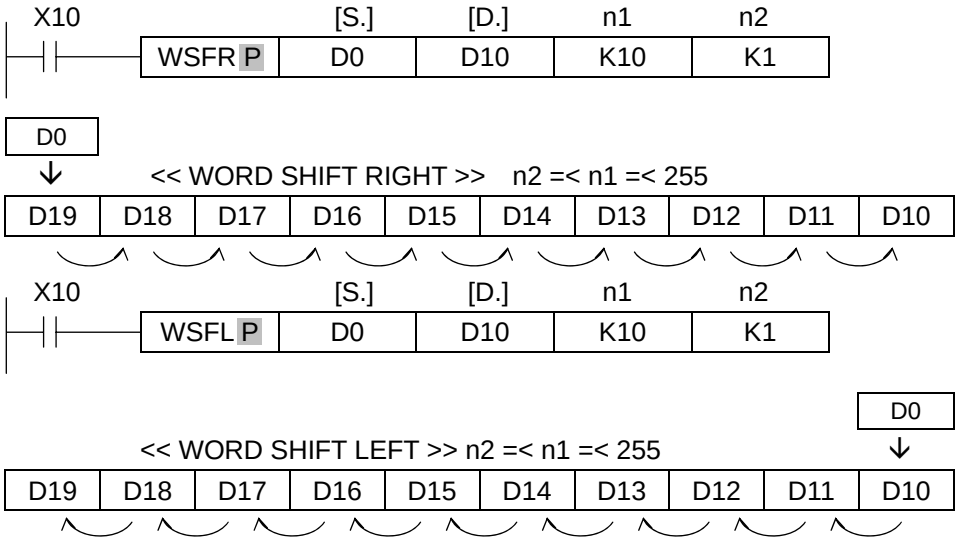
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

n

[D.]

n2 ≤ n1 ≤ 256

Flag:



Shift Register Write

FNC(38)			16 bits: SFWR & SFWR(P) - - - - - - - - - - - - - - - - - 7 Steps
	SFWR	P	

		J1n	J2n--

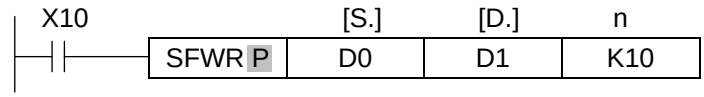
Operands: [S.]

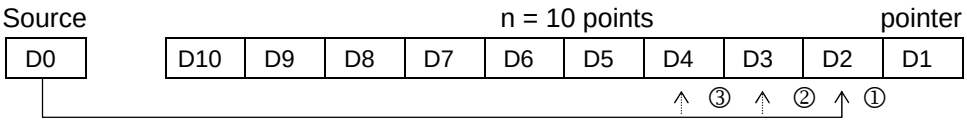
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

n [D.]

$2 \le n \le 256$

Flag:





- ♦ When X10 OFF → ON, content of D0 stored into D2 and D1=”1”. When next rising pulse, content of D0 stored into D3 and D1=”2”, the position of insertion into the stack is automatically calculated by controller.
- ♦ If content of [D.] exceeds the value “n-1” (n is length of the FIFO stack), then insertion into the FIFO stack is stopped. The carry flag M8022 is turned ON.
- ♦ Before starting to use a FIFO stack, ensure that contents of the head address register [D.] are equal to “0”.

Shift Register Read

FNC(39)			16 bits: SFRD &SFRD(P) - - - - - - - - - - - - - - - - - 7 Steps
	SFRD	P	

		J1n	J2n--

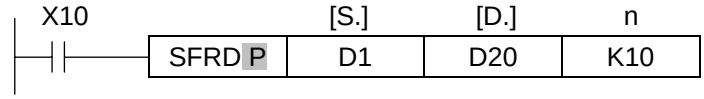
Operands: [S.]

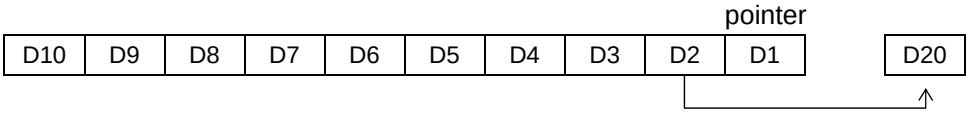
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

n [D.]

$2 \le n \le 256$

Flag:





- ♦ When X10 OFF → ON, content of D2 stored into D20 and content of D1 decreased 1 (D1=D1-1).
- ♦ When contents of source device [S.] are equal to “0”, i.e. the FIFO stack is empty, zero flag M8020 is turned on.
- ♦ This instruction will always read the source data from the register [S.]+1.

Zone Reset

FNC(40)		16 bits: ZRST(P) - 5 steps
	ZRST	P

		J1n	J2n--

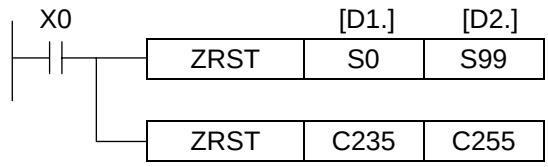
Operands: ← [D1.][D2.] →

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands: ← [D1.] [D2.] →

X	Y	M	S

Flag:



- ♦ The range of specified devices are reset, for data devices, the current value is set to “0”, and for bit elements, the bit status are turned OFF.
- ♦ The specified device range cannot contain mixed devices types, i.e. if C00 specified as the first destination devices [D1.], then cannot paired with T99 as the second devices.
- ♦ If [D1.] is bigger than (>) [D2.], then only [D1.] is reset.

Decode

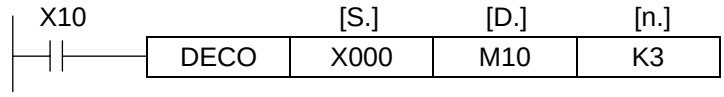
FNC(41)		16 bits: DECO(P) - 7 steps
	DECO	P

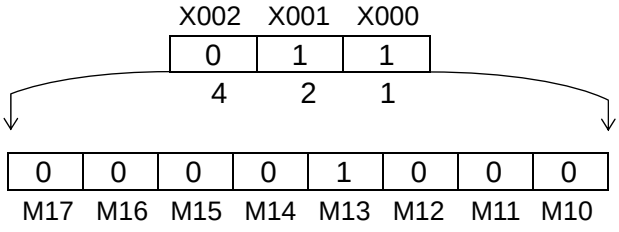
		J1n	J2n--

Operands: ← [S.] →

X	Y	M	S	K.H	T	C	D	V,Z

← [D.] → ⇩ n = 1 – 8 ← [D.] →





- ♦ If the specified device [D.] is T, C or D, then n ≤ 4.
- ♦ If the sources all are “0”, then M10 set to “1”.

Encode

FNC(42)			16 bits: ENCO(P) - 7 steps
	ENCO	P	

		J1n	J2n--

Operands:

[S.]					[S.]			
X	Y	M	S	K.H	T	C	D	V,Z

[D.]

n = 1 – 8

Flag:

X10

	[S.]	[D.]	[n.]
ENCO	M10	D10	K3

M17	M16	M15	M14	M13	M12	M11	M10
0	0	0	0	1	0	0	0
7	6	5	4	3	2	1	0

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

MSB D10 LSB

- If the specified device [S.] is T, C or D, then n ≤ 4.
- The number of active (ON) bits within the source device [S.] is more than one, only the lowest bit “1” is effective.
- If bits of source device [S.] all are “0”, then error occurred.

Sum

FNC(43)			16 bits: SUM(P) - 5 steps
D	SUM	P	32 bits: (D)SUM(P) -9 steps

		J1n	J2n--

Operands:

[S.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.]

Flag:

X00

	[S.]	[D.]
SUM	D0	D2

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

D 0

8 4 2 1

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

D 2

- The number of active (ON) bits within the source device [S.], i.e. bits which have a value of “1” are counted. The count is stored in the destination device [D.].
- If there is no bit as 0, then zero flag M8020 ON.

Bit On Check

FNC(44)			16 bits: BON(P) - 7 steps
D	BON	P	32 bits: (D)BON(P) - 13 steps

		J1n	**J2n--**

Operands: [S.]

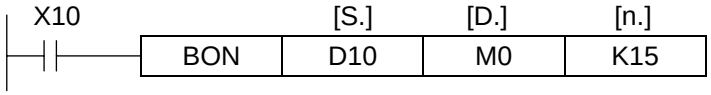
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[n.] = 0~15 or 0~31

Operands: [D.]

X	Y	M	S

Flag:



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

Bit15,D10=0, then M0 = OFF. LSB

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

Bit15,D10=1, then M0 = ON LSB

Mean

FNC(45)			16 bits: MEAN(P) - 7 steps
	MEAN	P	

		J1n	**J2n--**

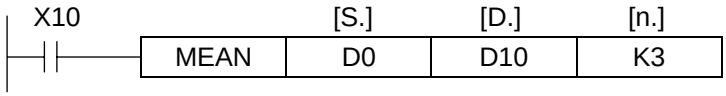
Operands: [S.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands: n [D.]

[n]=1-64

Flag:



♦ [(D0) + (D1) + (D2)] / 3 → (D10)

Annunciator Set

FNC(46)			16 bits: ANS - 7 steps
	ANS		

Reserved

Annunciator Reset

FNC(47)			16 bits: ANR(P) - 1 steps
	ANR		

Reserved

Square Root

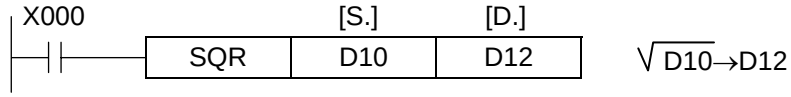
FNC(48)			16 bits: SQR(P) - 5 steps	**J2n--**
D	SQR	P	32 bits: (D)SQR(P) -9 steps	

Operands: ⟷ [S.] [S.] ⟷

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands: [D.] ⟷

Flag: M8020, M8021, M8022



- ♦ [S.] must be positive. When it is negative, error flag M8067 ON, and stop executing.
- ♦ When the result with decimal fraction, don't care it; but borrow flag M8021 will ON.
- ♦ When result is 0, zero flag M8020 will ON.

Float

FNC(49)			16 bits: FLT(P) - 5 steps
D	FLT	P	32 bits: (D)FLT(P) -9 steps

			J2n--

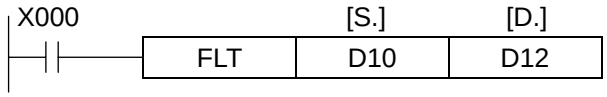
Operands: [S.] ⟷

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

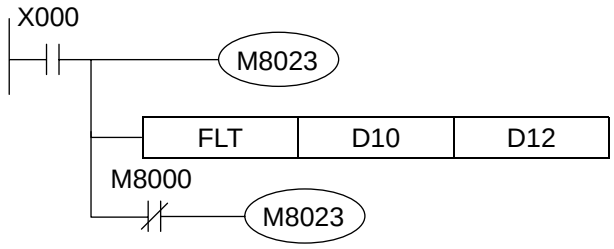
Operands: [D.] ⟷

Flag: M8020, M8021, M8022

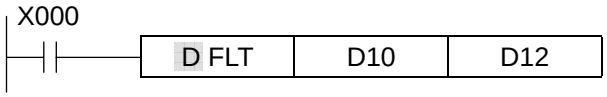
♦ FLT Instruction is converted command between BIN integer and binary floating data. Because constant K, H will automatically convert when floating data operate, then not fit this instruction



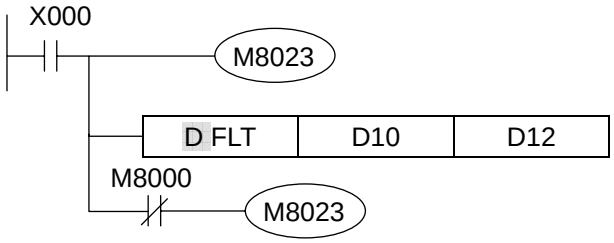
(D10) → (D13、D12)
BIN integer binary floating data



(D11、D10) → (D12)
binary floating data BIN integer
Decimal fraction don't care



(D11、D10) → (D13、D12)
BIN integer binary floating data



(D11、D10) → (D13、D12)
binary floating data BIN integer
Decimal fraction don't care

♦ When M8023 = ON, execute binary floating data → BIN integer。
When M8023 = OFF, then execute BIN integer → binary floating data.

♦ Binary floating data → BIN integer, the operating result is decimal fraction, don't care it, but M8021 / M8022 will ON; when result is 0, M8020 will ON

Output & Input Refresh

FNC(50)			16 bits: REF(P) - 5 steps			
	REF	P			**J1n**	**J2n--**

Operands:

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

|← n →|

Operands:

X	Y	M	S

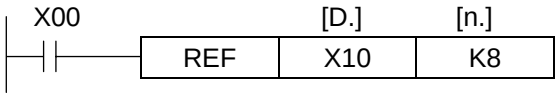
|←— [D.] —→|

[D.] should always be a multiple of 10, i.e. 00,10..

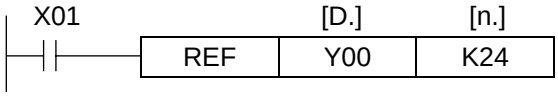
[n.] should always be a multiple of 8, i.e. 8,16,24..

♦ PLC input all refresh before program STEP 0 execute; output is executed after END or FEND instruction. It is not changed in performing process. If it needs immediately input data or output performing result in the performing process, then have to use output & input refresh instruction.

<< Input Fresh >> only X10 – X17 to be flashed



<< Output Fresh >> refresh Y00-Y07, Y10-Y17, Y20-Y27.



♦ In interruption program, FNC(50) REF command can not be used.

Refresh and Filter Adjustment

FNC(51)			16 bits: REFF(P) - 3 steps			
	REFF	P			**J1n**	**J2n--**

Operand: [n.] = 0 - 60

♦ To avoid noise interference, PLC input relay all designed with hardware RC filter to adjust software filter time.

♦ This instruction only change X00-X07 software filter time, i.e., content of D8020. If it has to change other input point filter time, please use MOV instruction.

Matrix

FNC(52)			16 bits: MTR - 9 Steps
	MTR		

		J1n	J2n--

Operands:

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

← n →

Operands ←[S.]→ ←[D1.]→

X	Y	M	S

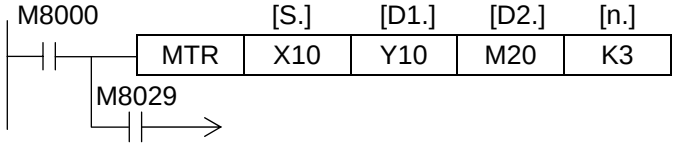
← [D2.] →

Operand: (S.): X00, X10, X20, X30 - - - - - - - X160, X170.

(D1.): Y00, Y10, Y20, Y30 - - - - - - Y160, Y170.

(D2.): Y, M, S multiple of 10, i.e. 00, 10, 20 etc.

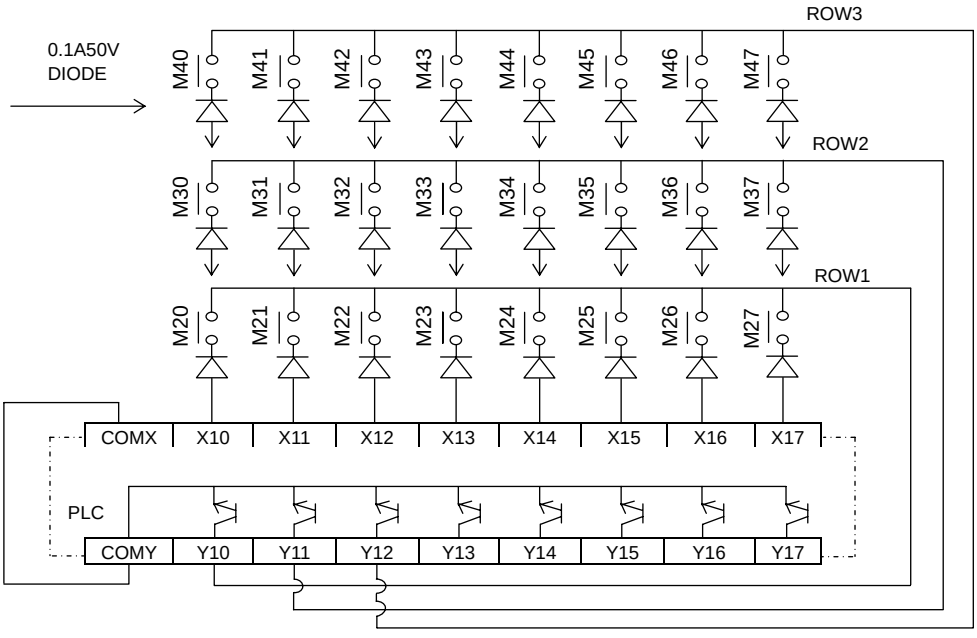
(n.): K, H. n=2 ~ 8.

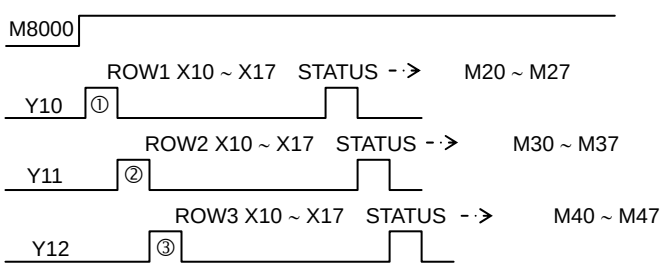


♦ MTR instruction allows 8 consecutive input devices [S.] to be used multiple (n) times. The result was stored in (D2.).
(S.): Head address of the input devices. (n.): row numbers.
(D1.): Head address of the output trigger devices.
(D2.): Head address of the matrix table.

♦ After completion of full reading of the matrix, the complete flag M8029 to be turned ON. This flag will be automatically reset when this instruction is executed.

♦ This instruction can be used once, and only the transistor module can be selected.





Set by High Speed Counter

FNC(53)			32 bits: HSCS - 13 Steps
D	HSCS		

		J1n	J2n--

Operands: ⟵ [S1.] ⟶

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

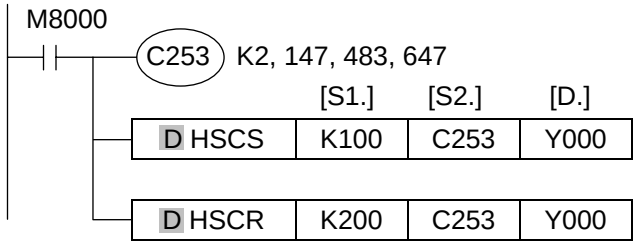
⟵⟶ [S2.] : C235~C255

Operands:

X	Y	M	S

⟵ [D.] ⟶ When [D.], can use Index to assign I010~I060 to interrupt.

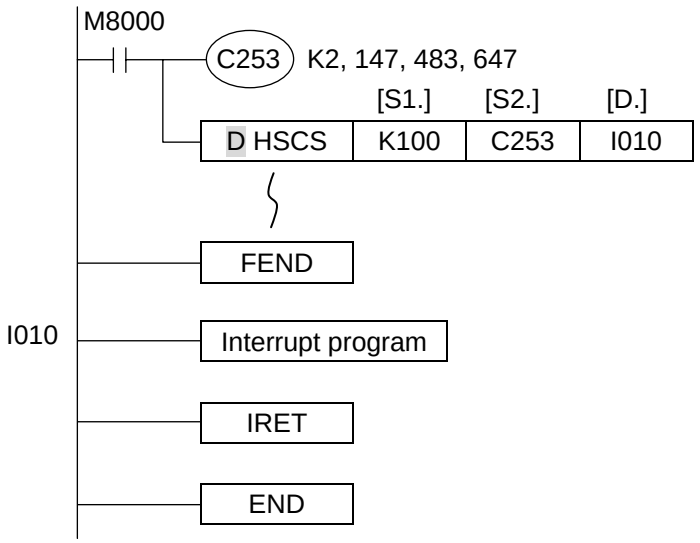
Flag:



♦ When use FNC53, operate external output action by interrupt. When current value of C253 changed from 99 to 100 and from 101 to 100, Y000 will be set. When current value of C253 is changed from 199 to 200 and from 201 to 200, Y000 OFF.

♦ This command is specialized instruction of 32 bits, please input D HSCS command.

♦ Only can use FNC53, FNC54, FNC55 once.



♦[D.] of D HSCS can assign I0 □ 0 = (□=1~6)(□=1~6 can not be reuse.)

♦Therefore, when current value of High Speed Counter which is assigned by [S2.] is as same as the value which is assigned by [S1.], interrupt main program and jump to execute I0 □ 0 interrupt program immediately.

♦When Special auxiliary relay M8059 ON, I010~I060 interrupt are all prohibited.

Reset by High Speed Counter

FNC(54)		
D	HSCR	

32 bits: HSCR - 13 Steps

		J1n	J2n--

Operands: ⟵ [S1.] ⟶

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

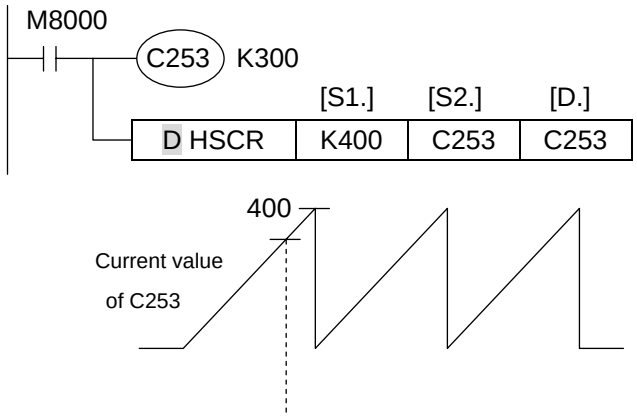
⟵⟶ [S2.] : C235~C255

Operands:

X	Y	M	S

⟵ [D.] ⟶ Can assign [D.] and [S2.] are the same High Speed Counter.

Flag:



♦ When current value of C253 is 400, C253 will be cleared immediately. Current value will become 0, and output contact will not act.

♦ This command is specialized instruction of 32 bits, so have to use D HSCR.

Zone Compare For High Speed Counter

FNC(55)		
D	HSZ	

32 bits: HSZ- 17 Steps

Operands: ⟵ [S1.][S2.] ⟶

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

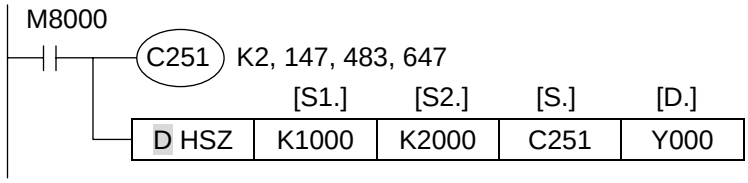
⟵⟶ [S.] : C235~C255

Operands:

X	Y	M	S

⟵ [D.] ⟶

Flag:



<Compare action of input>

K1000>C251 current value	Y000	ON
K1000≦C251 current value≦K2000	Y001	ON
K2000<C251 current value	Y002	ON

♦ This command is specialized instruction of 32 bits, so have to use D HSZ.

♦ Content of [S1.] and [S2.] is according to [S1.] ≦ [S2.] .

♦ When use FNC55, operate external output by Interrupt. Output will act without effect by scan-cycle.

Speed Detect

FNC(56)			16 bits: SPD - 7 Steps
	SPD		

		J1n	J2n--

Operands: (S1.): X000~X005. When C251 is used, X02 and X03 can not be used.

Operands: ←——— [S2.] ———→

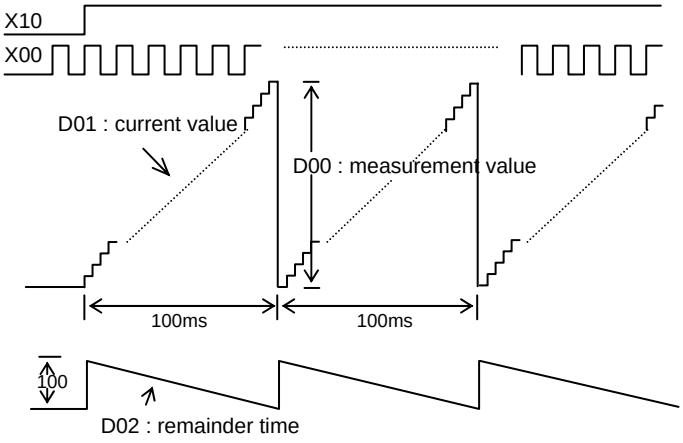
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

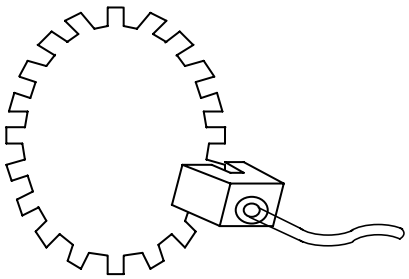
Flag:M8029 ←——— [D.] ———→

	[S1.]	[S2.]	[D.]
SPD	X00	K100	D00

- ♦ The input pulse assigned by [S1.], and the [S2.] assign measurement time, the result will be stored at [D.].
- ♦ This will automatic occupy 3 word devices from the head address of [D.]. (D00~D02)
- ♦ This example D01 count up the pulse number of X00 (OFF→ON), and put the result into D00 at 100msec after. Then reset D01to “0” and start counting again.
- ♦ D02 is used to measurement remainder time.
- ♦ The counting pulse amount of the assign time can’t be more than 65535
- ♦ Following formula can calculated RPM
  RPM : N = (D00×60) × 1000 / n × t
  n: (pulse/revolution), t: (measurement time).
- ♦ The pulse frequency of (X00-X05) is same with HSC.
- ♦ If input relay (X00-X05) is assigned by the SPD, they can’t be used to other purpose or interrupt input point.
- ♦ If pulse output assign Y00, then X00 can’t be used; if assign Y01, then X01 can’t be used.
- ♦ V1.45 or more, add complete flag M8029, easily reach many data of continuous measurement, then count an average value.

(i) measure frequency mode





(ii) measure pulse width mode

- ♦ The content of [S2.]=”0” only one pulse width then can measurement speed N pps(pulse/second) 。
- ♦ This example speed N store at D01,D00 。

Pulse Output

FNC(57)			32 bits: (D)PLSY- 13 steps
D	PLSY		

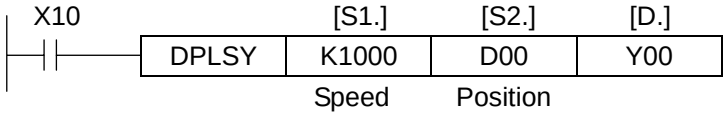
		J1n	J2n--

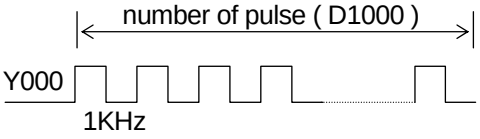
Operands: [S1.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.] : Y00 – Y03 [S2.]

Flag: M8029





- ♦ This instruction is pulse output without slope.
- ♦ [D.] assign pulse output point
  [S1.] assign output frequency (10~200,000Hz).
  [S2.] it will occupy continuous 100 words from assigned [S2.]. In this example, it occupies D1000~D1099.
  [S2.]+1, [S2.]+0 : number of output pulses [S2.]+3, [S2.]+2 : system reserved
  [S2.]+5, [S2.]+4 : start address [S2.]+7, [S2.]+6 : absolute address(for monitor)
  [S2.]+9, [S2.]+8 : increment address(for monitor)
- ♦ DPLSY is used to output a consecutive pulse. 32 bits range: 1 ~ 2,147,483,647 pulses.
- ♦ If [S2.]+1, [S2.]+0 are assigned to “0”, it will continue to generate pulse.
- ♦ It is fixed to 32 bits operation. If it is assigned to 16 bits operation, then error 6509 will be occurred.
- ♦ The pulse duty cycle is 50% ON 50% OFF.
- ♦ Value of [S2.]+1, [S2.]+0 can be changed during execution, but the new will not be effective until current operation has been completed, and complete flag M8029 set to ON .
- ♦ This instruction can be used once, and only the transistor module can be selected.

Pulse Width Modulation

FNC(58)			16 bits: PWM - 7 steps
	PWM		

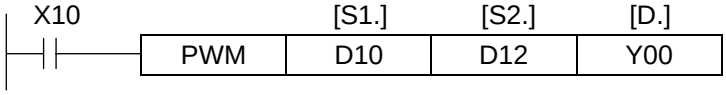
		J1n	J2n--

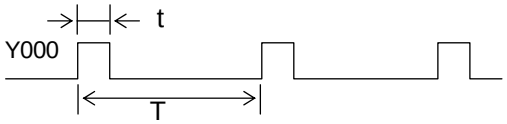
Operands: [S1.][S2.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.] : Y00 – Y07

Flag: None





- ♦ [S1.]: ON duty width (t). Y00 - Y01 range (0 - 32,767) x 0.01ms; Y02 - Y07 range: (0 - 32,767 msec)
- ♦ [S2.]: (T). Y00 - Y01 range (0 - 32,767) x 0.01ms ; Y02 - Y07 range: (0 - 32,767 msec)
- ♦ [D.]: Output point (Y). (by interrupt handing)
- ♦ If value of [S1.] is more than value of [S2.], then error occurred.
- ♦ This instruction is applicable for transistor module.

PULSE OUTPUT WITH SLOPE

FNC(59)			
D	PLSR		32 bits: (D)PLSR- 17 steps

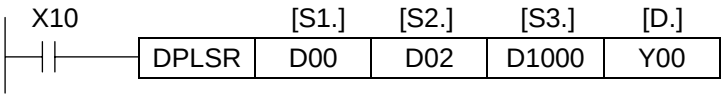
		J1n	J2n--

Operands: ← [S1.] →

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.] : Y00 – Y03 ← → [S2.][S3.]

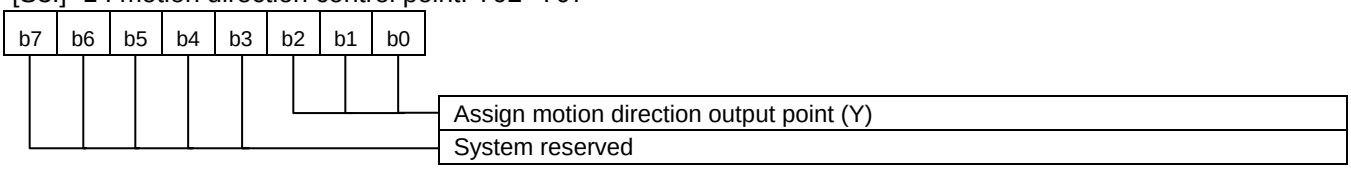
Flag: M8029



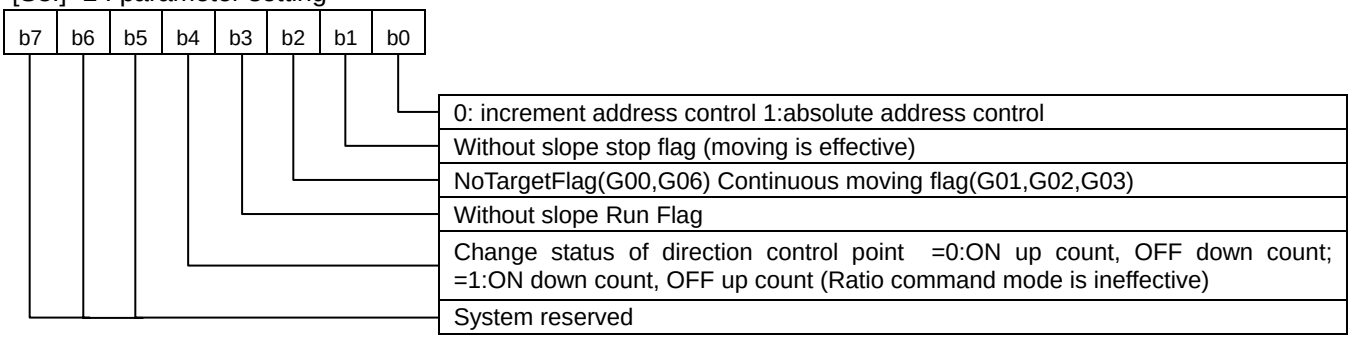
♦ [D.] assign pulse output point. Assign to Y04=pseudo axis (no real pulse output)
[S1.] assign output frequency.(10 ~ 200,000pps)
[S2.] assign number of output pulse. It will occupy continuous 8 words start from assigned [S2.]. In this example, it occupies D02~D09
[S3.] It will occupy continuous 100 words start from assigned [S3.]. In this example, it occupies D1000~D1099.
[S3.]+0 : motion mode: command value 0~99 as well as G00~G99

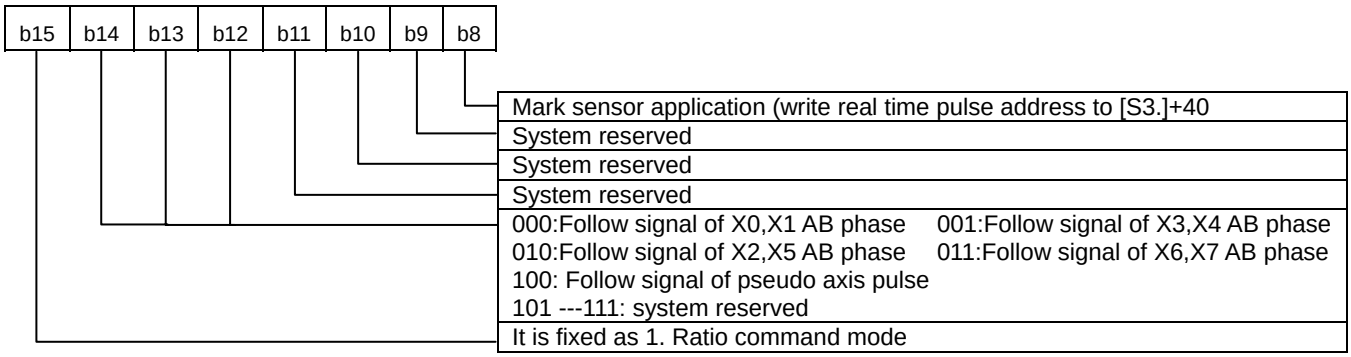
Command value	Content
00	Single position motion
01	Linear interpolation (J2nB only) n=2,4
02	Circular interpolation CW (J2nB only) n=2,4
03	Circular interpolation CCW (J2nB only) n=2,4
06	Ratio command(electronic gear must be fraction. numerator< denominator)
28	Zero Return

[S3.]+1 : motion direction control point: Y02~Y07

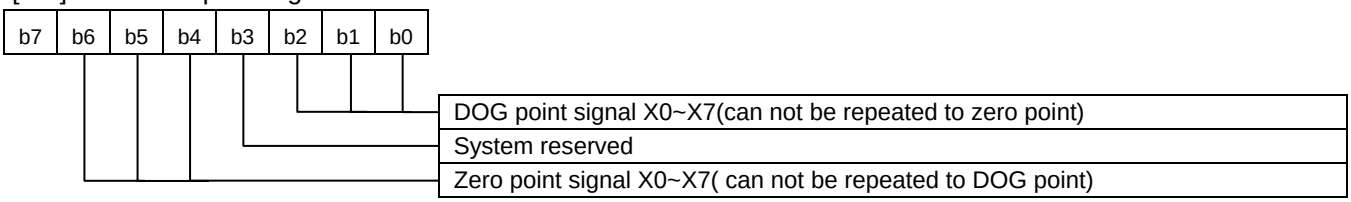


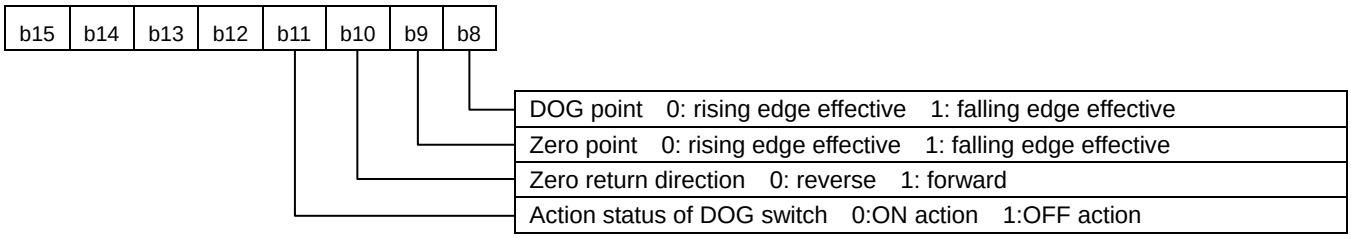
[S3.]+2 : parameter setting





[S3.]+3 : system reserved
[S3.]+5, [S3.]+4 : start address(for monitor)
[S3.]+7, [S3.]+6 : absolute address(for monitor)
[S3.]+9, [S3.]+8 : increment address(for monitor)
[S3.]+11, [S3.]+10 : the rest of pulses(for monitor)
[S3.]+13, [S3.]+12 : target address(for monitor)
[S3.]+15, [S3.]+14 : current speed(for monitor)
[S3.]+17, [S3.]+16 : maximum speed
[S3.]+19, [S3.]+18 : system reserved
[S3.]+20 : bias speed(pps)
[S3.]+21 : system reserved
[S3.]+22 : acceleration time(ms)
[S3.]+23 : deceleration time (ms)
[S3.]+24 : DOG point signal





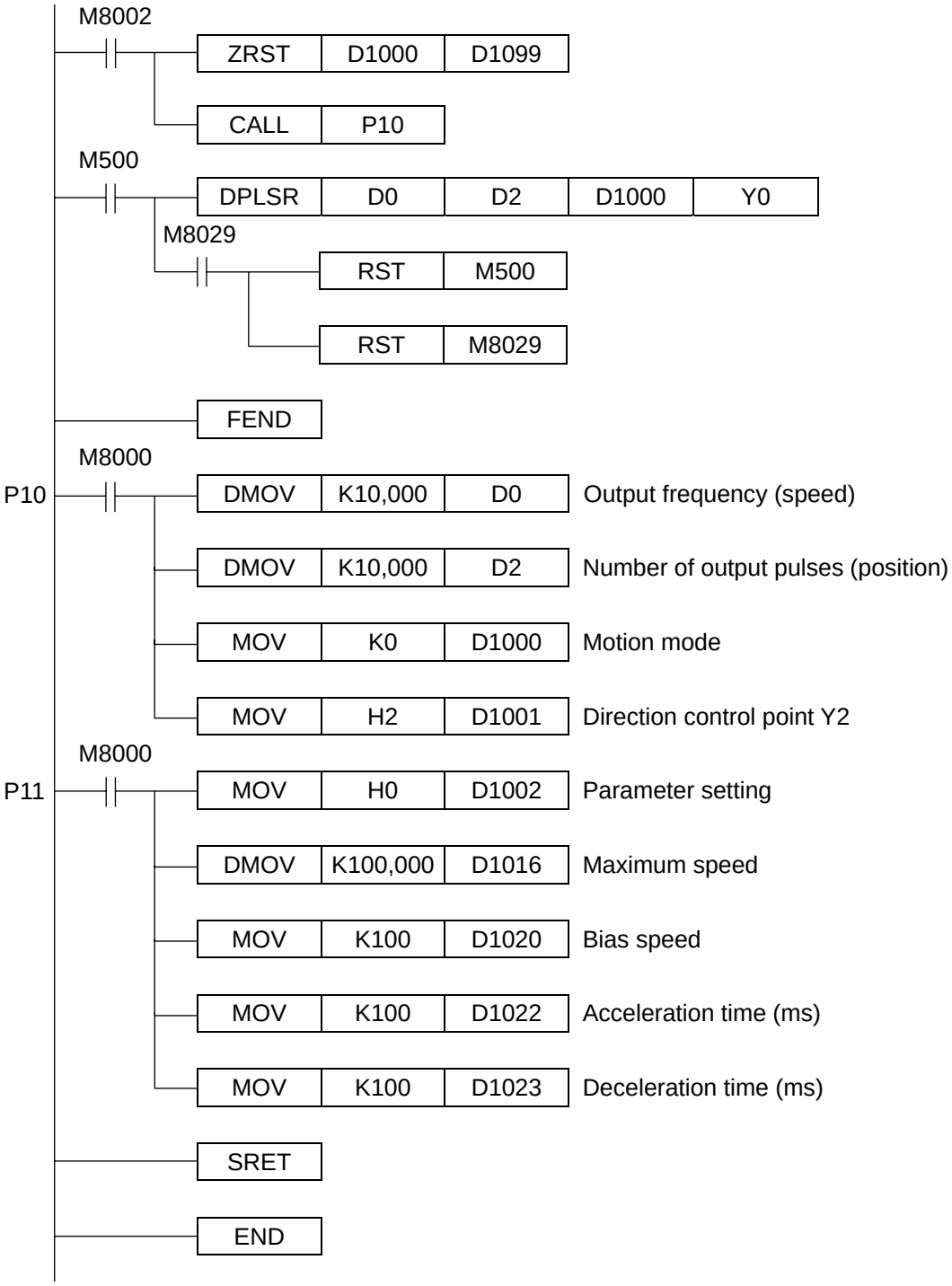
[S3.]+25 : zero-point signal setting value. If there is not zero-point signal (for stepping motor) when it turns to zero-point, then user would set number of search zero-point as “0”.
[S3.]+26 : zero-point signal count value (for monitor)
[S3.]+27 : system reserved
[S3.]+28 : electronic gear(numerator)
[S3.]+29 : electronic gear(denominator)
[S3.]+30 : system reserved
[S3.]+32 : system reserved

[S3.]+41, [S3.]+40 : PLSR-G00 mark sensor real time address buffer.
[S3.]+41, [S3.]+40 : PLSV number of output pulses. If value is 0, it is as without target operation.

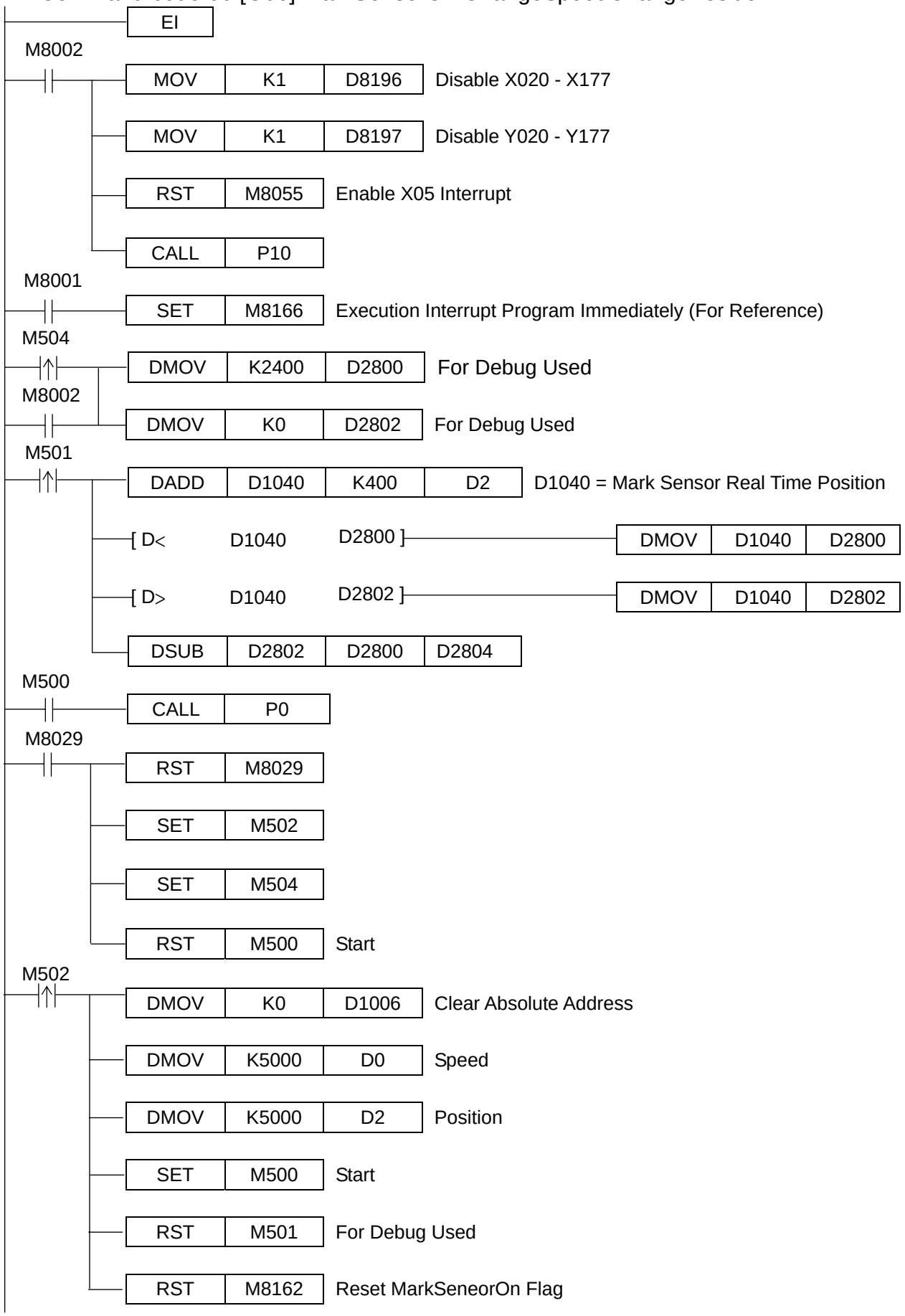
♦ MultiAxis moving : drive on the pseudo axis first, set the other axis to G06 ratio follow mode and assign to signal of pseudo axis pulse.

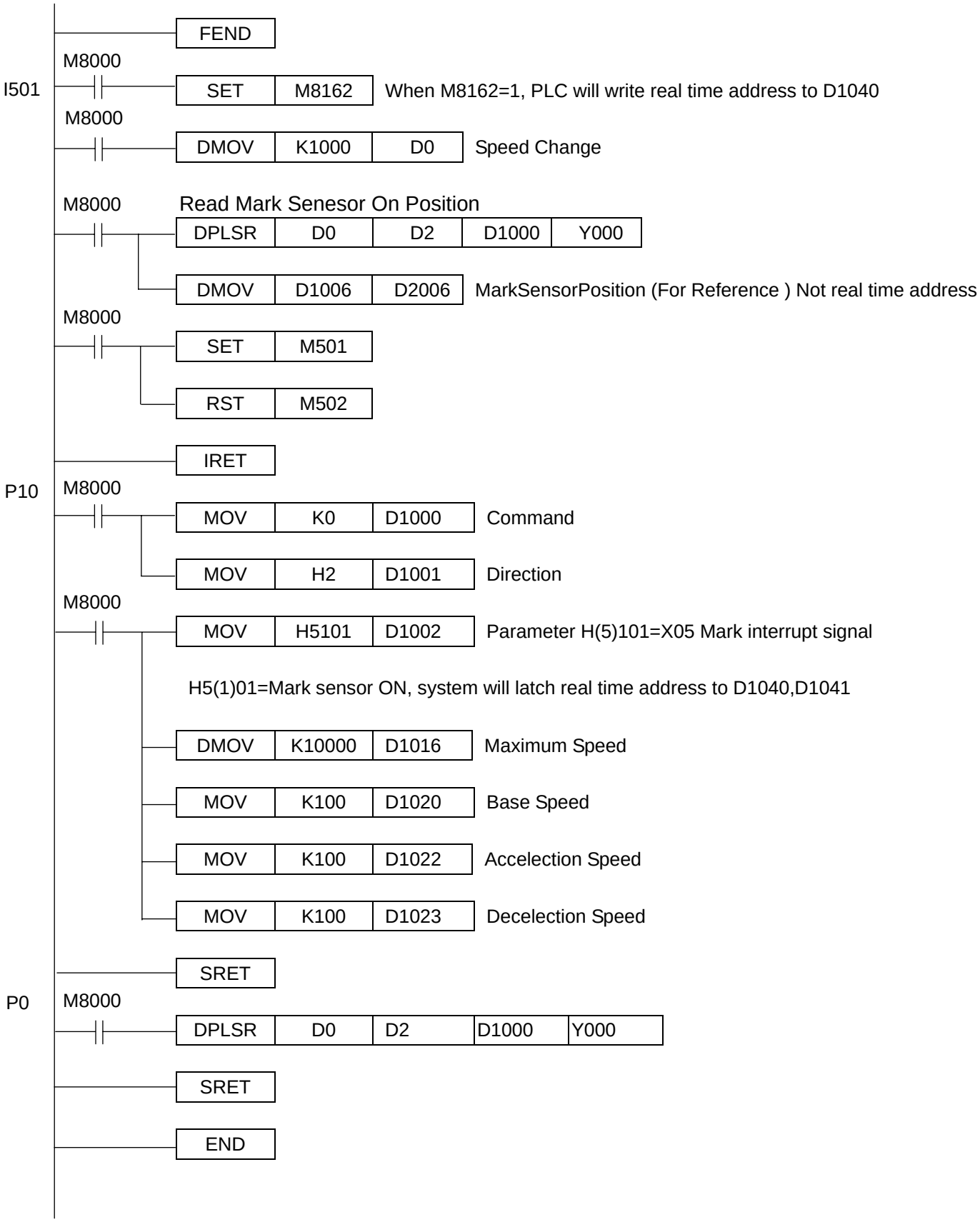
♦ When this instruction is used, increment distance or absolute address has to be converted to pulses, then stored to [S2.].
♦ When pulse output, X10 OFF, pulse is stopped outputting according to setting status of stop flag [S3]+2,b1.
♦ The pulse duty cycle is 50% ON, 50% OFF
♦ During G06 with slope with target instruction is under operation, it is ineffective to change content of [S2.]
♦ This instruction for Y00 or Y01 only can be used once (total twice), and has to select transistor output type.
♦ It is fixed to 32 bits operation. If user assigns 16 bits operation mode, then error 6509 will be occurred.
♦ There is only one kind of pulse output type in this instruction (Negative Logic Type, Pulse & Sign) can be controlled step or servo motor.

- Command value 00 [G00] Single position

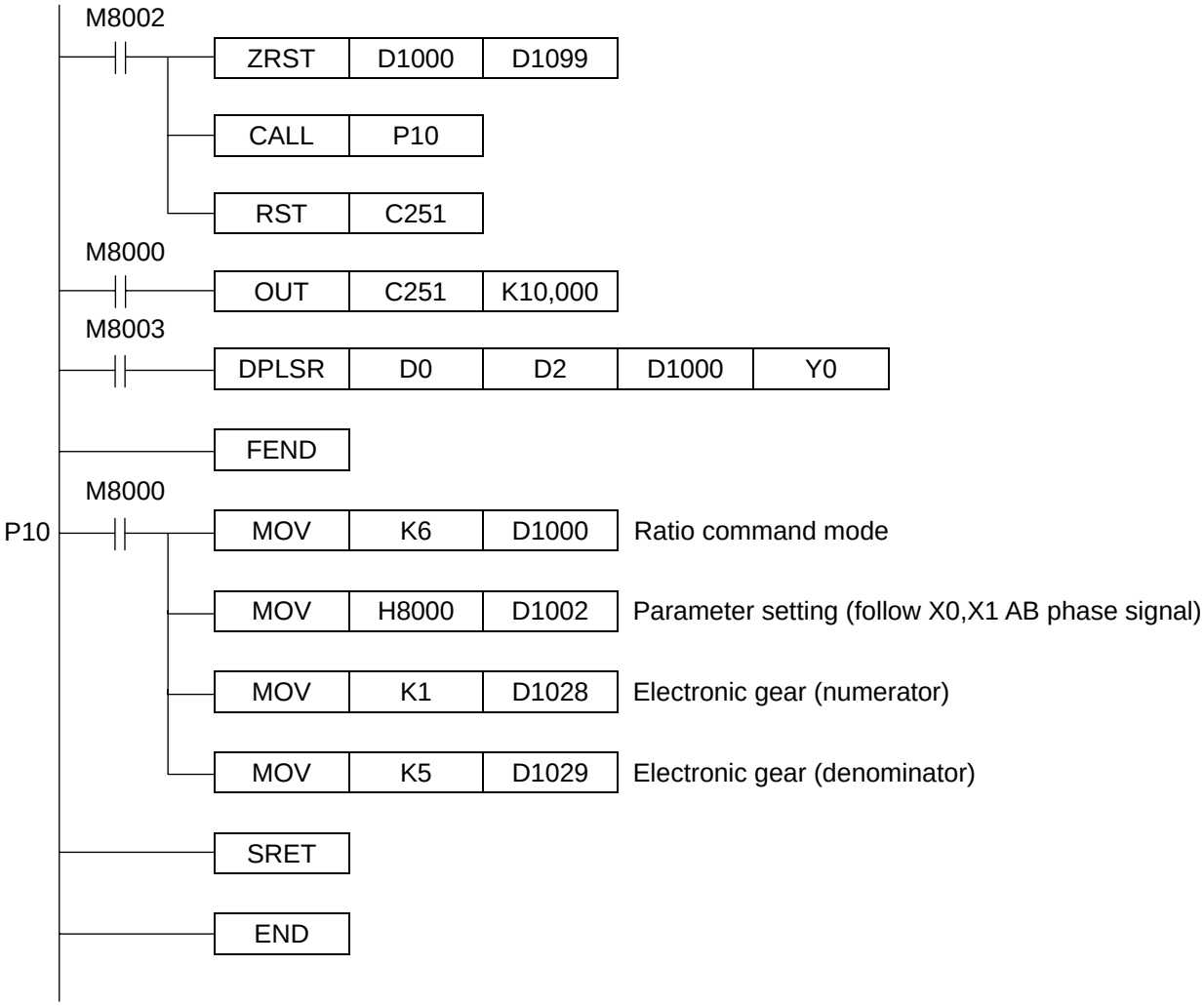


- Command code 00 [G00] MarkSensorOn ChangeSpeedChangePosition

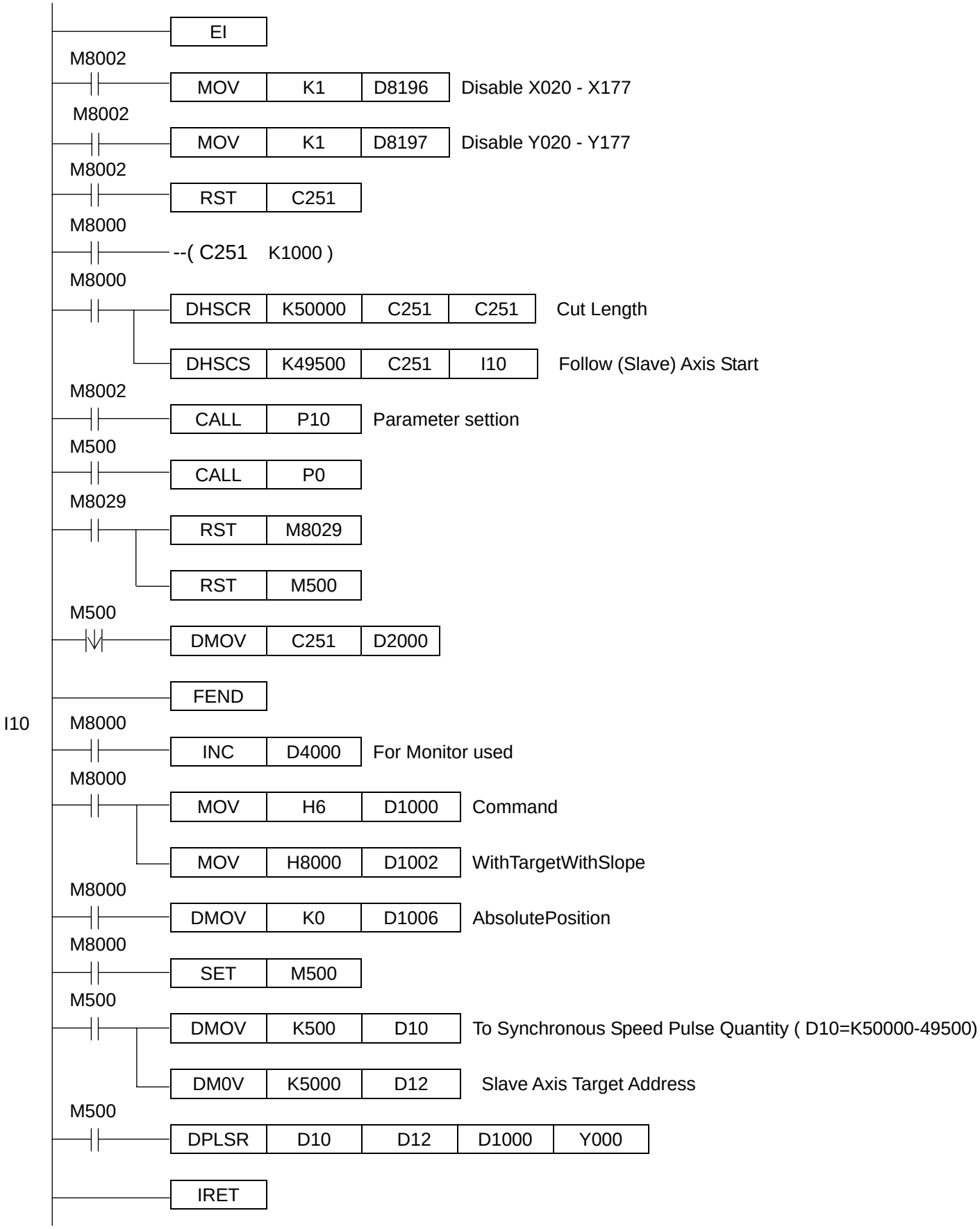


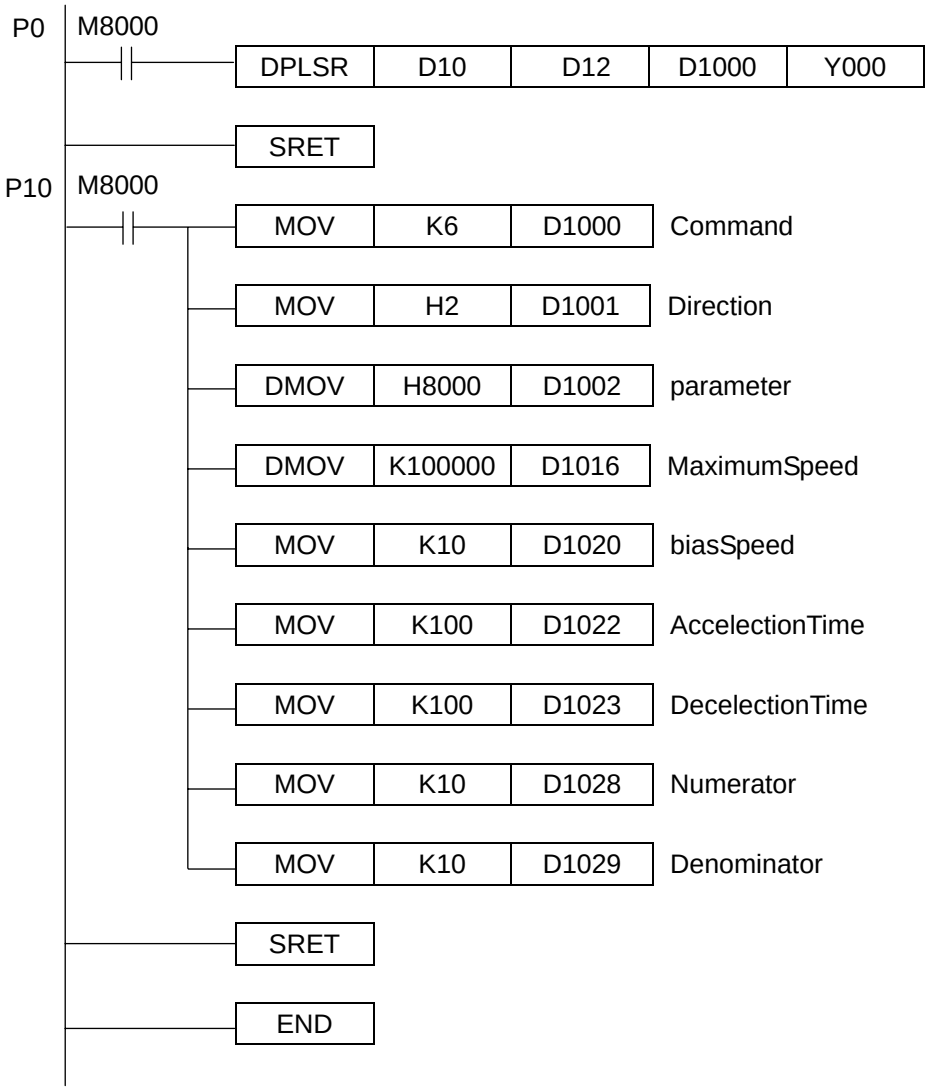

FEND
M8000
I501
SET M8162
When M8162=1, PLC will write real time address to D1040
M8000
DMOV K1000 D0
Speed Change
M8000
Read Mark Senesor On Position
DPLSR D0 D2 D1000 Y000
DMOV D1006 D2006
MarkSensorPosition (For Reference) Not real time address
M8000
SET M501
RST M502
IRET
P10
M8000
MOV K0 D1000
Command
MOV H2 D1001
Direction
M8000
MOV H5101 D1002
Parameter H(5)101=X05 Mark interrupt signal
H5(1)01=Mark sensor ON, system will latch real time address to D1040,D1041
DMOV K10000 D1016
Maximum Speed
MOV K100 D1020
Base Speed
MOV K100 D1022
Accelection Speed
MOV K100 D1023
Decelection Speed
SRET
P0
M8000
DPLSR D0 D2 D1000 Y000
SRET
END

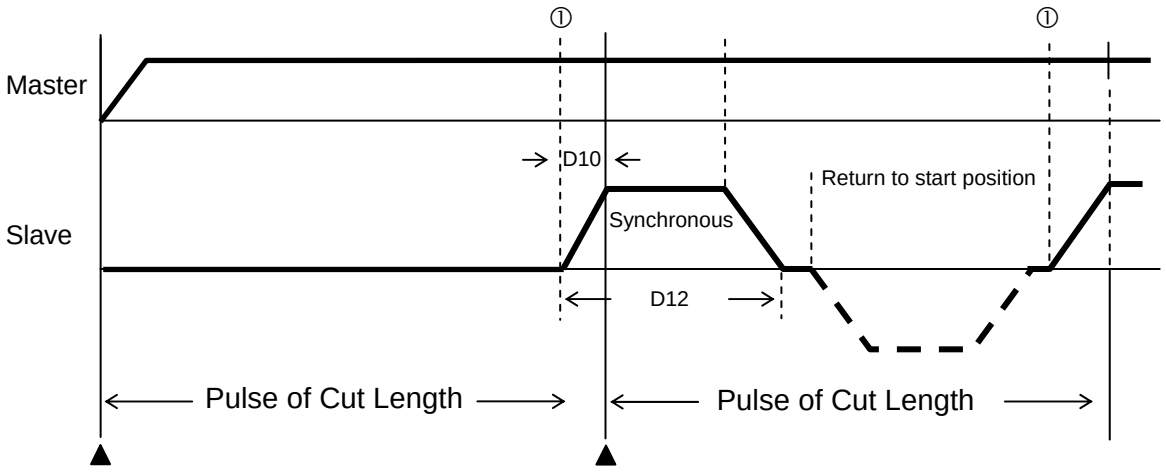
- Command 06 [G06] Ratio command (direction of Y0 axis is fixed as Y2; direction of Y1 is fixed as Y3)



- Command 06 [G06] Fly Saw Control

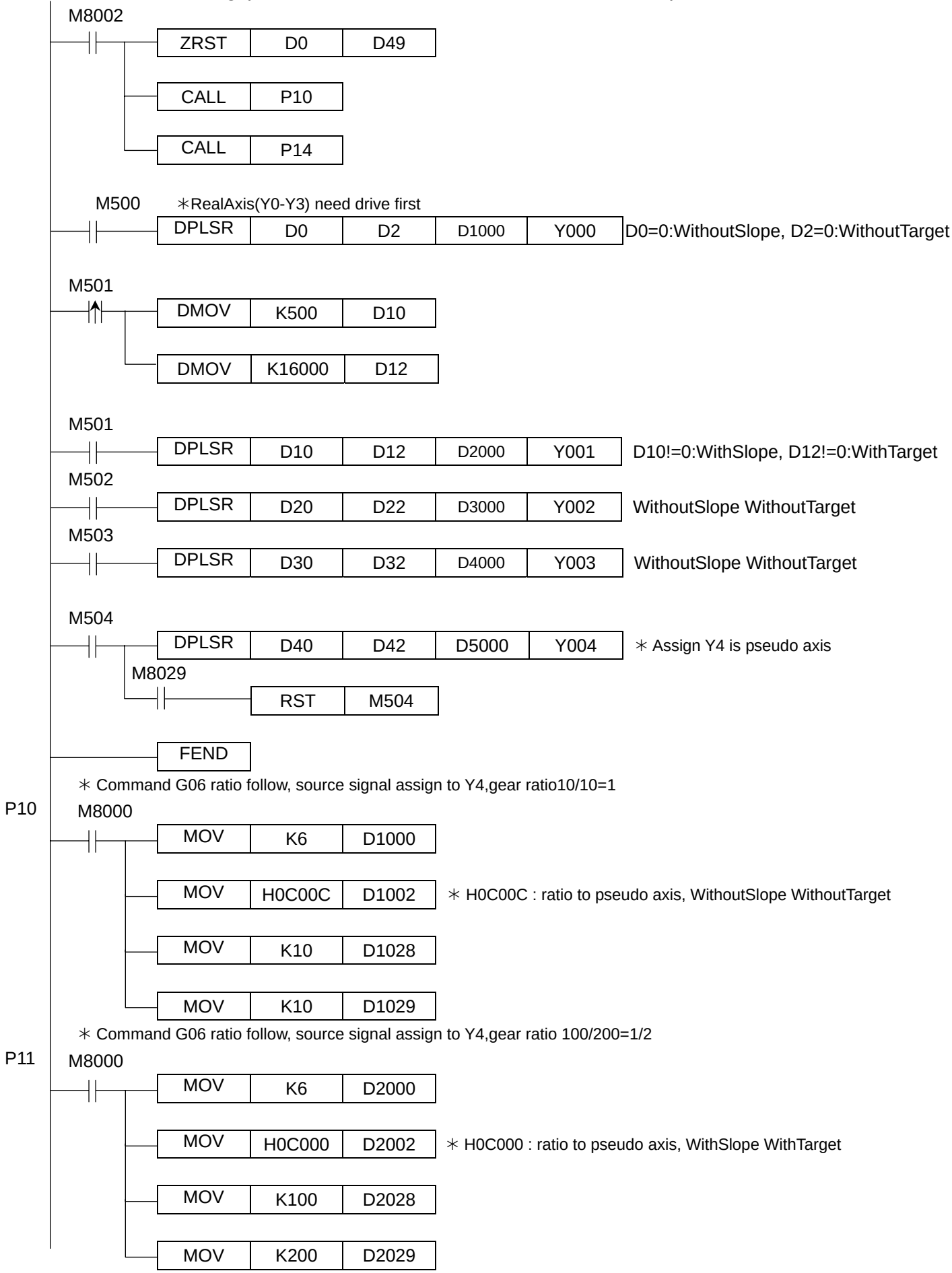


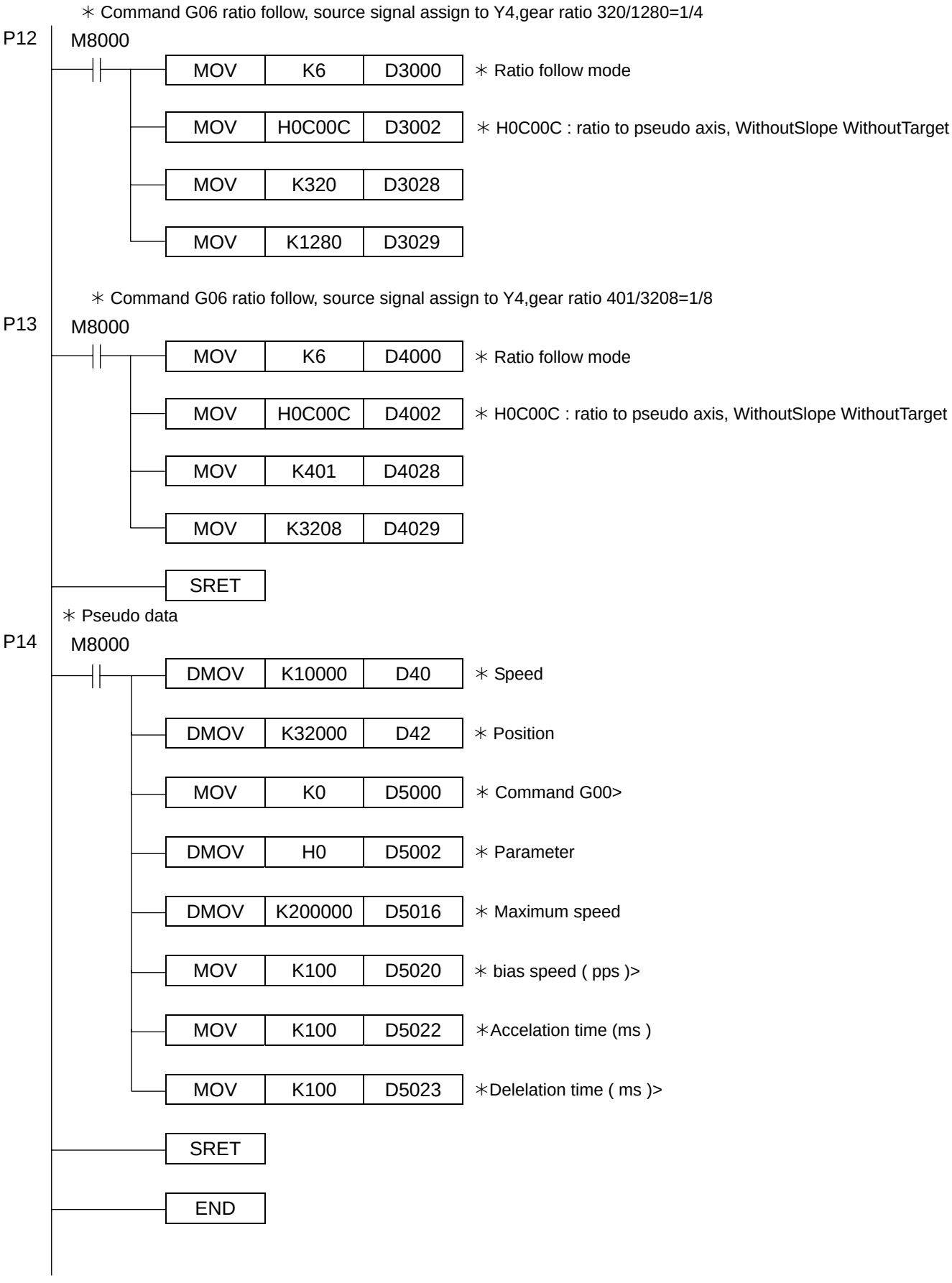




① Starting slave axis (When C251=Pulse of Cut Length - To Synchronous Speed Pulse D10)

- MultiAxis Moving (Virtual axis bit2,D5002=1 is no target position control), Real axis need drive first

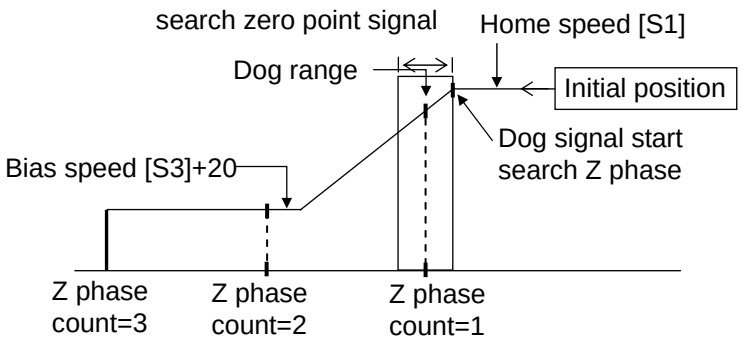



∗ Command G06 ratio follow, source signal assign to Y4,gear ratio 320/1280=1/4
P12
M8000
MOV K6 D3000 ∗ Ratio follow mode
MOV H0C00C D3002 ∗ H0C00C : ratio to pseudo axis, WithoutSlope WithoutTarget
MOV K320 D3028
MOV K1280 D3029
∗ Command G06 ratio follow, source signal assign to Y4,gear ratio 401/3208=1/8
P13
M8000
MOV K6 D4000 ∗ Ratio follow mode
MOV H0C00C D4002 ∗ H0C00C : ratio to pseudo axis, WithoutSlope WithoutTarget
MOV K401 D4028
MOV K3208 D4029
SRET
∗ Pseudo data
P14
M8000
DMOV K10000 D40 ∗ Speed
DMOV K32000 D42 ∗ Position
MOV K0 D5000 ∗ Command G00>
DMOV H0 D5002 ∗ Parameter
DMOV K200000 D5016 ∗ Maximum speed
MOV K100 D5020 ∗ bias speed (pps)>
MOV K100 D5022 ∗Accelation time (ms)
MOV K100 D5023 ∗Delelation time (ms)>
SRET
END

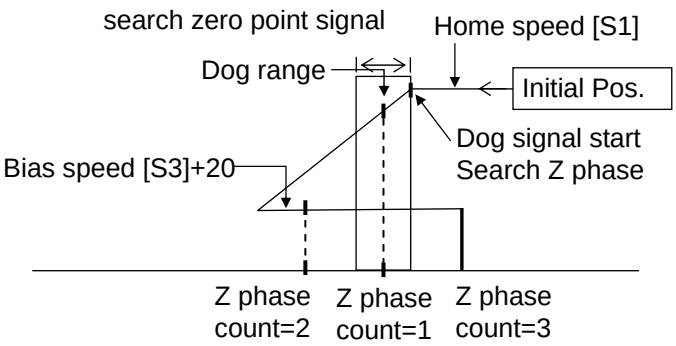
- Command value 28 [G28] Zero return (number of search for Z phase is not as 0)

```
M8002
 ||  ZRST  D1000  D1099
 ||  CALL  P10
M500
 |↑|  RST   M510   Clear zero return complete flag
M500
 ||  DPLSR  D0  D2  D1000  Y0
     M8029
      ||  RST   M500
      ||  SET   M510   Set zero return complete flag. This uses M510 (user can appoint M register)
      ||  RST   M8029
 ||  FEND
M8000
P10 ||  DMOV  K10,000  D0   Output frequency (speed)
      DMOV  K10,000  D2   Number of output pulses (position)
      MOV   K28      D1000  Zero return mode
      MOV   H0002    D1001  Direction control point Y2
      MOV   H0043    D1024  DOG signal X3 rising edge effective. Zero-point signal X4
                            rising edge effective. Initial operation direction as reverse
      MOV   K3       D1025  Set number of search Z phase. Speed of search Z phase is as
                            same as bias speed
P11 ||  [ ]   As same as example of DPLSR signal position
 ||  SRET
 ||  END
```

< MODE0 > D1024=H0043 Same direction search



< MODE1> D1024=H1043 different direction



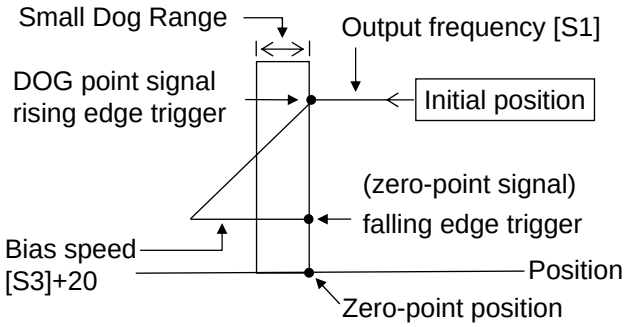
∗ Command value 28 [G28] Zero return

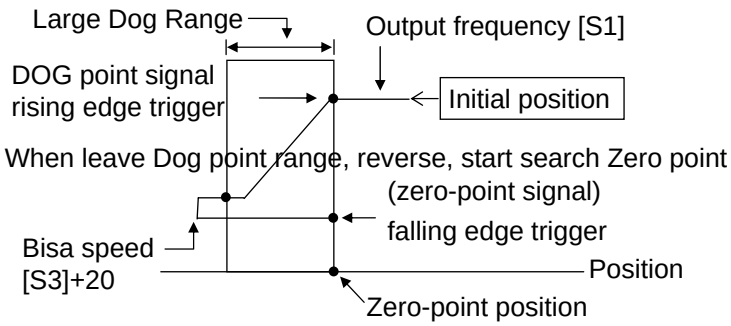
(number of search for Z phase is 0. DOG point signal and Zero point signal have to be set as the same point)

<< MODE0 >> First confirm DOG point and then decrement speed to Bias speed and need leave DOG effective range, reverse rotation and start searching ZERO point signal

D1024 = H0133 (DOG point signal X3 rising edge effective，Zero-point signal X3 falling edge effective，Initial operation direction as reserve direction)

D1025 = K0 (number of Z phase = 0)

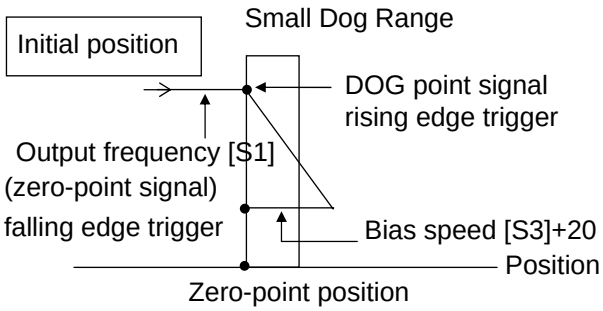


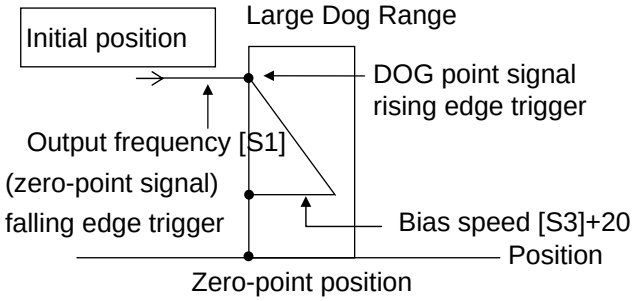


<< MODE1 >> First confirm DOG point and then decrement speed to Bias speed and don’t need leave DOG effective range, reverse rotation and start searching ZERO point signal

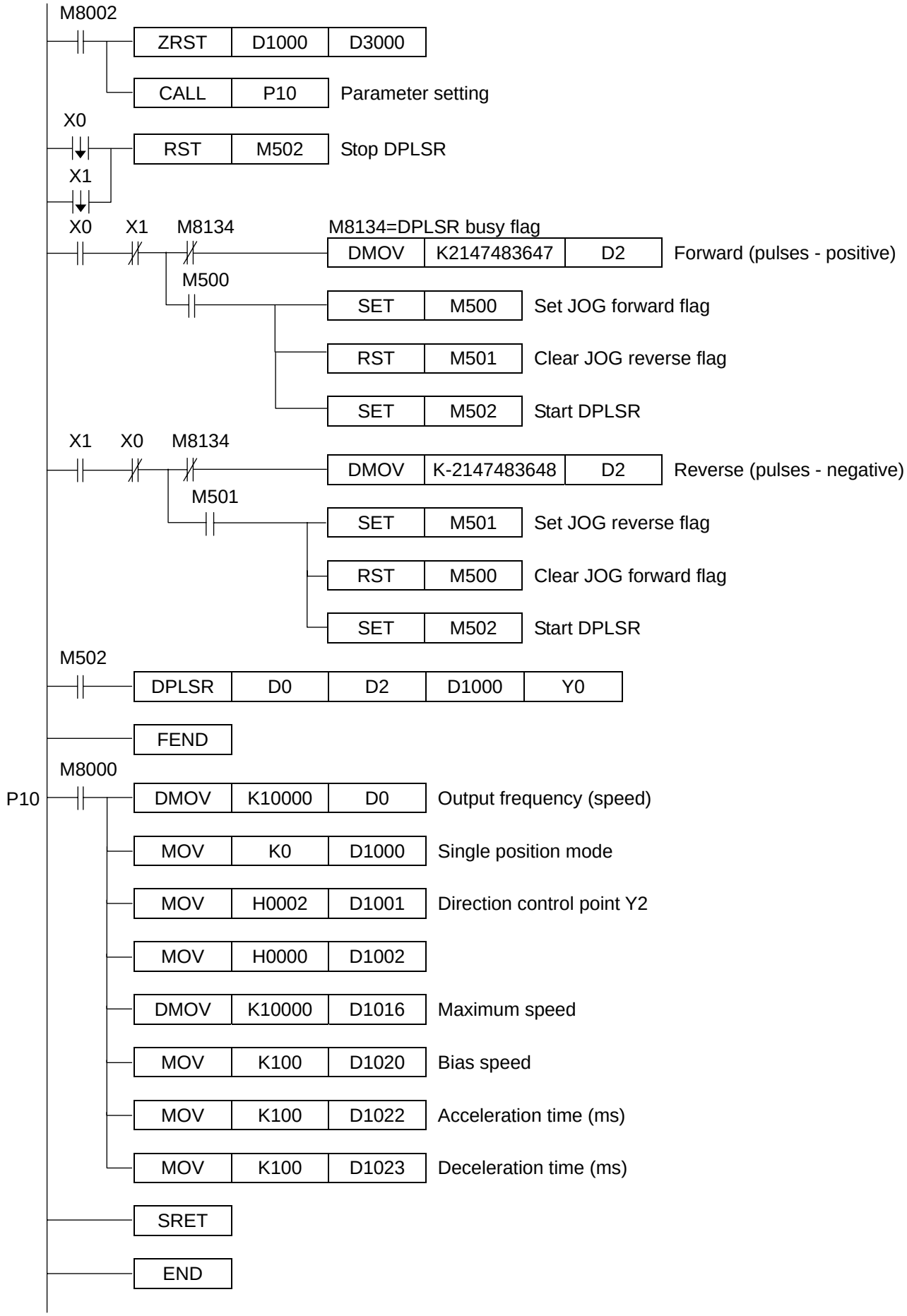
D1024 = H0133 (DOG point signal X3 rising edge effective，Zero-point signal X3 falling edge effective，Initial operation direction as forward direction)

D1025 = K0 (number of Z phase = 0)

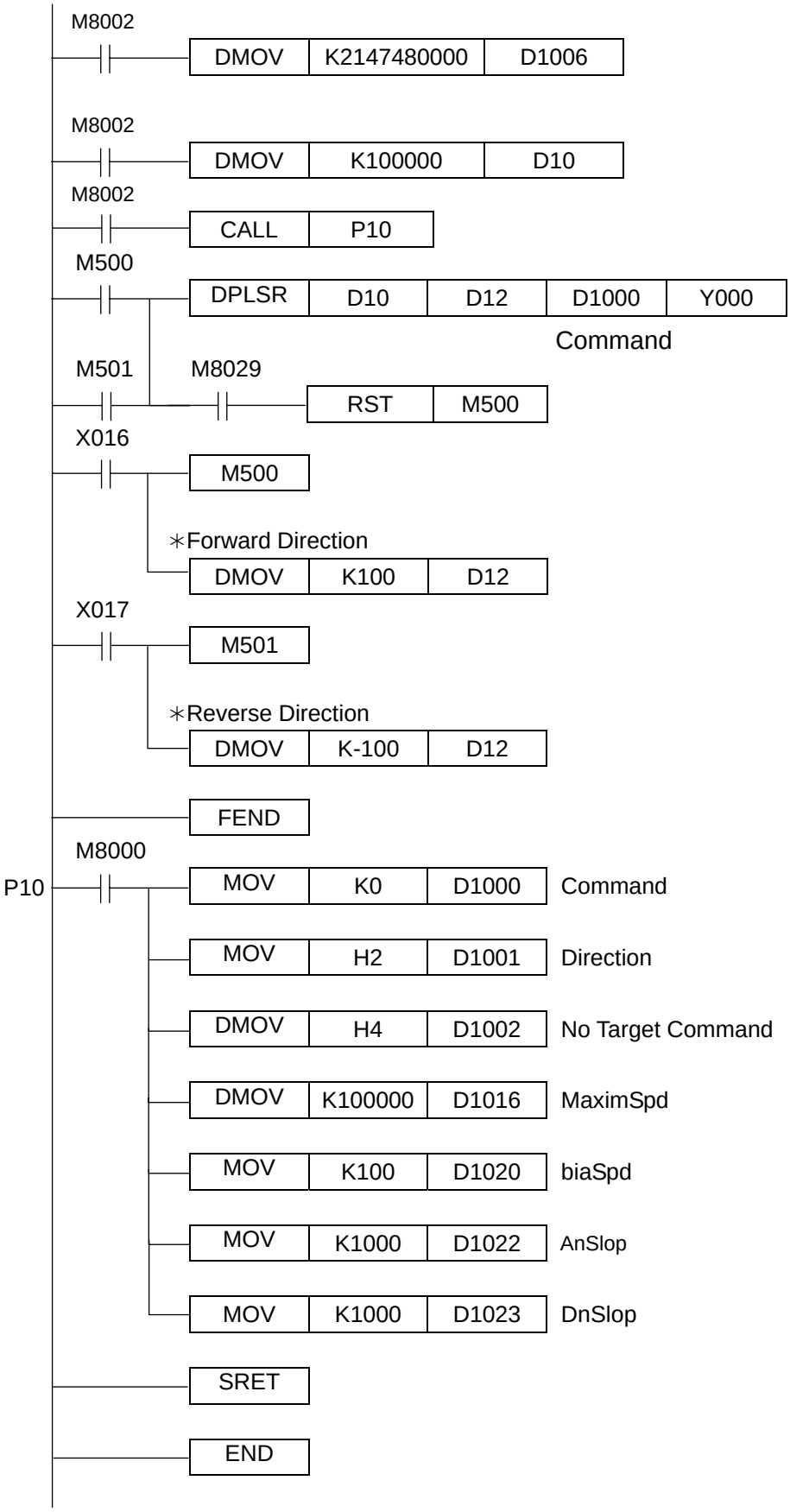




※ Sample program of DPLSR : JOG +/-



- Sample program of DPLSR : JOG +/- (No Rollover problem)



Initial State

FNC(60)			16 bits: IST - 7 steps
	IST		

Reserved

Data Search

FNC(61)			16 bits: SER(P)- -9 steps
D	SER	P	32 bits: (D)SER(P) - 17 steps

Reserved

Absolute Drum Sequence

FNC(62)			16 bits: ABSD - 9 steps
D	ABSD		32 bits: (D)ABSD -17 steps

		J1n	J2n--

Operands: [S1.]

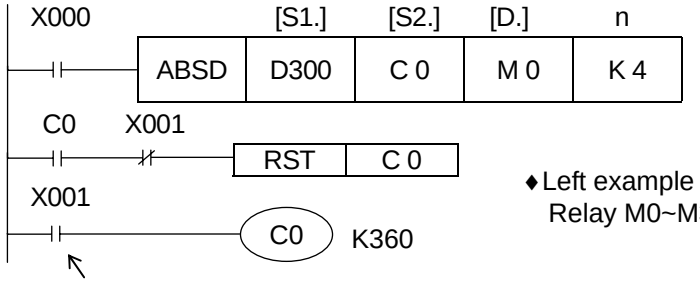
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

n ≤ 64 [S2.]

Operands:

X	Y	M	S

[D.]



This instruction is used to bring a varied output type to counter. It can detect the angle of the circle control action.

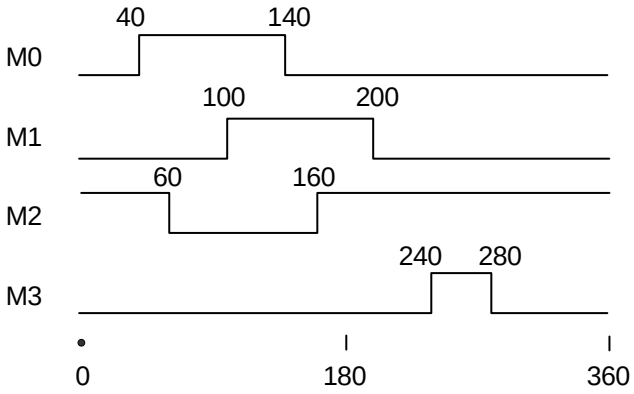
♦Left example is used to control ON/OFF status of Auxiliary Relay M0~M3 when rotation table rotate within a circle.

♦ Using MOVE instruction to write following values into D300~D307

ON setting value	OFF setting value	Output point
D300= 40	D301= 140	M0
D302= 100	D303= 200	M1
D304= 160	D305= 60	M2
D306= 240	D307= 280	M3

Put Turn ON value to even number of D device, and put Turn OFF value to Odd number of D device

♦When X0 ON, change of M0~M3 is mentioned as follows. Turn ON and Turn OFF value can re-change to write into D300~D307



♦Output point number is decided by setting value of [D.]

♦When X0 become OFF, output is not changed.

♦ABSD instruction just can be used once in one program.

♦When assign High Speed Counter in [S.], then also can use (D)ABSD instruction.
For current value of counter at this time, the output status will delay because of scan-time, recommend to use Table high-speed compare mode of HSZ instruction.

Incremental Drum Sequence

FNC(63)	16 bits: INCD - 9 steps
INCD	

		J1n	J2n--

Operands: [S1.]

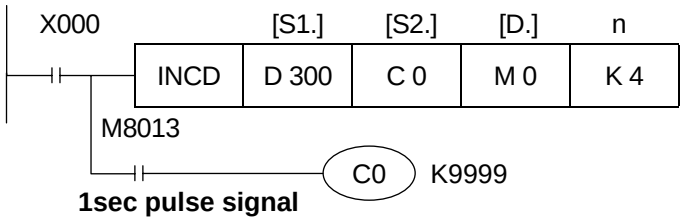
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

n ≤ 64 [S2.]

Operands:

X	Y	M	S

[D.]



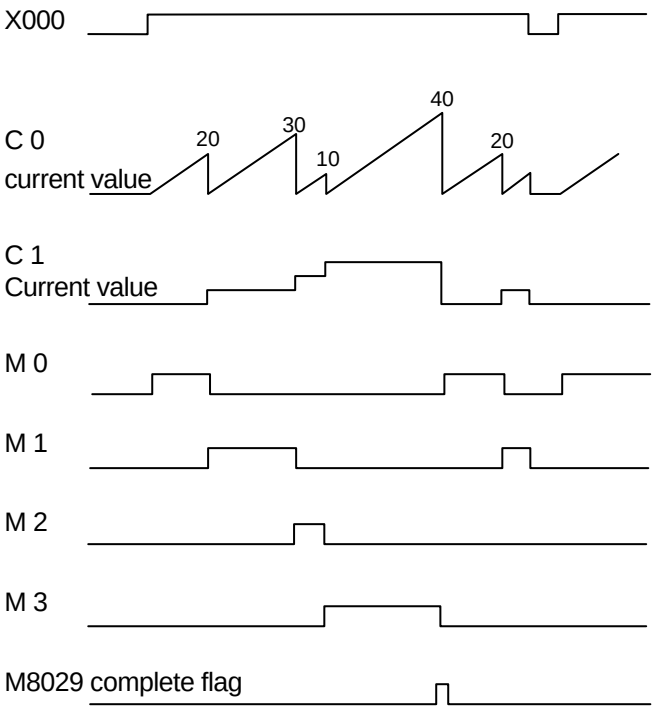
This instruction is used to produce a varied output when using a couple of counter.

Following is the control range of 4 points (M0~M3)

♦Use MOVE instruction to write following value into [S1.] in advance.

D300 = 20 D302 = 10

D301 = 30 D303 = 40



♦When counting value of C0 reach to setting value of D300~D303, C0 reset automatically in turn

♦C1 count occurred number of C0 reset.

♦M0~M3 act in turn according to counting value of C1.

♦After complete last operation of setting number by ”n”, flag M8029 become ON. Above mentioned action will be always repeated.

♦ When X0 OFF, C0 and C1 is cleared, M0~M3 become OFF, then operate again when X0 become ON.

♦INCD instruction only can be used once in one program.

Teaching Timer

FNC(64)				16 bits: TTMR - 5 steps				
	TTMR							

Reserved

Special Timer

FNC(65)				16 bits: STMR - 7 steps				
	STMR							

Reserved

Alternate Output

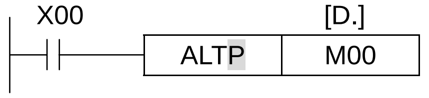
FNC(66)			16 bits: ALT(P) - - - - - - - - - - - - - - - - - - -- - - - - - - - 3 steps
	ALT	P	

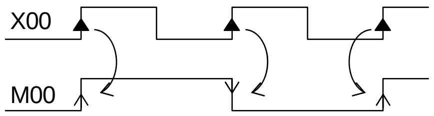
		J1n	J2n--

Operands: [D.]

X	Y	M	S

Flag:





Ramp

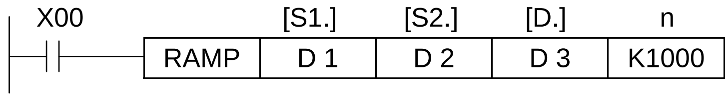
FNC(67)			16 bits: RAMP - - - - - - - - - - - - - - - - - - -- - - - - - - - 9 steps
	RAMP		

		J1n	J2n--

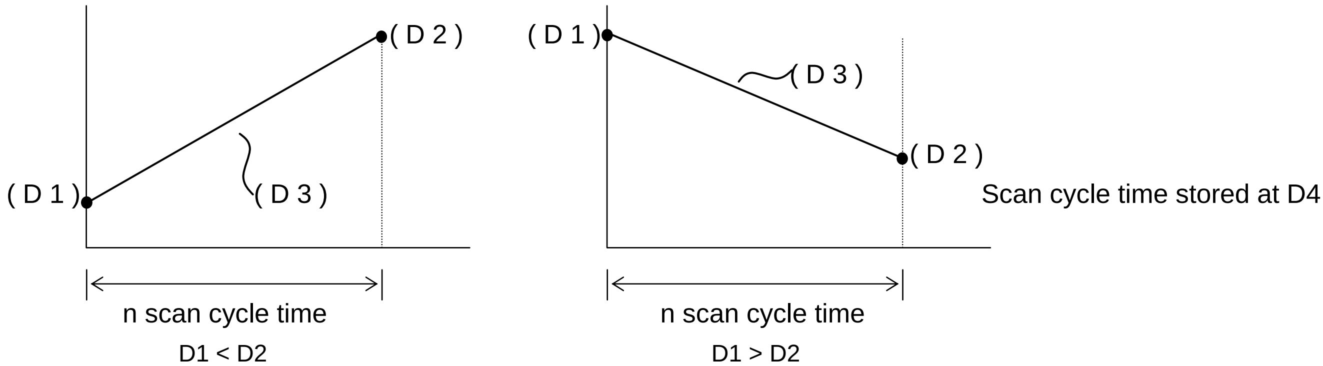
Operands: [S1.][S2.][D.] : D

n : K,H   n = 1 to 32,767

Flag: M8029



- ♦ When X0 ON, content of [S1.] and [S2.] are stored into [D.]. Content of [D.] is increased by "1" each scan cycle. n: the number of scan cycle.



♦ After M8029 is driven, write once scan-time value (longer than actual scan-time) into M8039, and then PLC will enter to fixed scan mode.

For example, n = K1000 in above example. If scan cycle is set to 20msec, then value in D3 will be changed from setting value of D1 to setting value of D2 within 20sec.

- ♦ If X0 become OFF when acting, then act of RAMP signal will stop in midway. If X0 ON again, then D4 will be cleared and D3 will restart by setting value of D1.
- ♦ After end of execution, flag M8029 act, and then value of D3 will return to value of D1.
- ♦ Control of start / end point can be executed by RAMP instruction and analog output.
- ♦ Enter into RUN status when X0 ON.

Rotary Control

FNC(68)				16 bits: ROTC -- - - - - - - - 9 steps
	ROTC			

Reserved

Sort

FNC(69)				16 bits: SORT -- - - - - - - 11 steps
	SORT			

Reserved

Tenkey Input

FNC(70)			16 bits: TKY - 7 Steps
D	TKY		32 bits: (D)TKY - 13 Steps

		J1n	J2n--

Operands: [D1.]

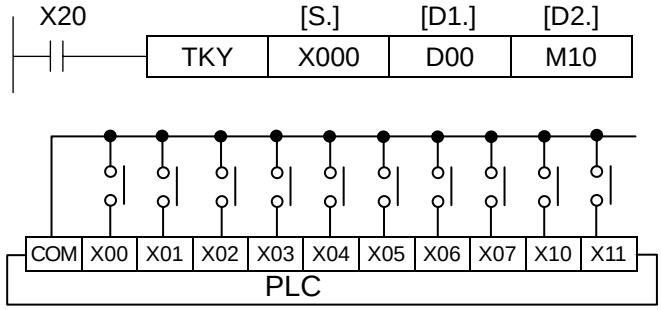
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands: [S.]

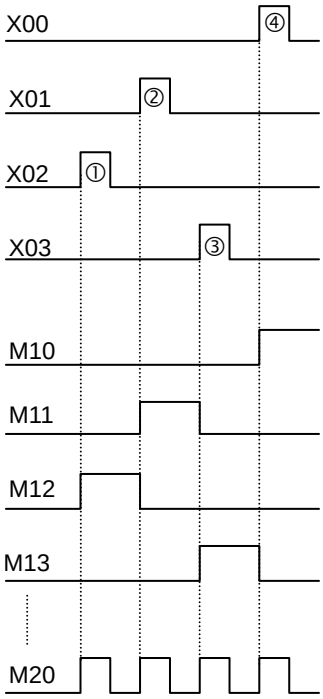
X	Y	M	S

[D2.]

Flag:



- ♦ This instruction can read 10 consecutive devices and will store an entered numeric string in [D1].
- ♦ In 16 bits operation, [D1] can store numbers from 0000 to 9999 (max. 4 digits). In 32 bits operation, [D1] value from 00000000 to 99999999 (max. 8 digits). In both cases, if the number exceeds the allowable ranges, the highest digit will overflow, and ignored it.
- ♦ When X20 OFF, all of the [D2.] devices are reset, but contents of [D1.] keep intact.



- ♦ This example will stored 2130 into D00.
- ♦ When two or more keys were pressed, only the First Key is effective.
- ♦ This example, the status of M10~M19 to be changed by the status of X00~X11.
- ♦ This example M20 is key release flag.

- ♦ This instruction may only be used once.

Hexadecimal Key

FNC(71)			16 bits: HKY - 9 Steps
D	HKY		32 bits: (D)HKY - 17 Steps

		J1n	J2n--

Operands: [D2.]

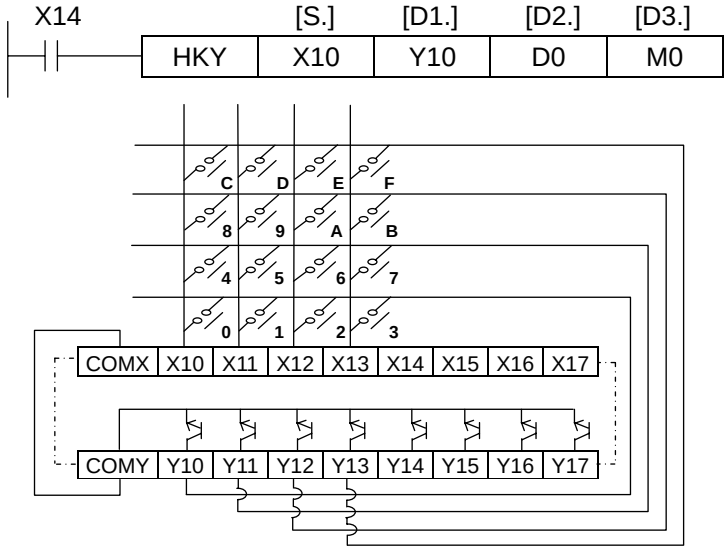
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands: [D3.]

X	Y	M	S

[S.] [D1.]

Flag:



- When the numeric key (0 ~ 9) be pressed, then causes bit device [D3.]+7 turn ON for the duration of key press.
- When the function key (A ~ F) be pressed, then causes bit device [D3.]+6 turn ON for the duration of key press.
- When the function key has been pressed, then will set bit devices [D3.]+0 to [D3.]+5 to ON, and remain ON until the next function key has been activated.

F	E	D	C	B	A
↓					↓
M5	M4	M3	M2	M1	M0

- In 16 bits operation, [D2.] can store numbers from 0000 to 9999 (max. 4 digits). In 32 bits operation, [D2.] value from 00000000 to 99999999 (max. 8 digits). In both cases, if the number exceeds the allowable ranges, the highest digit will overflow, and ignored it.
- When two or more keys were pressed, only the first key is effective. When X14 OFF, all [D3.] devices are reset, but contents of [D2.] keep intact.
- This instruction requires 8 scans cycle time to read the key input. After 8 scans, complete flag M8029 to be turned ON. This flag is automatically reset when this instruction execute.
- This may only be used once, and only the transistor module can be selected.

Digital Switch

FNC(72)			16 bits: DSW - 9 Steps
	DSW		

		J1n	J2n--

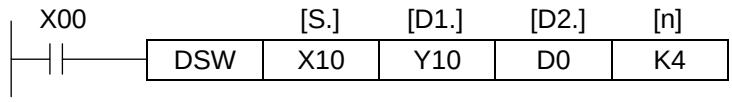
Operands: ←[n]→ =1~8 ←——— [D2.] ———→

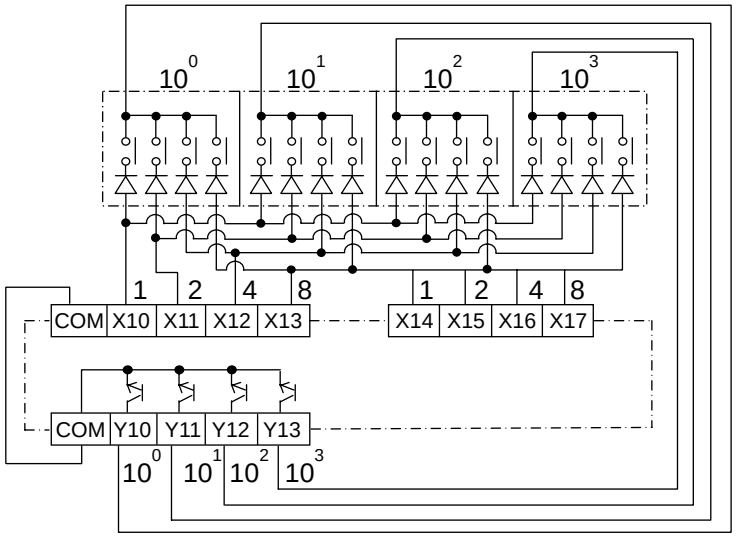
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands: ≤[S.]≥ ≤[D1.]≥

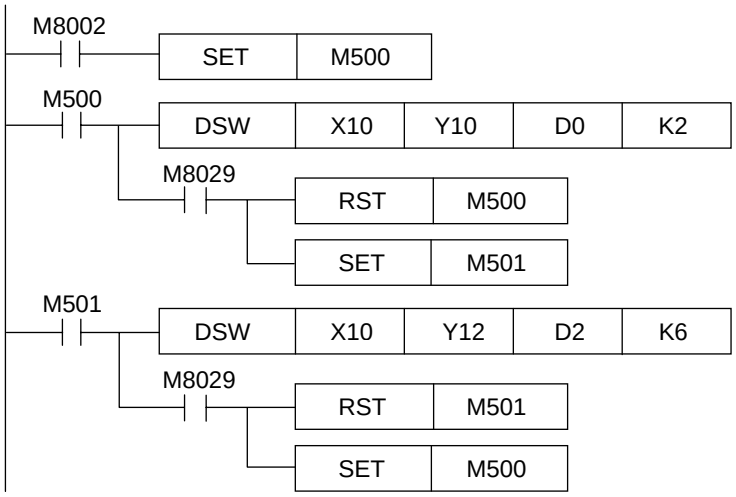
X	Y	M	S

Flag:M8029





- This instruction used n (1~8) output points and 4 input points to read in n (1~8) thumbwheel switch. If the read data is larger than 32 bits (n≥5), then [D2.] automatically occupy the next word device.
- This example the BCD 4 digit thumbwheel switch (1,2,4,8) is connected to X10~X13 or X14~X17, the source [S.] needs to be used X10,X14,X20,X24….as the head address.
- Once DSW execute, then the flag M8029 reset to “0”. When execution is completed, M8029 set to ”1”.
- Each pin (1,2,4,8) of the thumbwheel switch needs to be connected a diode (0.1A/50V)
- This may only be used once, and only the transistor module can be selected. If use M8029, then can control two or more DSW .



Seven Segment Decoder

FNC(73)			16 bits: SEGD(P) - 5 steps
	SEGD	P	

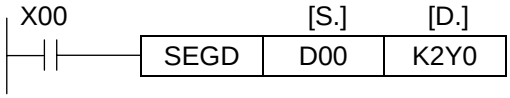
		J1n	J2n--

Operands:

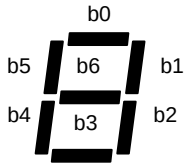
[S.]								
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.]

Flag:



- A single hexadecimal digit (0~9, A~F) occupying the lower 4 bits of the source device [S.] is decoded to a data format used to drive a seven segment display.
- The decoded data is stored in the lower 8 bits of destination device [D.]. The upper 8 bits was unchanged.

(S.)		Seven segment display	(D.)								data
Hex	Bit		b7	b6	b5	b4	b3	b2	b1	b0	
0	0000		0	0	1	1	1	1	1	1	0
1	0001		0	0	0	0	0	1	1	0	1
2	0010		0	1	0	1	1	0	1	1	2
3	0011		0	1	0	0	1	1	1	1	3
4	0100		0	1	1	0	0	1	1	0	4
5	0101		0	1	1	0	1	1	0	1	5
6	0110		0	1	1	1	1	1	0	1	6
7	0111		0	0	1	0	0	1	1	1	7
8	1000		0	1	1	1	1	1	1	1	8
9	1001		0	1	1	0	1	1	1	1	9
A	1010		0	1	1	1	0	1	1	1	A
B	1011		0	1	1	1	1	1	0	0	b
C	1100		0	0	1	1	1	0	0	1	C
D	1101		0	1	0	1	1	1	1	0	d
E	1110		0	1	1	1	1	0	0	1	E
F	1111		0		1	1	0	0	0	1	F

Seven Segment With Latch

FNC(74)			16 bits: SEGL(P) - 5 steps
	SEGL	P	

Reserved

Arrow Switch

FNC(75)			16 bits: ARWS(P) - 9 steps
	ARWS		

Reserved

Ascii Code Conversion

FNC(76)			16 bits: ASC -11 steps
	ASC		

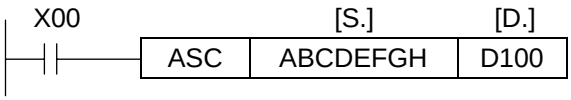
		J1n	J2n--

Operands: [S.]: 8 character or alphanumeric data.

Operands: ←— [D.] —→

	T	C	D

Flag:



♦ The source data string [S.] consists of up to 8 characters.

♦ The character ”A”~”H” is converted to ASCII codes, then stored into D100~D103.

When M8161 is OFF

M8161=OFF	Upper 8 bits	Lower 8 bits
D100	“B”	“A”
D101	“D”	“C”
D102	“F”	“E”
D103	“H”	“G”

When M8161 is ON

	Upper 8	Lower 8		Upper 8	Lower 8
D100	0	“A”	D104	0	“E”
D101	0	“B”	D105	0	“F”
D102	0	“C”	D106	0	“G”
D103	0	“D”	D107	0	“H”

Print

FNC(77)			16 bits: PR - 5 steps
	PR		

Operands: ←— [S.] —→

	T	C	D

Operands: [D.]: Y

Reserved

FROM

FNC(78)			16 bits: FROM(P) - 9 steps
D	FROM	P	32 bits: (D)FROM(P) - 17 steps

		J1n	**J2n--**

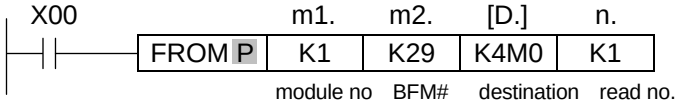
Operands: ← [D.] →

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands: ← → m1 = 0 ~ 7 no. of special module

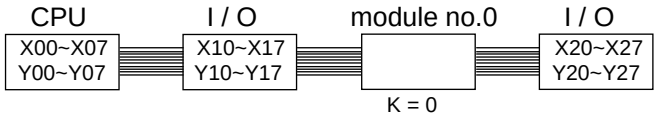
m2.= 0 ~ 31 no. of buffer memory (BFM)

n.= 1 ~ 31 no. of read (when D, n=1~15)



♦ When X00 ON, the buffer memory of special module BFM#29 to be read and stored into M00~M15.

<< Special Device Module Number m1>>



♦ The BFM is the memory address of special module.

♦ The number of special module is address to NO.0~NO.7 and beginning with the one closest to the CPU unit.

♦ The special module can up to 8 maximum, and no occupy I/O points.

TO

FNC(79)			16 bits: TO(P) - 9 steps
D	TO	P	32 bits: (D)TO(P) - 17 steps

		J1n	**J2n--**

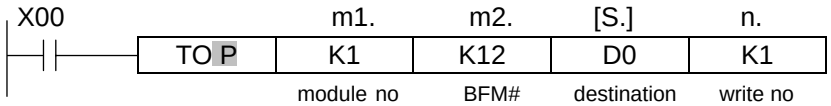
Operands: ← [S.] →

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands: ← → m1 = 0 ~ 7 no. of special module

m2.= 0 ~ 31 no. of buffer memory (BFM)

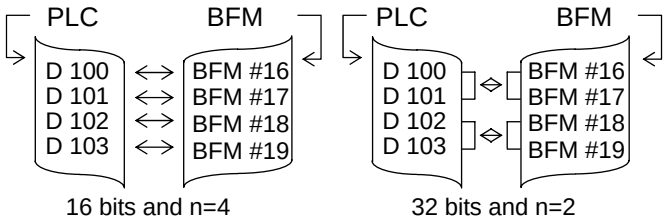
n.= 1 ~ 31 no. of write (when D, n=1~15)



♦ When X00 ON, the content of D0 to be write into the buffer memory BFM#12 of the special module NO.1

♦ If used pulse command can decrement cycle time.

<< Number of Read n >>



Communication

FNC(80)			16 bits: RS - 9 steps
	RS		

		J1n	J2n--

Operands: ←S,→

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z
←——→ m,n=1~128							← D.→	m,n

Flag:

<< Communication Format >> D8120

	Content	0	1
Bit0	Data length	7 bit	8 bit
Bit1	Parity	(00):none, (01):odd, (11):even	
Bit2			
Bit3	Stop Bit	1 bit	2 bit
Bit4	Baud rate (bps)	(0011):300, (0100):600	
Bit5		(0101):1200, (0110):2400	
Bit6		(0111):4800, (1000):9600	
Bit7		(1001):19200	
Bit8	Start 1	None	D8124
Bit9	End 1	None	D8125
Bit10	Reserved	-	-
Bit11	Reserved	-	-
Bit12	End 2	None	D8126
Bit13	RS Mode	User define	ModBus
Bit14	ModBus Mode	Ascii Mode	RTU Mode or Computer Link
Bit15	Protocol	Format 1	Format 4

♦ EXADP232/422/485 communication board connected to the 2nd communication port of EXPLC to execute transmitting and receiving data. The protocol is assigned by D8120

♦ The protocol and data frame are all defined by user, and can be selected different communication interface board, so EXPLC can be communicated with other kind of machines.

♦ When main unit start to operate, it will check if there is RS instruction by itself. If yes, then Computer link mode is ineffective. The Protocol will be changed to user define mode or Modbus mode.

♦ Computer link mode: Program of this mode can not be written RS instruction; i.e., all stations are slaver unit. It only set content of D8120 and D8121 (bit14 of D8120 have to be set 1), i.e., it can construct multi-station connection system.

Example:

```
M8002
 ||  [ MOV   HE086   D8120 ] ; Format 4; 9600; 7E1
 ||  [ MOV   K1      D8121 ] ; station no. (1~32)
```

♦ Modbus mode: Program of this mode have to use RS instruction to change protocol (bit13 of D8120 have to be set 1). Because there is RS instruction, then it can be master unit and can be slaver unit also. It uses M8122 and M8123 to control transmitting and receiving data.

Example:

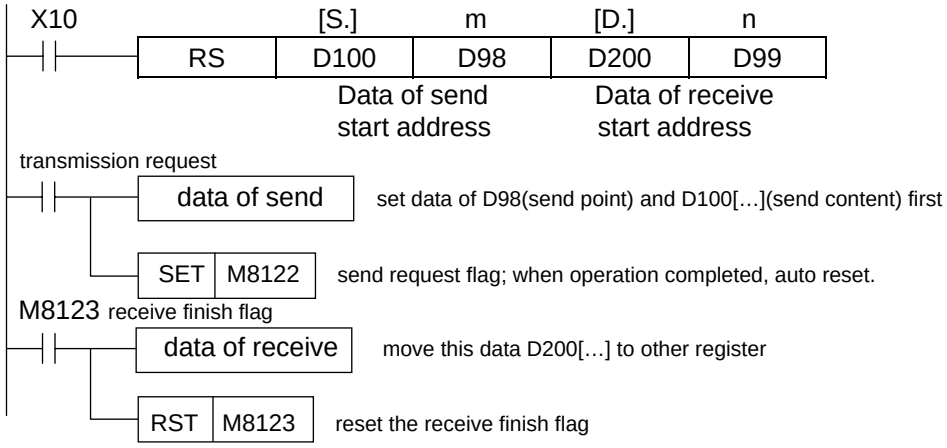
```
M8002
 ||  [ MOV   H3386   D8120 ] ; Modbus Ascii mode; 9600; 7E1
 ||  [ MOV   H00     D8121 ] ; Master station
 ||  [ RS    D100    D98   D200   D99 ]
```

♦ When RS executing, changing data of D8120 does not affect current operation.

♦ The using frequency of this instruction in program is not limited, but it only can use one execution command for one scan-time and it have to design more than one scan time of OFF time when changing.

♦ The communicate port of EXPLC can be as master unit or slaver unit. Therefore, once RS execute, then enable the function of communication and wait for trigger signal.

♦ If RS instruction is used, then PRUN instruction can't be used.



<< Request of transmission >> M8122

♦ When the transmit request flag M8122 to be driven in the waiting communicate status, then PLC will transmit from the head address of D100 for D98 number of bytes to slaver, and M8122 will auto reset after transmit completed.

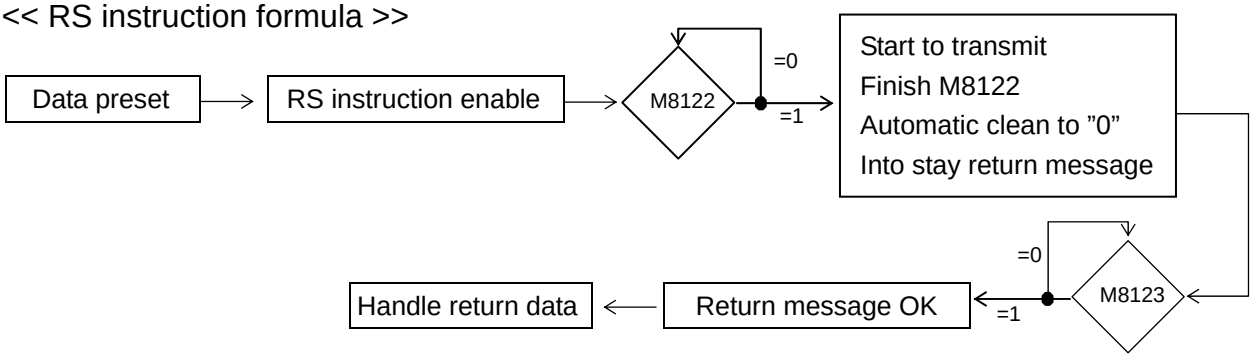
<< Receive Finish Flag >> M8123

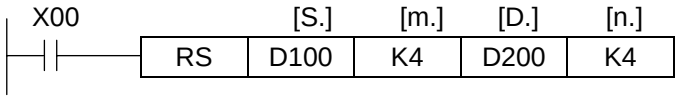
♦ When PLC finish to receive data, receive finish flag M8123 will set to “1”, user can use program to reset it.

<< Carrier Detect Flag >> M8124

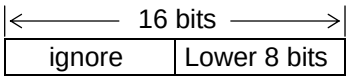
Reserved

<< RS instruction formula >>





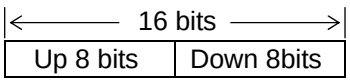
< 8 Bits Mode > M8161=ON is 8 bits operation



start	data of transmitting (m)				end
STX	D100 down	D101 down	D102 down	D103 down	ETX

start	data of receiving (n)				end
STX	D200 down	D201 down	D202 down	D203 down	ETX

< 16 Bits Mode > M8161=OFF is 16 bits operation



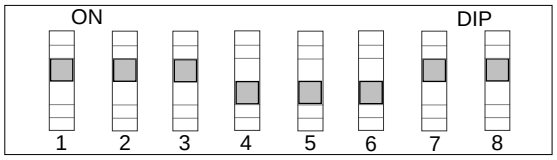
start	data of transmitting (m)				end
STX	D100 down	D100 up	D101 down	D101 up	ETX

start	data of receiving (n)				end
STX	D200 down	D200 up	D201 down	D201 up	ETX

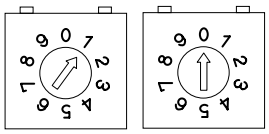
♦ If error occurrence was in the communication, then error flag M8063 to be set and error code in the D8063.

<< MODBUS RTU >> CRC error check mode

♦ switch of EXRM0808R/T



SW1



RSW1 RSW2

Ex: application note of Master and Remote I/O module

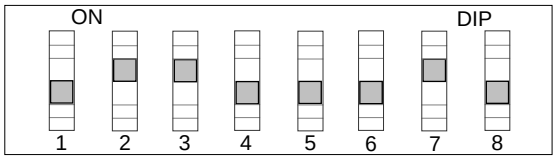
```
M8002
 |--[ZRST  D0    D255 ]
 |--[SET   M8161 ]  8 bits operation mode
 |--[MOV   H6087 D8120]  ModbusRTU, 9600, 8, E, 1
 |--[MOV   K0    D8121]  PLC station number 0 (master)
 |--[MOV   K200  D8129]  time-out monitor
X007
 |--/--[MOV   K1    D10 ]  station number 1 (slave)
 |--[MOV   H0006 D11 ]  write command
 |--[MOV   H0000 D12 ]  address Upper byte
 |--[MOV   H0004 D13 ]  address Lower byte
 |--[MOV   H0000 D14 ]  data Upper byte
 |--[MOV   H0084 D15 ]  data Lower byte
 |--[MOV   K6    D98 ]  number of send bytes
 |--[MOV   K64   D99 ]  number of receive bytes
X007
 |--[MOV   K1    D10 ]  station number 1 (slave)
 |--[MOV   H0003 D11 ]  read command
 |--[MOV   H0000 D12 ]  address Upper byte
 |--[MOV   H0000 D13 ]  address Lower byte
 |--[MOV   H0000 D14 ]  number of data Upper (word)
 |--[MOV   H0002 D15 ]  number of data Lower (word)
 |--[MOV   K6    D98 ]  number of send bytes
 |--[MOV   K64   D99 ]  number of receive bytes
M8000
 |--[BMOV  D10   D100  D98 ]  move data to buffer
 |--[RS    D100  D98   D200  D99 ]  assign send/receive start address
M500
 |--[SET   M8122 ]  send start flag
M8123
 |--[RST   M8123 ]  receive completion flag
 |--[INC   D998  ]
 |--[PLS   M500  ]
M8128
 |--[RST   M8128 ]  CRC ERROR flag
 |--[INC   D999  ]
 |--[END         ]
```

♦ At ModBus RTU mode, number of send data must be set correctly and communication format has to be no STX/ETX.

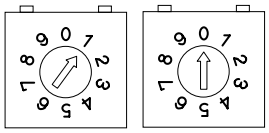
♦ Data of error check is not included to number of send bytes. It is counted by PLC automatically, and result is stored to next two registers.

<< MODBUS ASCII >> LRC error check mode

♦ switch of EXRM0808R/T



SW1



RSW1 RSW2

Ex: application note of Master and Remote I/O module

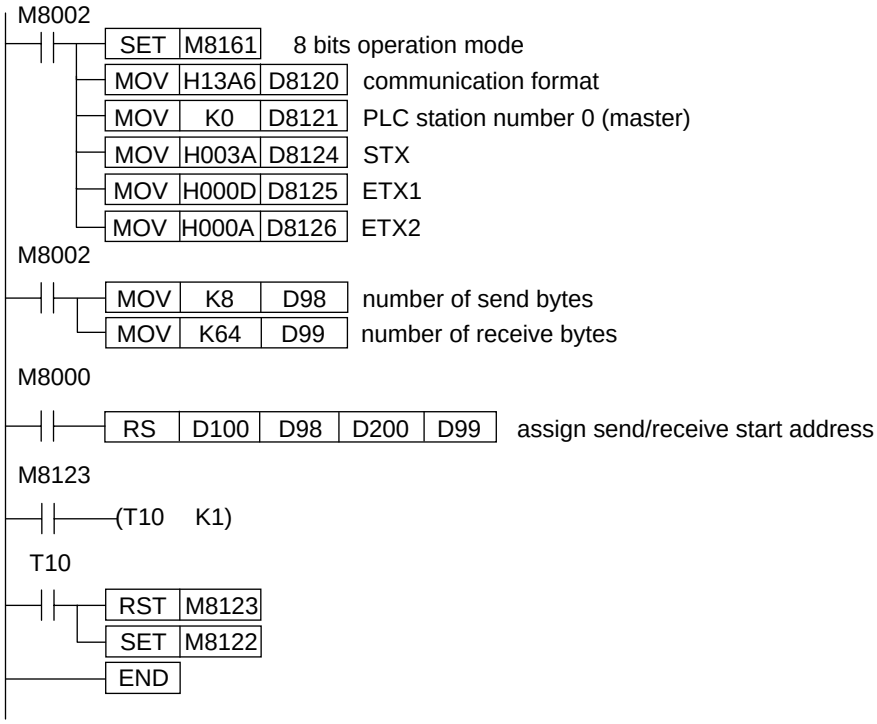
```
M8002
 |-[ZRST  D0    D255]
 |-[SET   M8161]   8 bits operation mode
 |-[MOV   H3386  D8120]  ModbusAscii, 9600, 7, E, 1
 |-[MOV   K0     D8121]  PLC station number 0 (master)
 |-[MOV   H003A  D8124]  STX
 |-[MOV   H000D  D8125]  ETX1
 |-[MOV   H000A  D8126]  ETX2
 |-[MOV   K200   D8129]  time-out monitor
X007
 |/[MOV   K1     D10]    station number 1 (slave)
 |-[MOV   H0006  D11]    write command
 |-[MOV   H0000  D12]    address Upper byte
 |-[MOV   H0004  D13]    address Lower byte
 |-[MOV   H0000  D14]    data Upper byte
 |-[MOV   H0084  D15]    data Lower byte
 |-[MOV   K6     D98]    number of send bytes
 |-[MOV   K64    D99]    number of receive bytes
X007
 |-[MOV   K1     D10]    station number 1 (slave)
 |-[MOV   H0003  D11]    read command
 |-[MOV   H0000  D12]    address Upper byte
 |-[MOV   H0000  D13]    address Lower byte
 |-[MOV   H0000  D14]    number of data Upper (word)
 |-[MOV   H0002  D15]    number of data Lower (word)
 |-[MOV   K6     D98]    number of send bytes
 |-[MOV   K64    D99]    number of receive bytes
M8000
 |-[BMOV  D10    D100  D98]      move data to buffer
 |-[RS    D100   D98   D200  D99]  assign send/receive start address
M500
 |-[SET   M8122]  send start flag
M8123
 |-[RST   M8123]  receive completion flag
 |-[INC   D998]
 |-[SET   M8122]
M8129
 |-[RST   M8129]  LRC ERROR flag
 |-[INC   D999]
 |-[END]
```

♦ At ModBus Ascii mode, number of send data must be set correctly and communication format has to be STX/ETX.

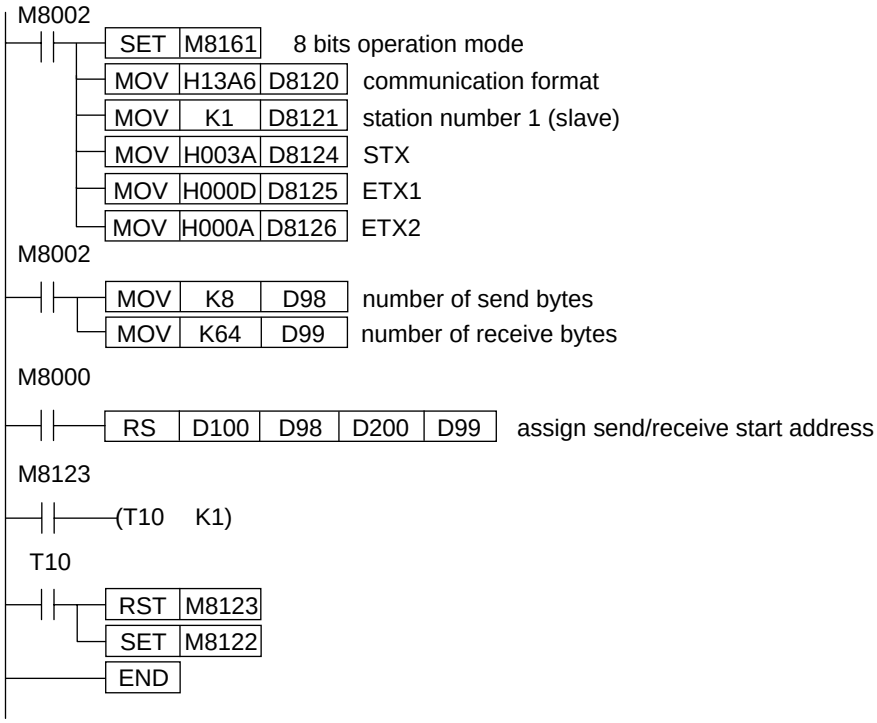
♦ Data of error check is not included to number of send bytes. It is counted by PLC automatically, and result is stored to next two registers.

<< User Defined Mode >> user defined error check

Ex1: application note of master at Ascii mode



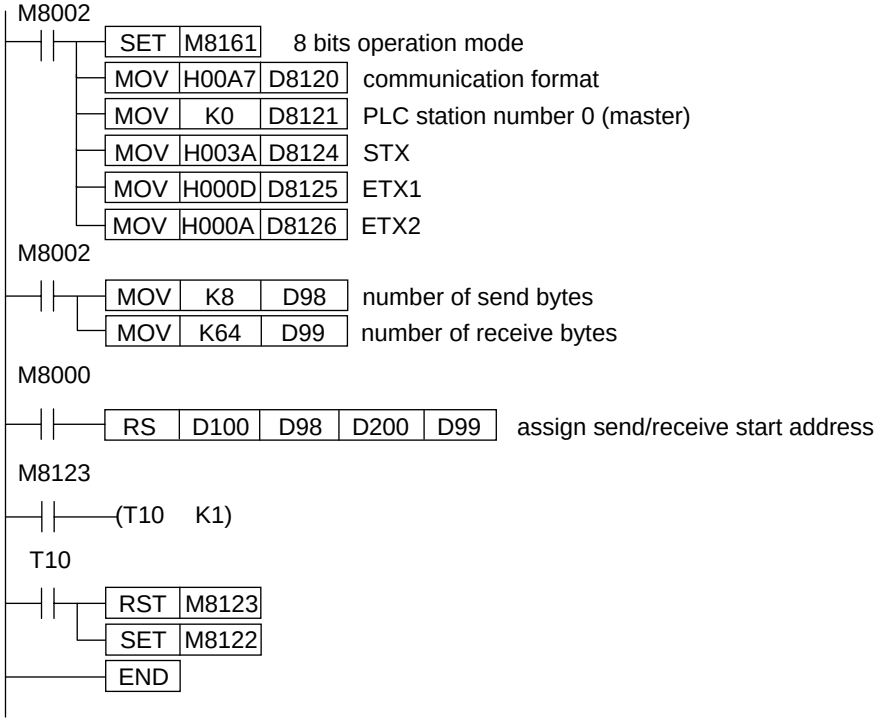
Application note of Slave



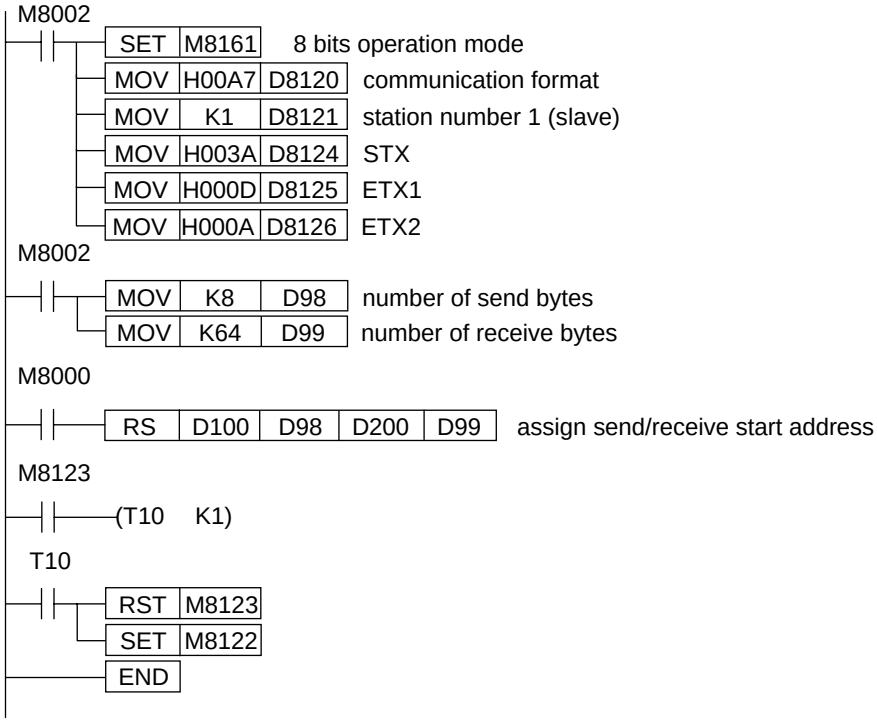
♦ At this mode, data of error check is counted by program designer, PLC do not calculate automatically.

♦ Send data must to be converted to Ascii and is stored to send area.

Ex2: application note of master at HEX mode



Application note of slave



♦ At this mode, data of error check is counted by designer, PLC do not calculate automatically.

Parallel Running

FNC(81)			16 bits: PRUN(P) - 5 Steps
D	PRUN	P	32 bits: (D)PRUN(P) -9 Steps

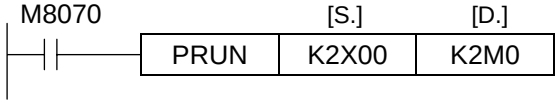
		J1n	J2n--

Operands: [S.]: KnX, KnM the lowest bit device is ”0”

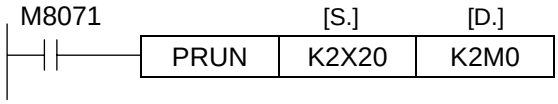
[D.]: KnM, KnY the lowest bit device is ”0”

Flag: M8073, M8129

Master program M8070=1, [S.] [D.] is pseudo operand



Slaver program M8071=1, [S.] [D.] is pseudo operand



- ♦ The content of D490~D497 of the master will transmit to D490~D497 of the slaver (M8070=1).
- ♦ The content of D500~D507 of the slaver will transmit to D500~D507 of the master (M8070=0).
- ♦ This instruction just set the status of M8070 and M8071, don’t need to assign data register (D), then will auto communicate.
- ♦ Because only the data register communicate each other, just used MOV to execute conversion, then input relay of master can control the output relay of slaver, and the input relay of slaver can control the master.
- ♦Relative parameter

  M8122: start communication transmitted flag.

  M8123: receive finished flag

  M8070: master flag

  M8071: slaver flag

  M8129: sum check error flag

  M8073: overtime flag

  D8070: overtime register(ms)

  D8072: communication taking time(ms)
- ♦ Example program please refer to EXPLC Application Note F081。
- ♦ When PRUN instruction used, then can’t use RS instruction.

Hex To Ascii Conversion

FNC(82)			16 bits: ASCI(P) - 7 Steps
	ASCI	P	

		J1n	J2n--

Operands: [S.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.]

n = (1 ~ 64)

Flag:

	[S.]	[D.]	n
ASCI	D100	D200	K4

- The hexadecimal data of source [S.] to be converted ASCII code and stored into upper/lower byte of destination device [D.] for n number of bytes.
- When M8161=OFF, 16 bits operation mode.
  example: (D100)=0ABCH, (D101)=1234H

	K1	K2	K3	K4	K5	K6	K7	K8
D200 down	“C”	“B”	“A”	“0”	“4”	“3”	“2”	“1”
D200 up		“C”	“B”	“A”	“0”	“4”	“3”	“2”
D201 down			“C”	“B”	“A”	“0”	“4”	“3”
D201 up				“C”	“B”	“A”	“0”	“4”
D202 down					“C”	“B”	“A”	“0”
D202 up						“C”	“B”	“A”
D203 down							“C”	“B”
D203 up								“C”

X10

(M8161) 8 bits operation mode

	[S.]	[D.]	n
ASCI	D100	D200	K4

M8000

(M8161) when M8161 = ON, then 8 bits mode

16 bits

0	lower 8 bits

Data of destination

- The hexadecimal data of source [S.] to be converted ASCII code and stored into lower byte of destination device [D.] for n number of bytes.
- When M8161=ON, 8 bits operation mode.
  example: (D100)=0ABCH, (D101)=1234H

	K1	K2	K3	K4	K5	K6	K7	K8
D200 down	“C”	“B”	“A”	“0”	“4”	“3”	“2”	“1”
D201 down		“C”	“B”	“A”	“0”	“4”	“3”	“2”
D202 down			“C”	“B”	“A”	“0”	“4”	“3”
D203 down				“C”	“B”	“A”	“0”	“4”
D204 down					“C”	“B”	“A”	“0”
D205 down						“C”	“B”	“A”
D206 down							“C”	“B”
D207 down								“C”

Ascii To Hex Conversion

FNC(83)			16 bits: HEX(P) - 7 Steps
	HEX	P	

		J1n	J2n--

Operands: [S.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.]

n = (1 ~ 64)

Flag:

	[S.]	[D.]	n
HEX	D200	D100	K4

♦ The ASCII code of the upper/lower byte in source [S.] to be converted to the hexadecimal data and stored into the destination device [D.] for n number byte.

♦ When M8161=OFF, 16 bits operation mode.

Ex.: D200 down =”0”, D200 up =”A”, D201 down =”B”, D201 up =”C”
D202 down =”1”, D202 up =”2”, D203 down =”3”, D203 up =”4”

	D102	D101	D100
K1			0H
K2			0AH
K3			0ABH
K4			0ABCH
K5		0H	ABC1H
K6		0AH	BC12H
K7		0ABH	C123H
K8		0ABCH	1234H

X10 (M8161) 8 bits mode

	[S.]	[D.]	n
HEX	D200	D100	K4

M8000 (M8161) when M8161 = ON, then 8 bits mode

16 bits

ignore	Lower 8 bits

source data

♦ The ASCII code of the lower byte in source [S.] to be converted to the hexadecimal data and stored into the destination device [D.] for n number byte.

♦ When M8161=ON, 8 bits operation mode.

Ex: D200=”0”, D201=”A”, D202=”B”, D203=”C”
D204=”1”, D205=”2”, D206=”3”, D207=”4”

	D102	D101	D100
K1			0H
K2			0AH
K3			0ABH
K4			0ABCH
K5		0H	ABC1H
K6		0AH	BC12H
K7		0ABH	C123H
K8		0ABCH	1234H

Check Code

FNC(84)			16 bits: CCD(P) - 7 Steps
	CCD	P	

		J1n	J2n--

Operands:

[S.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

n = 1~64

[D.]

Flag:

M8000	[S.]	[D.]	[n.]
CCD	D100	D0	K10

♦Calculation the data of n bytes (16 bits) from the head address of source [S.], then put the Sum→D00, Vertical Parity→D01([D.]+1).

M8161=OFF 16 bit mode									
(S.)		Bit Pattern							
D100 L	K100	0	1	1	0	0	1	0	0
D100 H	K111	0	1	1	0	1	1	1	1
D101 L	K100	0	1	1	0	0	1	0	0
D101 H	K98	0	1	1	0	0	0	1	0
D102 L	K123	0	1	1	1	1	0	1	1
D102 H	K66	0	1	0	0	0	0	1	0
D103 L	K100	0	1	1	0	0	1	0	0
D103 H	K95	0	1	0	1	1	1	1	1
D104 L	K210	1	1	0	1	0	0	1	0
D104 H	K88	0	1	0	1	1	0	0	0
Vertical parity		1	0	0	0	0	1	0	1
Sum	K1091								

X10

(M8161) M8161=ON, 8 bits operation mode.

	[S.]	[D.]	n
CCD	D100	D 0	K4

M8000

(M8161)

16 bits

ignore	Lower 8 bits

source data

♦Calculation the data of n bytes (8 bits) from the head address of source [S.], then put the Sum→D00, Vertical Parity→D01([D.]+1).

M8161=ON 8 bit mode									
(S.)		Bit Pattern							
D100	K100	0	1	1	0	0	1	0	0
D101	K111	0	1	1	0	1	1	1	1
D102	K100	0	1	1	0	0	1	0	0
D103	K98	0	1	1	0	0	0	1	0
D104	K123	0	1	1	1	1	0	1	1
D105	K66	0	1	0	0	0	0	1	0
D106	K100	0	1	1	0	0	1	0	0
D107	K95	0	1	0	1	1	1	1	1
D108	K210	1	1	0	1	0	0	1	0
D109	K88	0	1	0	1	1	0	0	0
Vertical parity		1	0	0	0	0	1	0	1
SUM	K1091								

Volume Read

FNC(85)			16 bits: VRRD(P) - 5 Steps
	VRRD	P	

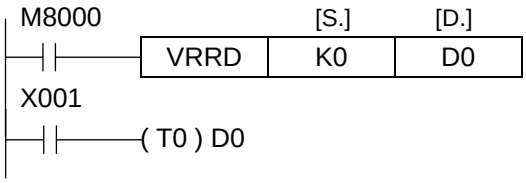
		J1n	J2n--

Operands:

[D.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[S.] = (0 – 3)



- The identified volume [S.] of the master unit is read as an analog input and converted to 8 bits binary code (0-255) stored into the destination device [D.].
- The content of [D.] can as Timer data or Counter data.

Volume Scale

FNC(86)			16 bits: VRSC(P) - 5 Steps
	VRSC	P	

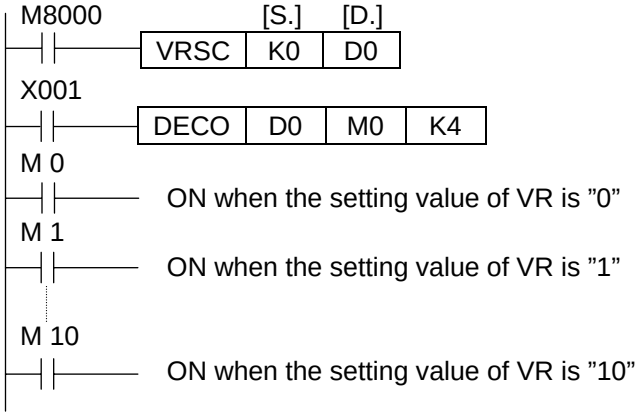
		J1n	J2n--

Operands:

[D.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[S.] = (0 – 3)



- The identified volume [S.] of the master unit is read as an analog input and converted to 8 bits binary code (0-255) then divided 16, the result (0-15) stored into the destination device [D.].
- This function the volume can as a 16 (0-15) position rotary switch.

PID

FNC(88)	16 bits: PID - 9 Steps
PID	

		J1n	J2n--

Operands: [S1.][S2.][S3.] ⟷

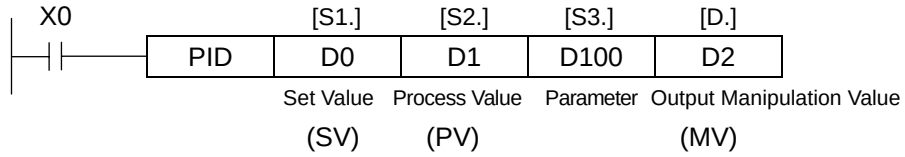
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

≤ [D.] ≥

Operands:

X	Y	M	S

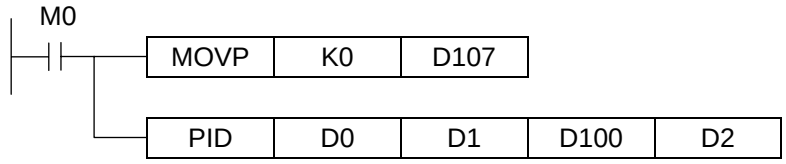
Flag:



[S1.] : Set Value
[S2.] : Process Value
[S3.] ~ [S3.]+6 : Control Parameter
} Use setting execute program as left mentioned, and stored the result (MV) into [D]
[D.] : Output manipulation value data register

♦ It will occupy continuous 25 devices from assigned [S3.]. In this example, it occupies D100 ~ D124.

♦ When execute in first time, have to clear the content of [S3.]+7 to be 0.



♦ Before execute PID operation, have to use MOV command to write the parameter set value for PID control first.

[S3.]	Sampling Time (Ts)	1~32767 (ms) (can't set shorter than scan-time)
[S3.] + 1	Act direction (ACT)	BIT0 : 0 : forward action ; 1 : reverse action
		BIT1 : 0 : Without input change Alarm ; 1 : With input change Alarm
		BIT2 : 0 : Without output change Alarm ; 1 : With output change Alarm
		BIT3 : reserved
		BIT4 : reserved
		BIT5 : 0 : Without output limit ; 1 : With output limit
		BIT6 ~ BIT15 : reserved
[S3.] + 2	Input Filter (α)	0 ~ 99 (%)
[S3.] + 3	Proportion Constant (Kp)	1 ~ 32767 (%)
[S3.] + 4	Integral Time Constant (Ti)	1 ~ 32767 (x 100ms), 0 is without integral action
[S3.] + 5	Derivative Filter Constant (Kd)	0 ~ 100 (%)
[S3.] + 6	Time Derivative Constant (Td)	1 ~ 32767 (x 10ms), 0 is without derivative action
[S3.] + 7 ~ [S3.] + 19	} For internal operation when execute PID	
[S3.] + 20	System reserved	
[S3.] + 21	System reserved	
[S3.] + 22	Output maximum value limitation, it is effective when [S3.]+1, BIT5=1	
[S3.] + 23	Output minimum value limitation, it is effective when [S3.]+1, BIT5=1	
[S3.] + 24	System reserved	

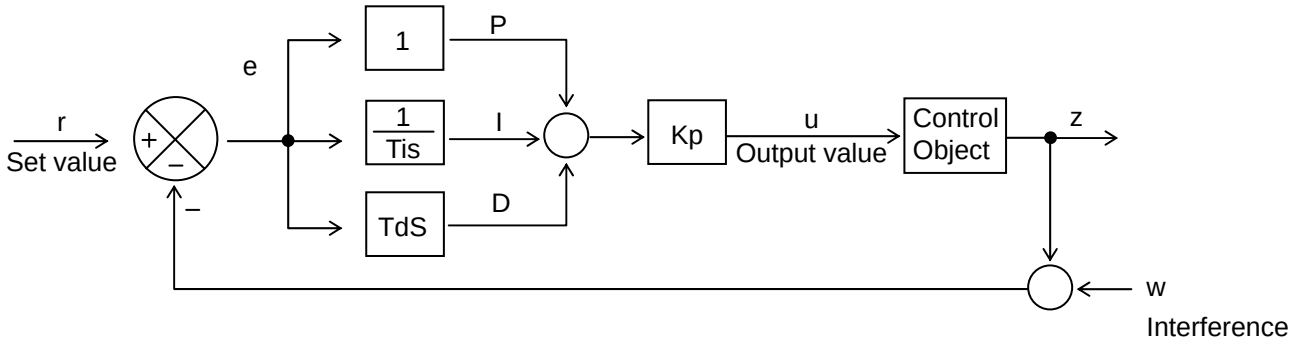
♦ Basic operation of PID instruction :

This instruction is based on speed form, measure Derivative calculation formula to execute PID operation.

In PID control, execute operation formula of forward action or reverse action according to the content of “Act direction” which is assigned by [S3.].

PID basic formula:

Output $u(t) = Kp \{ e(t) + \frac{1}{Ti} \int_0^t e(t)dt + Td \frac{de(t)}{dt} \}$ e(t) = error value



FNC(89)							

FNC(90)							

FNC(91)							

FNC(92)							

FNC(93)							

FNC(94)							

FNC(95)							

FNC(96)							

FNC(97)							

FNC(98)							

FNC(99)							

Floating Point Compare

FNC(110)			
D	ECMP	P	32 bits:(D)ECMP & (D)ECMP(P) - - - - - - - - - - - 13 steps

			J2n--

Operands: |<[S1.]>|

K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

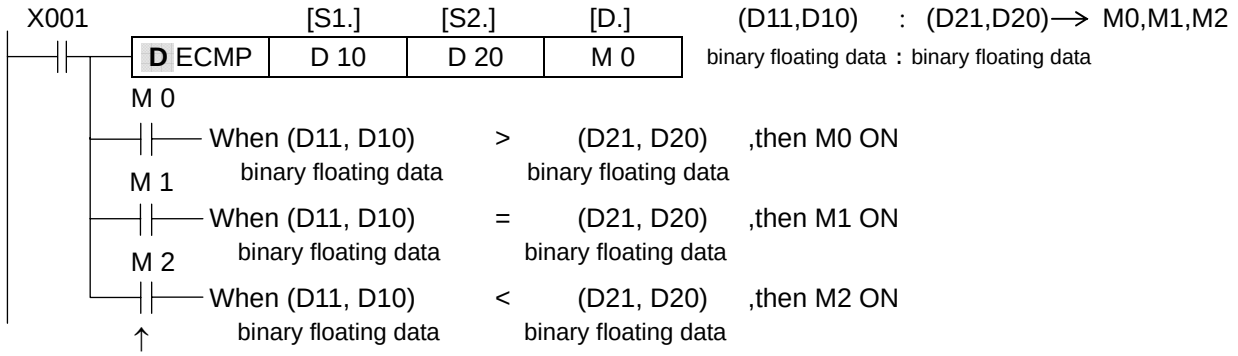
|<[S2.]>|

Operands: |<——— [D.] ———>|

X	Y	M	S

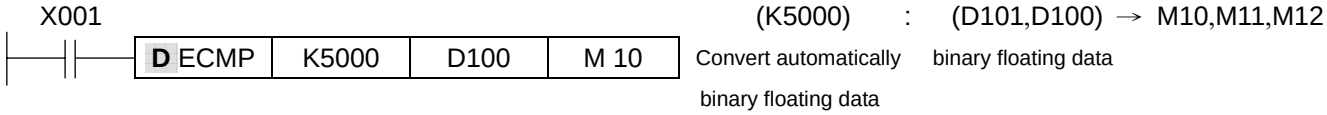
The result is indicated by 3 bit devices specified with the head address entered as D.

Flag: none



When X001 OFF, then not execute ECMP, M0~M2 status unchanged

- ♦ Compare the binary floating data of the source devices [S1.] and [S2.], this will automatic ON/OFF 3 bit devices from the head address of [D.].
- ♦ When source operand assigned by constant K or H, it will be converted to binary floating data automatically.



Floating Point Zone Compare

FNC(111)			
D	EZCP	P	32 bits:(D)EZCP & (D)EZCP(P) - - - - - - - - - - - - 17 steps

			J2n--

Operands: |←→| [S1.] [S2.] [S.] [S1.] [S2.] [S.] |←→|

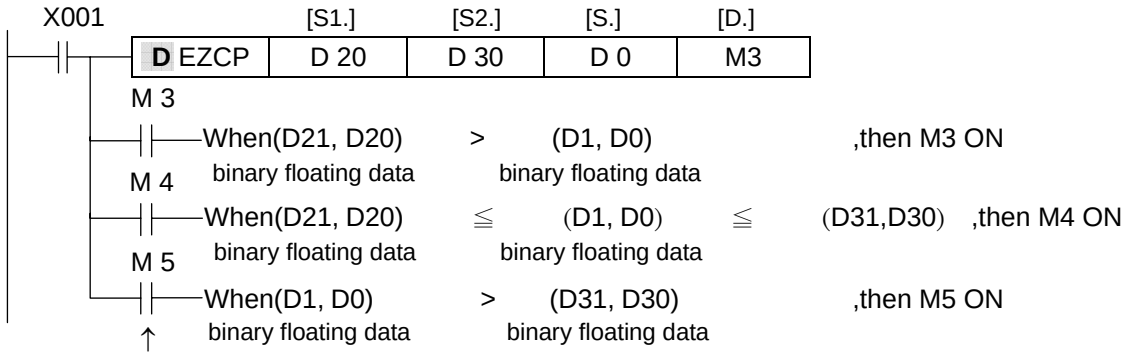
K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands: |← [D.] →|

X	Y	M	S

[D.] occupy 3 bit devices from the head address,[S1.]、[S2.]set [S1.]≦[S2.]

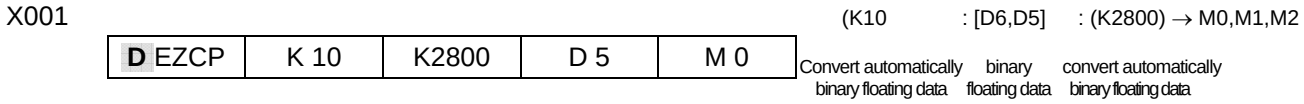
Flag: None



If X001OFF, then not execute ECMP, M3~M5 status unchanged.

♦ The result will automatically set 3 bit devices from the head address of [D.]

♦ When source operand assigned by constant K or H, it will be converted to binary floating data automatically.



♦Set [S1.] ≦ [S2.], if [S1.] > [S2.], then data of [S2.] is as same as data of [S1.].

Float to Scientific conversion

FNC(118)			
D	EBCD	P	32 bits:(D)EBCD & (D)EBCD(P) - - - - - - - - - - - - - 9 steps

Operands:

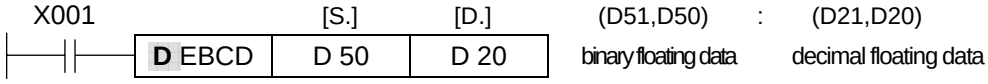
K,H	KnX	KnY	KnM	KnS	T	C	D [S.]	V,Z

D [D.]

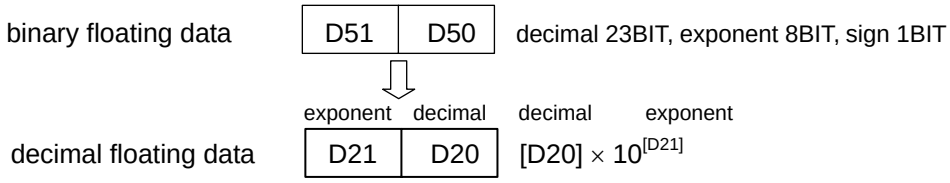
Operands:

X	Y	M	S

Flag:



♦ Convert binary floating data assigned by source device to decimal floating data, and store it to destination device.



♦Reserved

Scientific to Float conversion

FNC(119)			
D	EBIN	P	32 bits:(D)EBIN & (D)EBIN(P) - - - - - - - - - - - - - 9 steps

Operands:

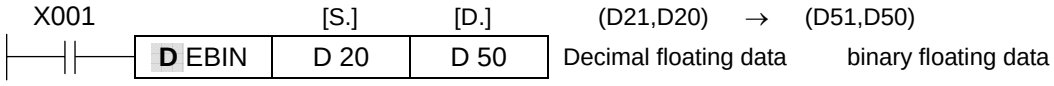
K,H	KnX	KnY	KnM	KnS	T	C	D [S.]	V,Z

D [D.]

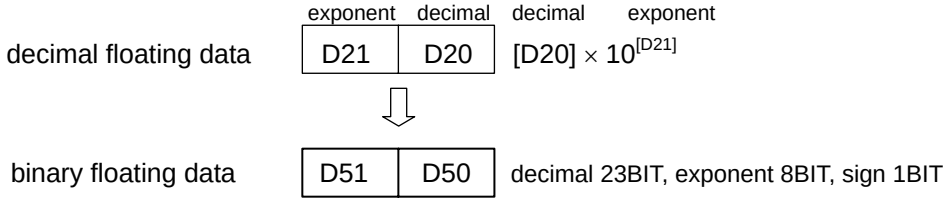
Operands:

X	Y	M	S

Flag:



♦ Convert binary floating data assigned by source device to decimal floating data, and store it to destination device.



♦ Reserved

Floating Point Addition

FNC(120)			
D	EADD	P	32 bits:(D)EADD & (D)EADD(P) - - - - - - - - - - - - 13 steps

				J2n--

Operands: ←→ [S1.] [S2.] [S1.] [S2.] ←→

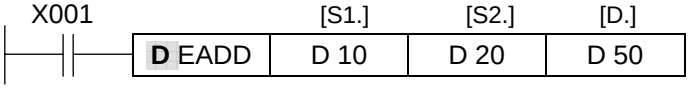
K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

←[D.]→

Operands:

X	Y	M	S

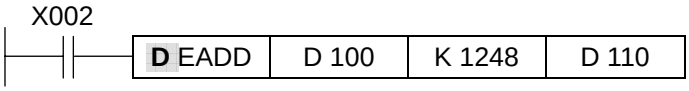
Flag: None



(D11,D10) + (D21,D20) → (D51, D50)
binary floating data binary floating data binary floating data

♦ Two devices of binary floating data are added, then the result stored by form of binary floating data at destination device.

♦ When source operand assigned by constant K or H, it will be converted to binary floating data automatically.



(D101,D100) + (K1248) → (D111, D110)
binary floating data convert automatically to binary floating data binary floating data

♦ Enable assign source operand [S.] and destination operand [D.] to same device number.

Floating Point Subtraction

FNC(121)			
D	ESUB	P	32 bits:(D)ESUB & (D)ESUB(P) - - - - - - - - - - - - 13 steps

				J2n--

Operands: ←→ [S1.] [S2.] [S1.] [S2.] ←→

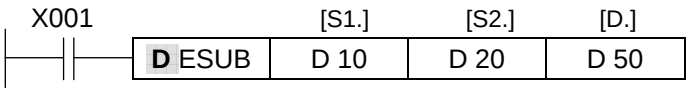
K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

←[D.]→

Operands:

X	Y	M	S

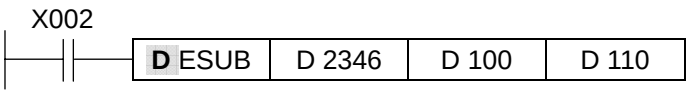
Flag: None



(D11,D10) − (D21, D20) → (D51, D50)
binary floating data binary floating data binary floating data

♦ Binary floating data of [S1.] subtract binary floating data of [S2.], then the result stored by form of binary floating data at destination device of [D.].

♦ When source operand assigned by constant K or H, it will be converted to binary floating data automatically.



(K2346) − (D101,D100) → (D111, D110)
Convert automatically to binary floating data binary floating data binary floating data

♦Enable assign source operand [S.] and destination operand [D.] to same device number.

Floating Point Multiplication

FNC(122)			
D	EMUL	P	32 bits:(D)EMUL & (D)EMUL(P) - - - - - - - - - - - - 13 steps

			J2n--

Operands: ←→ [S1.] [S2.] [S1.] [S2.] ←→

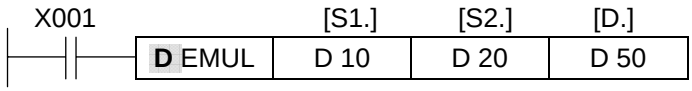
K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

←[D.]→

Operands:

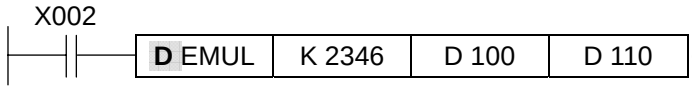
X	Y	M	S

Flag: None



(D11,D10) × (D21, D20) → (D51, D50)
binary floating data binary floating data binary floating data

- ♦ Two source devices, binary floating data of [S1.] is multiplied by binary floating data of [S2.], then the result stored by form of binary floating data at destination device of [D.]. .
- ♦ When source operand assigned by constant K or H, it will be converted to binary floating data automatically.



(K2346) × (D101,D100) → (D111, D110)
Convert automatically to binary floating data binary floating data binary floating data

Floating Point Division

FNC(123)			
D	EDIV	P	32 bits:(D)EDIV & (D)EDIV(P) - - - - - - - - - - - - 13 steps

			J2n--

Operands: ←→ [S1.] [S2.] [S1.] [S2.] ←→

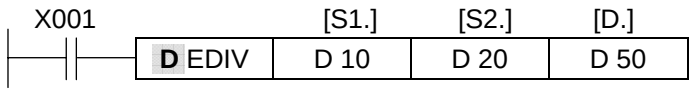
K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

←[D.]→

Operands:

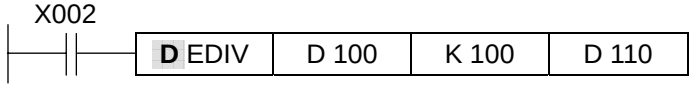
X	Y	M	S

Flag: None



(D11,D10) ÷ (D21, D20) → (D51, D50)
binary floating data binary floating data binary floating data

- ♦ The binary floating data of assignation device [S1.] is divided by binary floating data of assignation device [S2.], then the result stored by form of binary floating data at destination device of [D.].
- ♦ When source operand assigned by constant K or H, it will be converted to binary floating data automatically.



(D101,D100) ÷ (K100) → (D111, D110)
binary floating data convert automatically to binary floating data binary floating data

Floating Point Square Root

FNC(127)			
D	ESQR	P	32 bits:(D)ESQR & (D)ESQR(P) - - - - - - - - - - - 13 steps

			J2n--

Operands: |←[S.]→| |←[S.]→|

K,H	KnX	KnY	KnM	KnS	T	C	D	Z

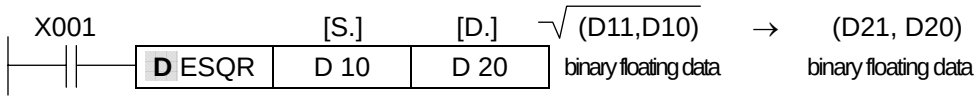
|←[D.]→|

Operands:

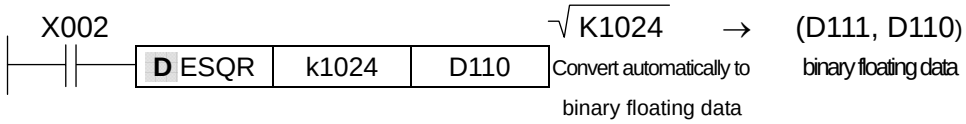
X	Y	M	S

The content of [S.] is positive number, then effective

Flag: M8020



- To be square root of binary floating data of source device [S.], then the result stored by binary floating data at destination device of [D.].
- When source operand assigned by constant K or H, it will be converted to binary floating data automatically.



- If the result is “0”, then zero flag M8020 will ON.
- The content of source operand is effective only when it is positive. If the number is negative, then error flag M8067 will ON and stop executing.

Float to Integer

FNC(129)			16 bits:INT & INT - 5 steps
D	INT	P	32 bits:(D)INT & (D)INT(P) - - - - - - - - - - - - - - - - -9 steps

			J2n--

Operands: |←[S.]→|

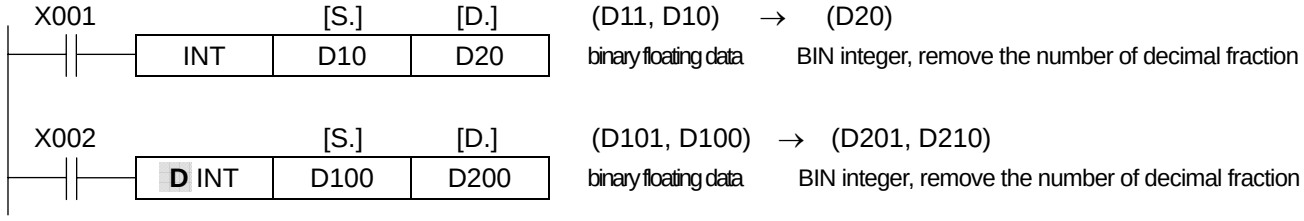
K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

|←[D.]→|

Operands:

X	Y	M	S

Flag:



- Convert binary floating data of assigned device [S.] to BIN integer, then store the result at destination device [D.]
- When the result is “0”, then zero flag M8020 will ON.
  When it converts and becomes 0 because of less than 1, borrow flag M8021 will ON.
  If the calculating result more than following limit, then will overflow and carry flag M8022 will ON.
  When 16 bit operation**:** -32,768~32,767
  When 32 bit operation**:** -2,147,483,648~2,147,483,647

Sine

FNC(130)			
D	SIN	P	32 bits:(D)SIN & (D)SIN(P) - - - - - - - - - - - - - - - - 9 steps

			J2n--

Operands:

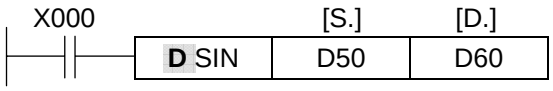
K,H	KnX	KnY	KnM	KnS	T	C	D ←[S.]→	V,Z ←[D.]→

Operands:

X	Y	M	S

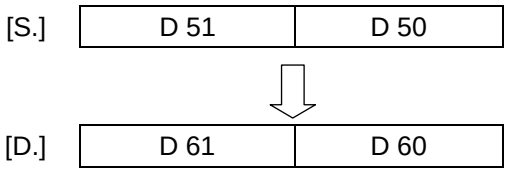
0° ≦ angle<360°

Flag



(D51, D50)RAD → (D61, D60)SIN
binary floating data binary floating data

♦ Used assigned radian (RAD) by source [S.] to get SIN value, then store the result at destination device [D.].

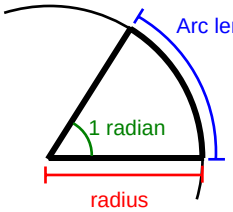


RAD value (angle x π /180)
assign to binary floating data

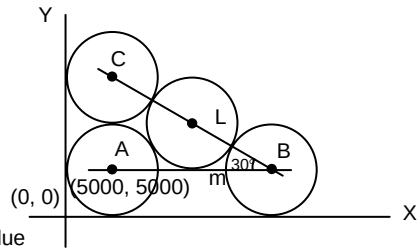
SIN value
binary floating data

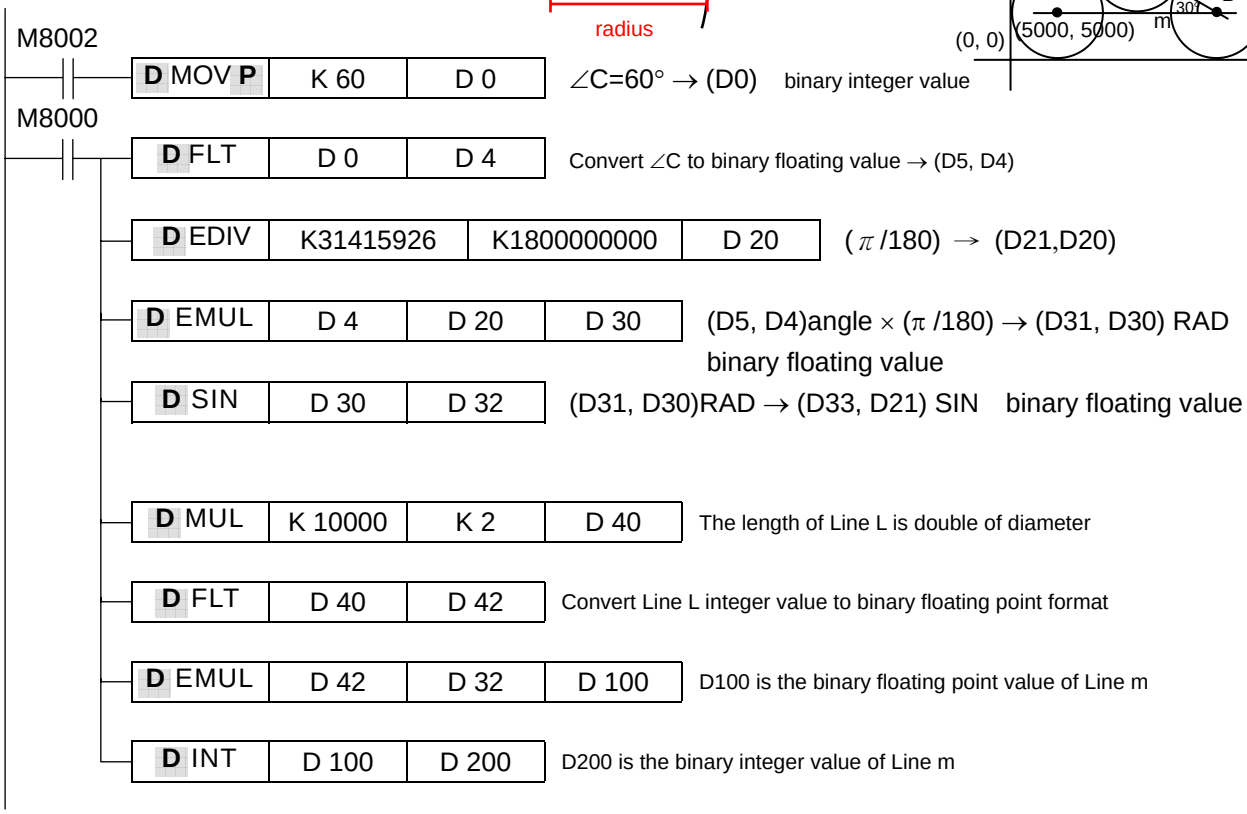
♦One radian : the angle subtended at the center of a circle by an arc is equal in length to the radius of the circle. It is also called “RAD”.

♦1 rad = 180/π° ; 1°=π/180 rad



♦To get length of m





∠C=60° → (D0) binary integer value

Convert ∠C to binary floating value → (D5, D4)

(π /180) → (D21,D20)

(D5, D4)angle × (π /180) → (D31, D30) RAD
binary floating value

(D31, D30)RAD → (D33, D21) SIN binary floating value

The length of Line L is double of diameter

Convert Line L integer value to binary floating point format

D100 is the binary floating point value of Line m

D200 is the binary integer value of Line m

Cosine

FNC(131)			
D	COS	P	32 bits:(D)COS & (D)COS(P) - - - - - - - - - - - - - 9 steps

			J2n--

Operands:

|←[S.]→|

K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

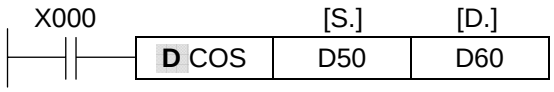
|←[D.]→|

Operands:

X	Y	M	S

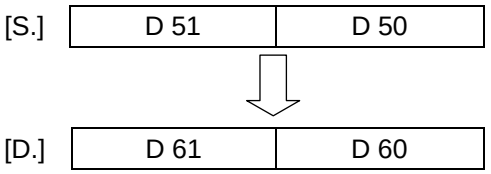
0° ≦ angle <360°

Flag:



(D51, D50)RAD → (D61, D60)COS
binary floating data binary floating data

♦ Used assigned angle (RAD) by source device [S.] to get COS value, then store the result at destination device [D.].



RAD value (angle x π /180) assign to binary floating data

COS value binary floating data

Tangent

FNC(132)			
D	TAN	P	32 bits:(D)TAN & (D)TAN(P) - - - - - - - - - - - - - - - 9 steps

			J2n--

Operands:

|←[S.]→|

K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

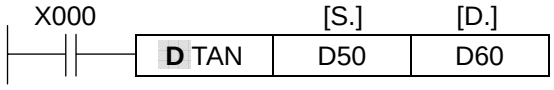
|←[D.]→|

Operands:

X	Y	M	S

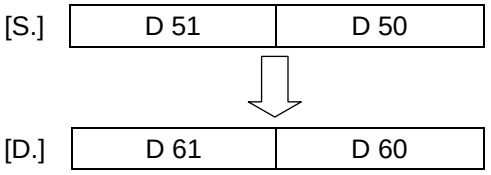
0° ≦ angle <360°

Flag:



(D51, D50)RAD → (D61, D60)TAN
binary floating data binary floating data

♦ Used assigned angle (RAD) by source device [S.] to get TAN value, then store the result at destination device [D.].



RAD value (angle x π /180) assign to binary floating data

TAN value binary floating data

Byte Swap

FNC(147)			16 bits:SWAP & SWAP(P) - - - - - - - - - - - - - - - - 5 steps
D	SWAP	P	32 bits:(D)SWAP & (D)SWAP(P) - - - - - - - - - - - -9 steps

		J1n	J2n--

Operands:

[S.]

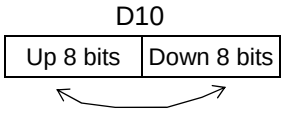
K,H	KnX	KnY	KnM	KnS	T	C	D	Z

Operands:

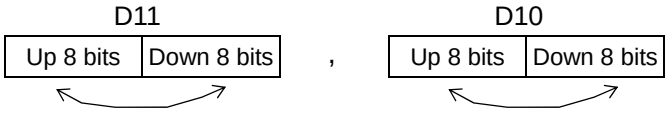
X	Y	M	S

Flag

♦ when 16bits, Down 8 bits and Up8 bits exchange.



♦ when 32 bits, Up 8 bits and Down 8 bits exchange

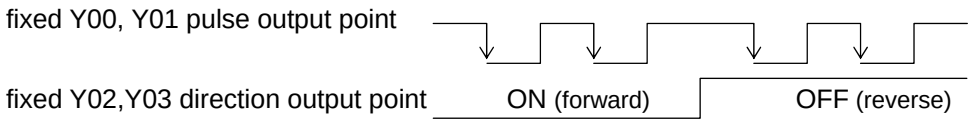


♦ If use continuative executing instruction, each scan cycle will execute to exchange, please pay attention.

♦ This instruction is as same as FNC17 (XCH) function of expanded.

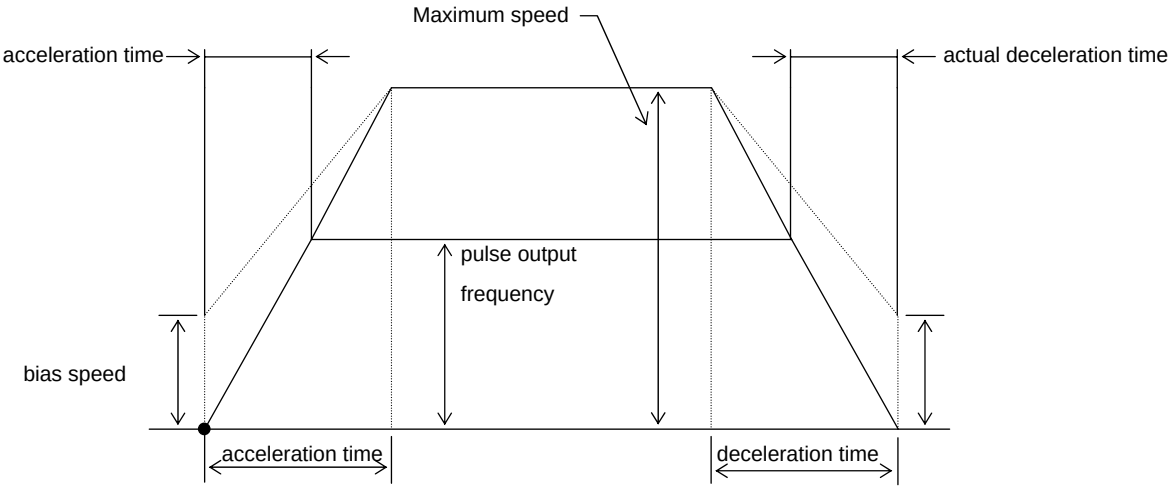
FNC150 – 159 Position Control

♦The Ex series of controller pulse output signal: pulse (negative logic) + sign, as following drawing



♦ The pulse duty cycle is 50% ON, 50% OFF

♦ Single position control. The curve condition of controller and relative device.



Absolute current value read

FNC(155)			16 bits:ABS - 7 steps
D	ABS		32 bits:(D)ABS - 11 steps

Operands:

[S.]

K,H	KnX	KnY	KnM	KnS	T	C	D	Z

Operands:

X	Y	M	S

Flag: M8029

Reserved

Zero return

FNC(156)			32 bits:(D)ZRN - 17 steps
D	ZRN		

		J1n	J2n--

Operands: [S1.]

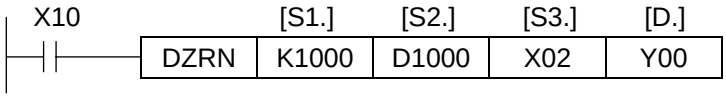
K,H	KnX	KnY	KnM	KnS	T	C	D	Z

[S2.]

Operands: [S3.] [D.]

X	Y	M	S

Flag: M8029



♦ [D.] assign pulse output point

[S1.] assign speed of zero-return search for DOG point (Home Speed) 10 ~ 200,000 pps。

[S2.] it will occupy continuous 100 words from assigned [S2.]. In this example, it occupies D1000~D1099.

[S2.]+0 : speed of search for zero-point 10~32,767 pps

[S2.]+1 : operation direction control point Y2~Y7

b7	b6	b5	b4	b3	b2	b1	b0	
								Assign operation direction output point (Y)
								System reserved

[S2.]+2 : parameter setting

b7	b6	b5	b4	b3	b2	b1	b0	
								System reserved
								Without slope stop flag (X10 OFF stop effective)
								Continuous motion flag
								System reserved
								Change status of operation control point 0:ON Up OFF Down; 1:ON Down OFF Up
								System reserved

[S2.]+3 ~ [S2.]+99 : as same as FNC(59) PLSR 的[S3.]+3 ~ [S3.]+99

[S3.] assign DOG point input signal. Effective range X00~X07 (pulse catch signal M8170~M8177)。

Zero-return signal is set by [S2.]+24.

♦ When ZRN command is executed, zero-return point busy flag M8154~M8157 will be set automatically to avoid drive DRVI, DRVA at the same time.

♦ This command Y00 ~ Y03 can be used once and has to select transistor output module.

♦ It is fixed to 32 bits operation. If user assigns 16 bits operation mode, then error 6509 will be occurred.

Pulse V

FNC(157)			
D	PLSV		32 bits:(D)PLSV - 13 steps

		J1n	J2n--

Operands: |< [S.] >|

K,H	KnX	KnY	KnM	KnS	T	C	D	Z

Operands: |<[D2.]>|

X	Y	M	S

|<[D1.]>|

Flag: M0

M0	[S.]	[D1.]	[D2.]
DPLSV	D100	Y00	Y02

♦ [D1.] assign operation pulse output point. (It is fixed to Y00~Y03 as output point)
[D2.] assign operation direction output point.. (It is fixed to Y02~Y07 as output point)
[S.] It will occupy continuous 100 words start from assigned [S.]. In this example, it occupies D1000~D1099.
[S.]+1, [S]+0 : assign output frequency. [32bits]:10 ~ 200,000 Hz
[S.]+2 : Parameter setting

b7	b6	b5	b4	b3	b2	b1	b0

System reserved
Without slope stop flag (M0 OFF stop effective)
Continuous motion flag
System reserved
Chang status of operation control point 0:ON Up OFF Down; 1:ON Up OFF Down
System reserved

[S.]+41, [S.]+40 : PLSV number of output pulses. Value = 0 is without target operation.
[S.]+52 : Mark signal

b7	b6	b5	b4	b3	b2	b1	b0

Mark signal X0~X7
0:upper effective 1:lower effective
0:without target function 1:with target function

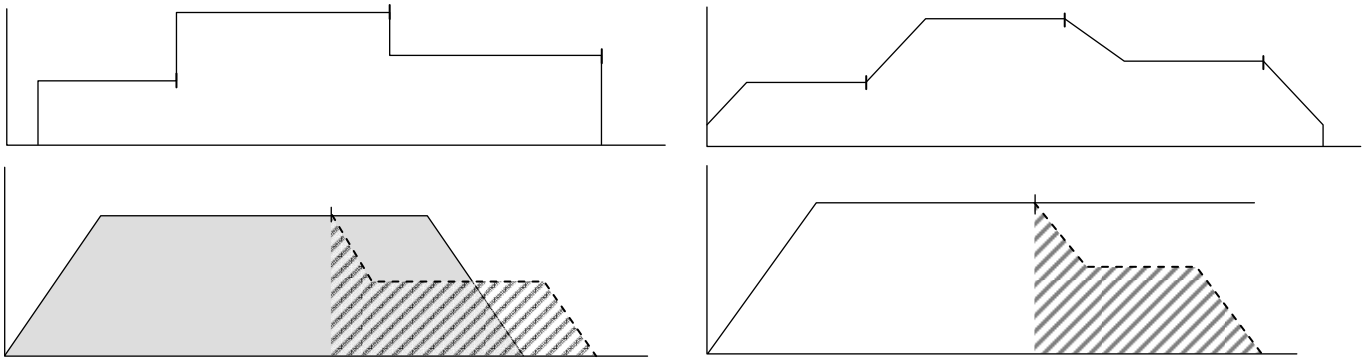
[S.]+3 ~ [S.]+99 : as same as FNC(59) PLSR [S3.]+3 ~ [S3.]+99

♦ When PLSV command is executed, busy flag M8142~M8145 will be set automatically.

♦ Value of [S.] can be changed during pulse output, but symbol (+,-) can not be changed. If drive contact OFF, then decelerate to bias speed stop directly.

♦ It is fixed to 32 bits operation. If user assigns 16 bits operation mode, then error 6509 will be occurred.

♦ Following modes can be achieved,



Drive to increment

FNC(158)		
D	DRVI	

32 bits:(D)DRVI - 17 steps

		J1n	J2n--

Operands: [S1.]

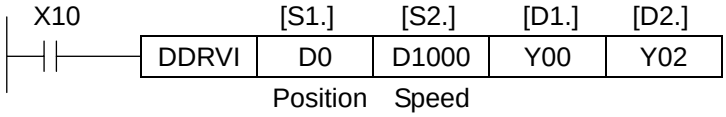
K,H	KnX	KnY	KnM	KnS	T	C	D	Z

[S2.]

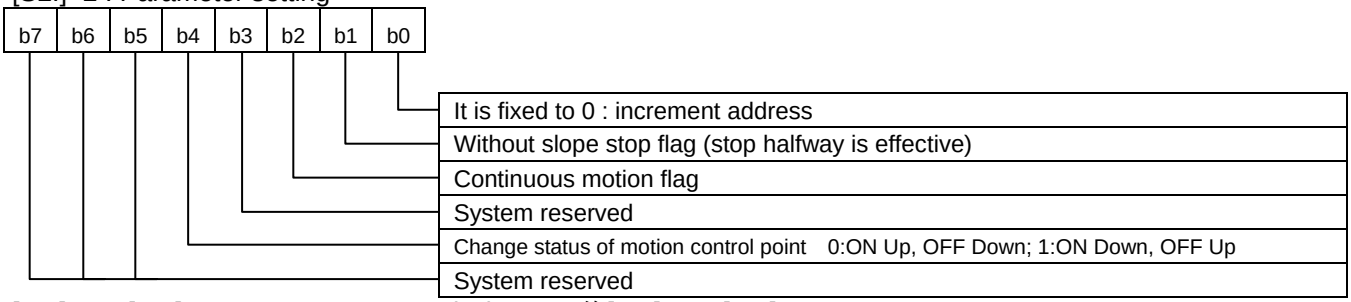
Operands: [D1.] [D2.]

X	Y	M	S

Flag: M8029

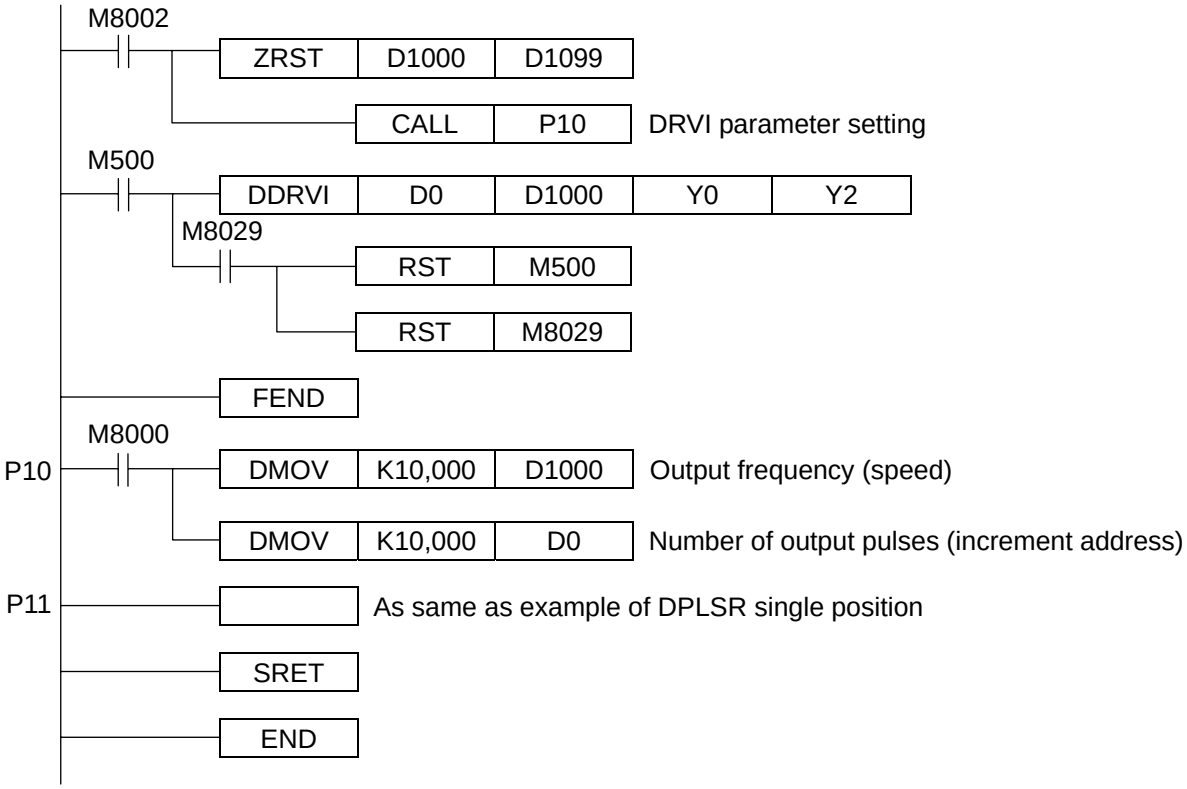


♦ [D1.] assign operation pulse output point. (only Y00~Y03).
[D2.] assign operation direction output point. (only Y02~Y07).
[S1.] assign number of output pulses for increment address. (positive value: forward; negative value: reverse)
[S2.] It will occupy continuous 100 words start from assigned [S2.]. In this example, it occupies D1000~D1099.
[S2.]+1, [S2]+0 : assign output frequency. [32bits]:10 ~ 200,000 Hz.
[S2.]+2 : Parameter setting



[S2.]+3 ~ [S2.]+99 : as same as FNC(59) PLSR 的[S3.]+3 ~ [S3.]+99

♦ This instruction for Y0~Y3 only can be used once, and has to select transistor output module.
♦ When DDRVI are executed, busy flag M8146~M8149 will be set automatically by system.
♦ During output pulse, to modify value of [S1], [S2]+1, [S2]+0 is ineffective.
♦ It is fixed to 32 bits operation. If user assigns 16 bits operation mode, then error 6509 will be occurred.



Drive to absolute

FNC(159)			
D	DRVA		32 bits:(D)DRVA - 17 steps

		J1n	J2n--

Operands: [S1.]

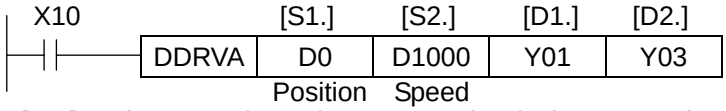
K,H	KnX	KnY	KnM	KnS	T	C	D	Z

[S2.]

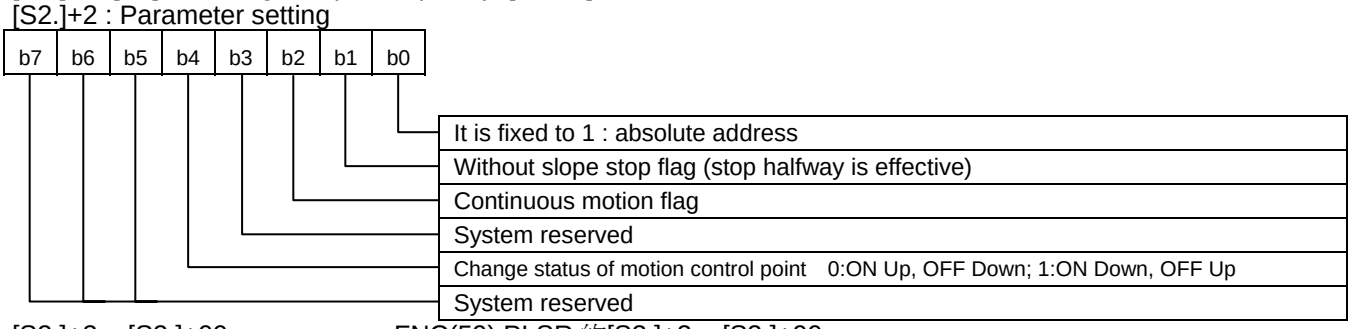
Operands: [D1.] [D2.]

X	Y	M	S

Flag: M8029

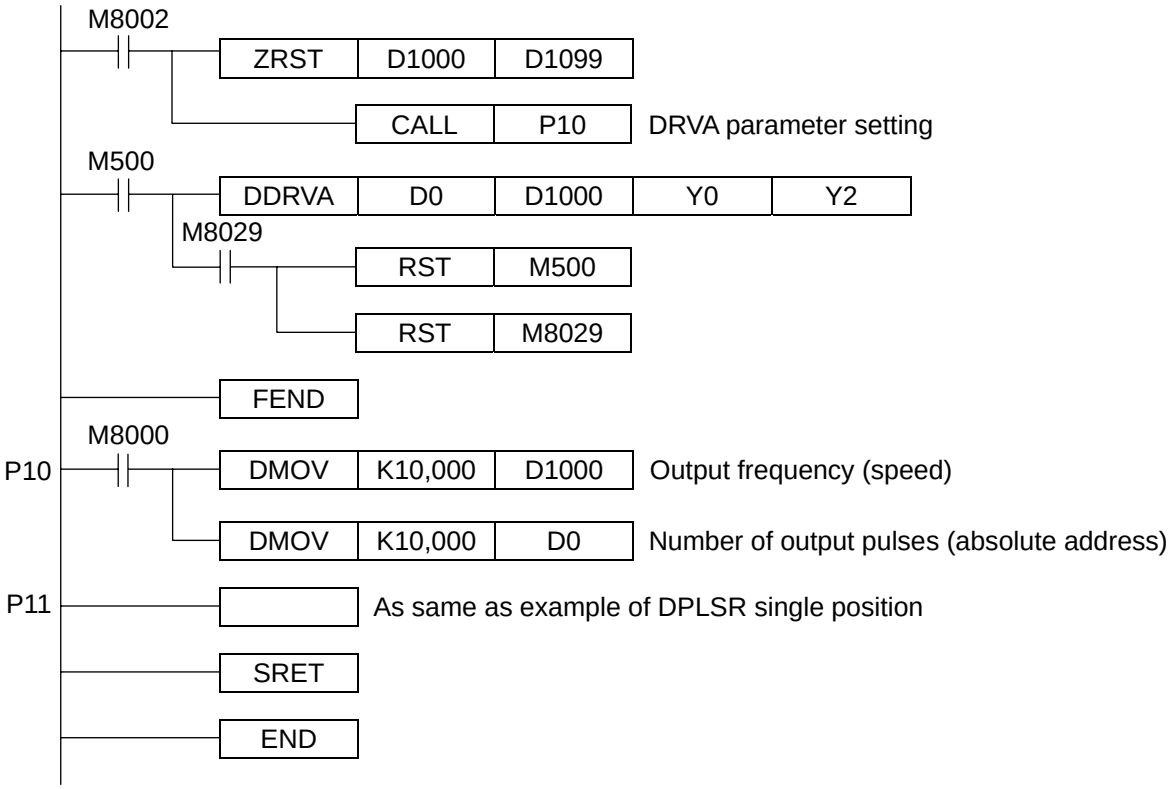


♦ [D1.] assign operation pulse output point. (only Y00~Y03).
[D2.] assign operation direction output point. (only Y02~Y07).
[S1.] assign number of output pulses for absolute address. (compare with bias address to decide forward or reverse)
[S2.] It will occupy continuous 100 words start from assigned [S2.]. In this example, it occupies D1000~D1099.
[S2.]+1, [S2]+0 : assign output frequency. [32bits]:10 ~ 200,000 Hz.
[S2.]+2 : Parameter setting



[S2.]+3 ~ [S2.]+99 : as same as FNC(59) PLSR 的[S3.]+3 ~ [S3.]+99

♦ This instruction for Y0~Y3 only can be used once, and has to select transistor output module.
♦ When DDRVA are executed, busy flag M8150~M8153 will be set automatically by system.
♦ During output pulse, to modify value of [S1], [S2]+1, [S2]+0 is ineffective.
♦ It is fixed to 32 bits operation. If user assigns 16 bits operation mode, then error 6509 will be occurred.



Time Compare

FNC(160)			16 bits:TCMP & TCMP(P) - - - - - - - - - - - - - - - - - 5 steps
	TCMP	P	

		J1n	J2n--

Operands: [S1.][S2.][S3.]

K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

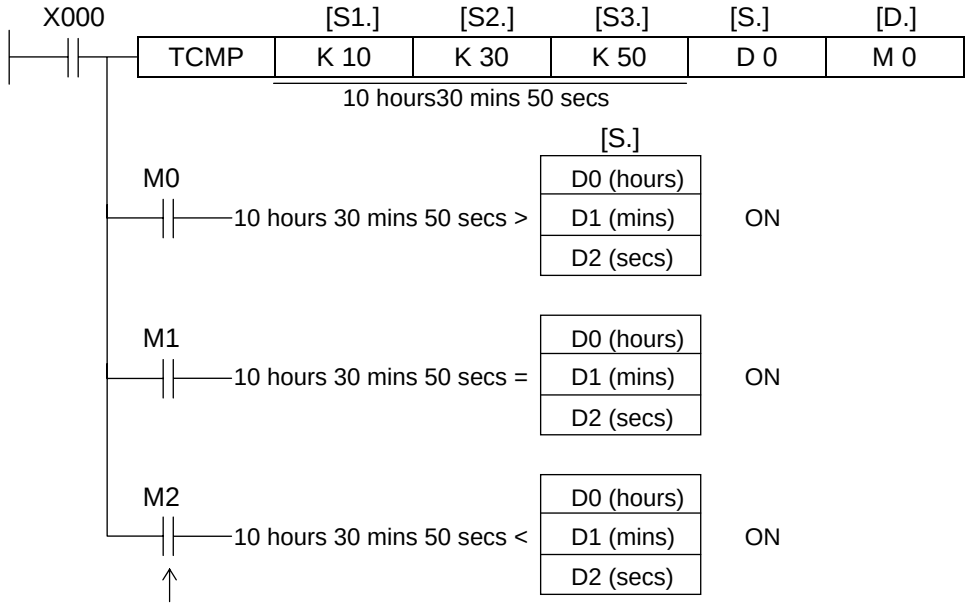
[S.]

Operands: [D.]

X	Y	M	S

[D.] occupy continuative 3 bits

Flag: M8020, M8021, M8022



When X000 OFF, not execute TCMP, M0~M2 status unchanged.

♦ Time of source device「[S1.],[S2.],[S3.]」compare with time value which stored at 3 bits from the head address of [S.]. According the result, the device of 3 bits from the head address of [D.] will be ON/OFF automatically.

[S1.] **:** "hour" assign「0~23」hour.
[S2.] **:** "min" assign「0~59」min.
[S3.] **:** "sec" assign「0~59」sec.

[S.] **:** "hour" assign「0~23」hour.
[S.] + 1 **:** "min" assign「0~59」min.
[S.] + 2 **:** "sec" assign「0~59」sec.

[D.] , [D.] +1 , [D.] +2 **:** according the result, device of 3 bits from the head address of [D.] is ON/OFF automatically.

♦ The content of real time clock stored at special register D8015(hour), D8014(min), D8013(sec).

Time Zone Compare

FNC(161)			16 bits:TZCP & TZCP(P) - - - - - - - - - - - - - - - - - - 9 steps
	TZCP	P	

		J1n	J2n--

Operands: |← [S1.][S2.][S3.] →|

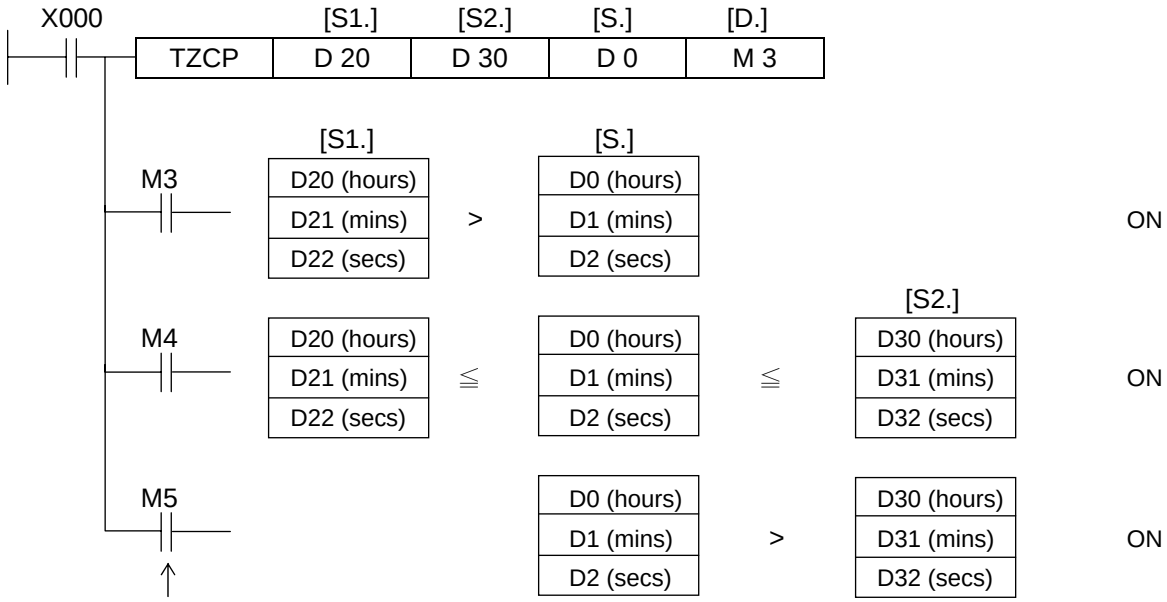
K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands: |← [D.] →|

X	Y	M	S

Occupy 3 bits from the head address of [D.], set [S1.]≦[S2.]

Flag: M8020, M8021, M8022



When X000 OFF, then not execute TZCP, M3~M5 status unchanged.

♦ Compare it with time value zone of 3 bits from the head address of [S.]. According to the result, then 3 bits from the head address of [D.] will be ON/OFF automatically.

[S1.], [S.] +1, [S.] +2 **:** The lower limit of compare range, assign “hour” , “min” , “sec”.
[S2.], [S2.] +1, [S2.] +2 **:** The topper limit of compare range, assign “hour” , “min” , “sec”.

[S.], [S.] +1, [S.] +2 **:** real time clock, assign “hour” , “min” , “sec”.
[D.], [D.] +1, [D.] +2 **:** According result of comparison, device of 3 bits from the head address of [D.] is ON/OFF automatically.

Setting range of “hour” , “min” , “sec” compare with real time clock, reference to FNC160 (TCMP).

Time Addition

FNC(162)			16 bits: TADD & TADD(P) - - - - - - - - - - - - - - - - - 7 steps
	TADD	P	

		J1n	J2n--

Operands: |←——[S1.][S2.]——→|

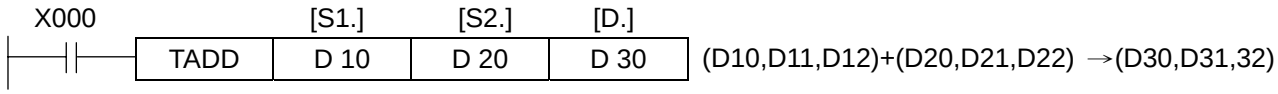
K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

|←—— [D.] ——→|

Operands:

X	Y	M	S

Flag: M8020, M8021, M8022



[S1.]		[S2.]		[D.]
D10 10(hours)		D20 3(hours)		D30 13(hours)
D11 30(mins)	+	D21 10(mins)	→	D31 40(mins)
D12 10(secs)		D22 5(secs)		D32 15(secs)
10hours30mins10secs		3hours10mins5secs		13hours40mins15secs

- The time value stored at 3 bits from the head address of [S1.] add the time value stored at 3 bits from the head address of [S2.], then stored the result at the device of 3 bits from the head address of [D.].
- If the result greater than “24”, carry flag M8022 ON, and the value of addend subtract 24, then stored at [D.].

[S1.]		[S2.]		[D.]
18 (hours)		10 (hours)		4 (hours)
10 (mins)	+	20 (mins)	→	30 (mins)
30 (secs)		5 (secs)		35 (secs)
18hours10mins10secs		10hours20mins5secs		4hours30mins35secs

- The result is “0” (0hour 0min 0sec), then zero flag M8020 ON.
- Setting range of “hour” , “min” , “sec” compare with real time clock, reference to FNC160 (TCMP) instruction.

Time Subtraction

FNC(163)			16 bits: TSUB & TSUB(P) - - - - - - - - - - - - - - - - - 7 steps
	TSUB	P	

		J1n	J2n--

Operands:

|←——[S1.][S2.]——→|

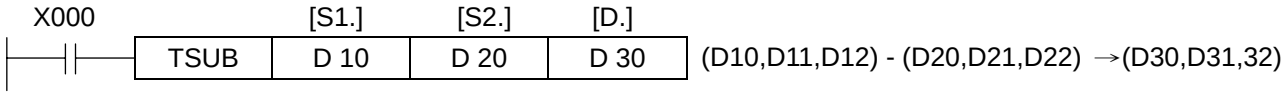
K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

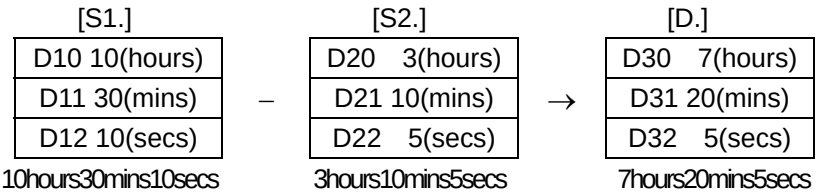
|←—— [D.] ——→|

Operands:

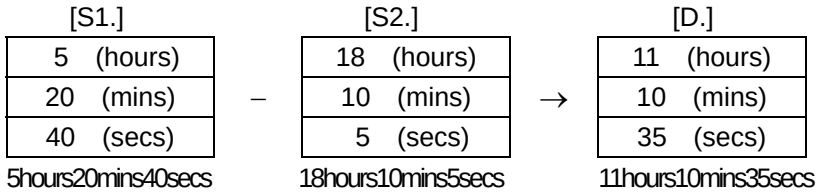
X	Y	M	S

Flag: M8020, M8021, M8022





- ♦ The time value stored at 3 bits from the head address of [S1.] subtract the time value stored at 3 bits from the head address of [S2.], then stored the result at the device of 3 bits from the head address of [D.].
- ♦ The result less than “0”, borrow flag ON, and the result of subtraction added 24, then stored at [D.]



- ♦ The result is “0” (0hour 0min 0sec), then zero flag M8020 ON.
- ♦ Setting range of “hour” , “min” , “sec” compare with real time clock, reference to FNC160(TCMP) instruction.

Time Read

FNC(166)			16 bits: TRD & TRD(P) - - - - - - - - - - - - - - - - - - - 5 steps
	TRD	P	

		J1n	J2n--

Operands:

K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

|← [D.] →|

Operands:

X	Y	M	S

Occupy 7 bits from the head address of [D.]

Flag:

The data of real time clock stored at destination device of 7 bits from the head address of [D.]

Device	Item	Data		Device	Item
D8018	Year	0~99(last two figure)	→	D0	Year
D8017	Month	1~12	→	D1	Month
D8016	Date	1~31	→	D2	Date
D8015	Hours	0~23	→	D3	Hours
D8014	Minutes	0~59	→	D4	Minutes
D8013	Seconds	0~59	→	D5	Seconds
D8019	Week	0(Sun)~6(Sat)	→	D6	Week

Time Write

FNC(167)			16 bits: TWR & TWR(P) - - - - - - - - - - - - - - - - - - 5 steps
	TWR	P	

		J1n	J2n--

Operands:

[S.]

K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands:

X	Y	M	S

Flag:

New time data write into special register D8013-D8019

Device	Item	Data		Device	Item
D10	Year	0~99(last two figure)	→	D8018	Year
D11	Month	1~12	→	D8017	Month
D12	Date	1~31	→	D8016	Date
D13	Hours	0~23	→	D8015	Hours
D14	Minutes	0~59	→	D8014	Minutes
D15	Seconds	0~59	→	D8013	Seconds
D16	Week	0(Sun)~6(Sat)	→	D8019	Week

GRAY CODE

FNC(170)			16 bits:GRY & GRY(P) - - - - - - - - - - - - - - - - - - - 5 steps
D	GRY	P	32 bits:(D)GRY & (D)GRY(P) - - - - - - - - - - - - - - -9 steps

		J1n	J2n--

Operands: [S.]

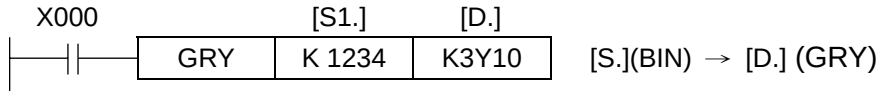
K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

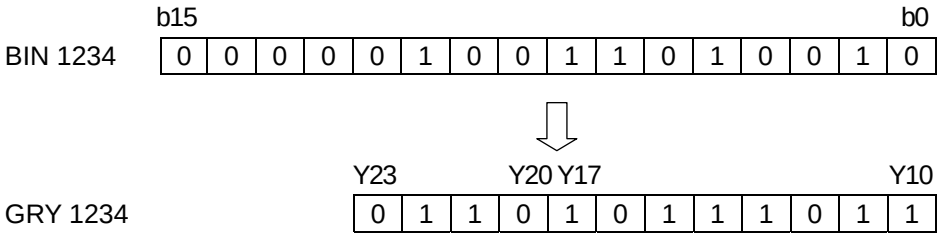
[D.]

Operands:

X	Y	M	S

Flag:





♦ [S.] effective value range

When 16 bits operation **:** 0~32,767

When 32 bits operation **:** 0~2,147,483,647

GRAY CODE

FNC(171)			16 bits:GBIN & GBIN(P) - - - - - - - - - - - - - - - - - - 5 steps
D	GBIN	P	32 bits:(D)GBIN & (D)GBIN(P) - - - - - - - - - - - - - -9 steps

		J1n	J2n--

Operands:

[S.]

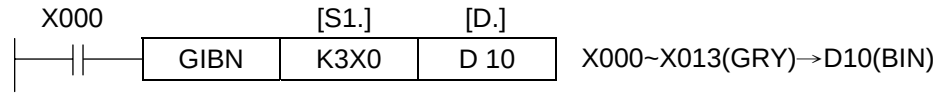
K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.]

Operands:

X	Y	M	S

Flag:



	X13			X10	X7							X0
GRY 1234	0	1	1	0	1	0	1	1	1	0	1	1

	b15															b0
BIN 1234	0	0	0	0	0	1	0	0	1	1	0	1	0	0	1	0

- ♦ When FNC51 (REFF) be used, need notice filter time (D8020-D8037) will response time.
- ♦ [S.] effective value range
  When 16 bits operation**:** 0~32,767
  When 32 bits operation**:** 0~2,147,483,647

LD ※ (LoaD compare)

FNC(224~230)			16 bits: - 5 steps
D	LD ※		32 bits: - 9 steps

		J1n	J2n--

※ : = , > , < , <> , ≦ , ≧

Operands:

[S1.][S2.]

K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

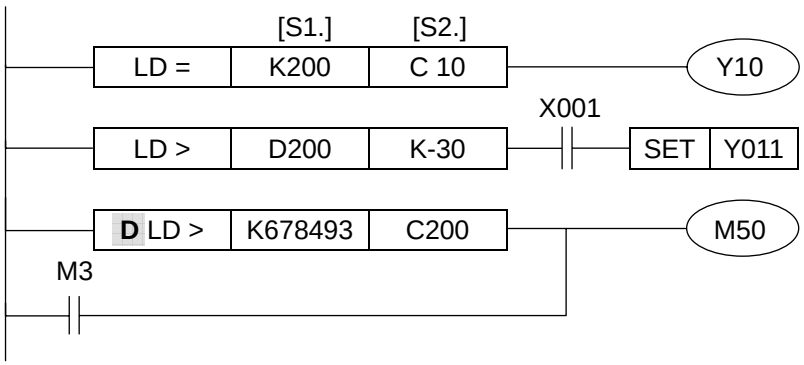
Operands:

X	Y	M	S

Flag:

♦ Comparison of BIN to the content of two source operands, according the result, update operate status

FNC No.	16 bits instruction	32 bits instruction	ON	OFF
224	LD =	**D** LD =	[S1.] = [S2.]	[S1.] ≠ [S2.]
225	LD >	**D** LD >	[S1.] > [S2.]	[S1.] ≦ [S2.]
226	LD <	**D** LD <	[S1.] < [S2.]	[S1.] ≧ [S2.]
228	LD < >	**D** LD < >	[S1.] ≠ [S2.]	[S1.] = [S2.]
229	LD ≦	**D** LD ≦	[S1.] ≦ [S2.]	[S1.] > [S2.]
230	LD ≧	**D** LD ≧	[S1.] ≧ [S2.]	[S1.] < [S2.]



♦ The upper bit of [S1.][S2.] is sign bit, i.e. 0: positive, 1: negative

♦ If use 32 bits counter (C200~) to compare, have to use 32 bits instruction.
If use 16 bits instruction to compare, then error will occur.

AND ※ (AND compare) AND=, AND>, AND<, AND<>, AND<=, AND>=

FNC(232~238)			16 bits: - 5 steps
D	AND ※		32 bits: - 9 steps

		J1n	J2n--

※ : = , > , < , <> , ≦ , ≧

Operands: [S1.][S2.]

K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

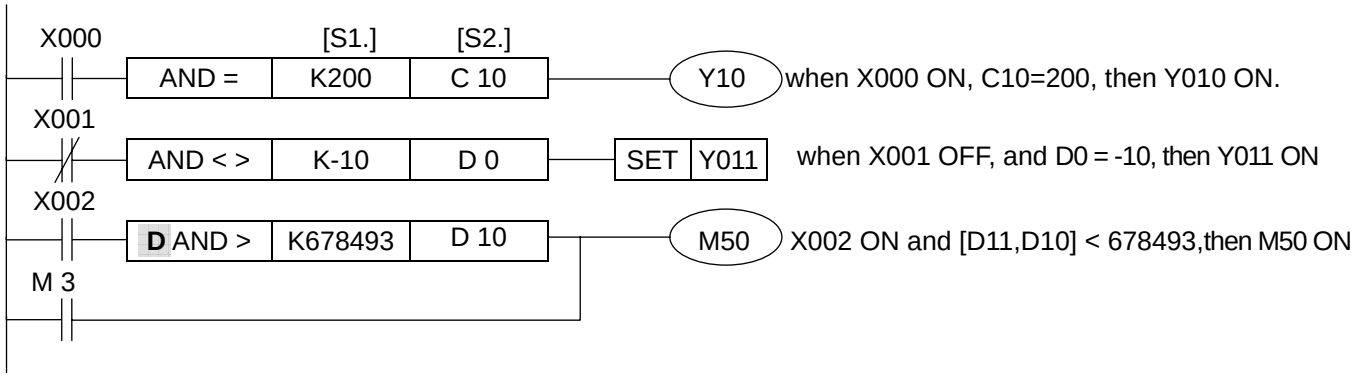
Operands:

X	Y	M	S

Flag:

♦ Comparison of BIN to the content of two source operands, according to the result, update operate status.

FNC No.	16 bits instruction	32 bits instruction	ON	OFF
232	AND =	**D** AND =	[S1.] = [S2.]	[S1.] ≠ [S2.]
233	AND >	**D** AND >	[S1.] > [S2.]	[S1.] ≦ [S2.]
234	AND <	**D** AND <	[S1.] < [S2.]	[S1.] ≧ [S2.]
236	AND < >	**D** AND < >	[S1.] ≠ [S2.]	[S1.] = [S2.]
237	AND ≦	**D** AND ≦	[S1.] ≦ [S2.]	[S1.] > [S2.]
238	AND ≧	**D** AND ≧	[S1.] ≧ [S2.]	[S1.] < [S2.]



♦ The upper bit of [S1.][S2.] is sign bit, i.e. 0: positive, 1: negative

♦ Use 32 bits counter (C200~) to compare, have to use 32 bits instruction.
If use 16 bits instruction to compare, then error will occur.

OR ※ (OR compare) OR=, OR>, OR<, OR<>, OR<=, OR>=

FNC(240~246)			16 bits: - 5 steps
D	OR ※		32 bits: - 9 steps

		J1n	J2n--

※ : = , > , < , <> , ≦ , ≧

Operands: [S1.][S2.]

K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

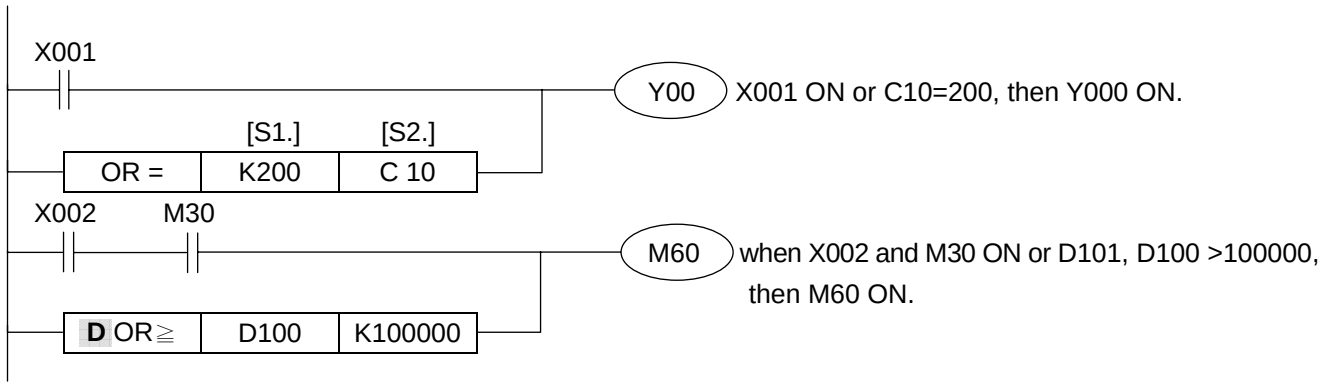
Operands:

X	Y	M	S

Flag:

♦Comparison of BIN to the content of two source operands, according the result, update operate status.

FNC No.	16 bits instruction	32 bits instruction	ON	OFF
240	OR =	**D** OR =	[S1.] = [S2.]	[S1.] ≠ [S2.]
241	OR >	**D** OR >	[S1.] > [S2.]	[S1.] ≦ [S2.]
242	OR <	**D** OR <	[S1.] < [S2.]	[S1.] ≧ [S2.]
244	OR < >	**D** OR < >	[S1.] ≠ [S2.]	[S1.] = [S2.]
245	OR ≦	**D** OR ≦	[S1.] ≦ [S2.]	[S1.] > [S2.]
246	OR ≧	**D** OR ≧	[S1.] ≧ [S2.]	[S1.] < [S2.]



♦ The upper bit of [S1.][S2.] is sign but, i.e. 0:positive, 1:negative

♦ When use 32 bits counter (C200~) to compare, then have to use 32 bits instruction. If use 16 bits instruction to compare, then error will occur.