04 Program flow instructions

last modified by Stone Stone

on 2022/06/15 11:27

Table of Contents

Program jump

CJ/Conditional jump

When the jump instruction is ON, the program with the specified pointer number in the same program file is executed.

-[CJ (P) (P)]

Content, range and data type

Parameter	Content	Range	Data type	Data type (label)
(P)	The pointer number of the jump target	P0 to P4095	Device name	POINTER

Device used

Instruction	Parameter	Offset modification	Pulse extension	other
		[D]	XXP	P
CJ	Parameter 1		●	●

Features

- CJ(P)

When the execution instruction is ON, the program with the specified pointer number is executed.

When the execution instruction is OFF, execute the next program.

1. Execute instructions.
2. Each scan is executed.
3. One scan is executed.

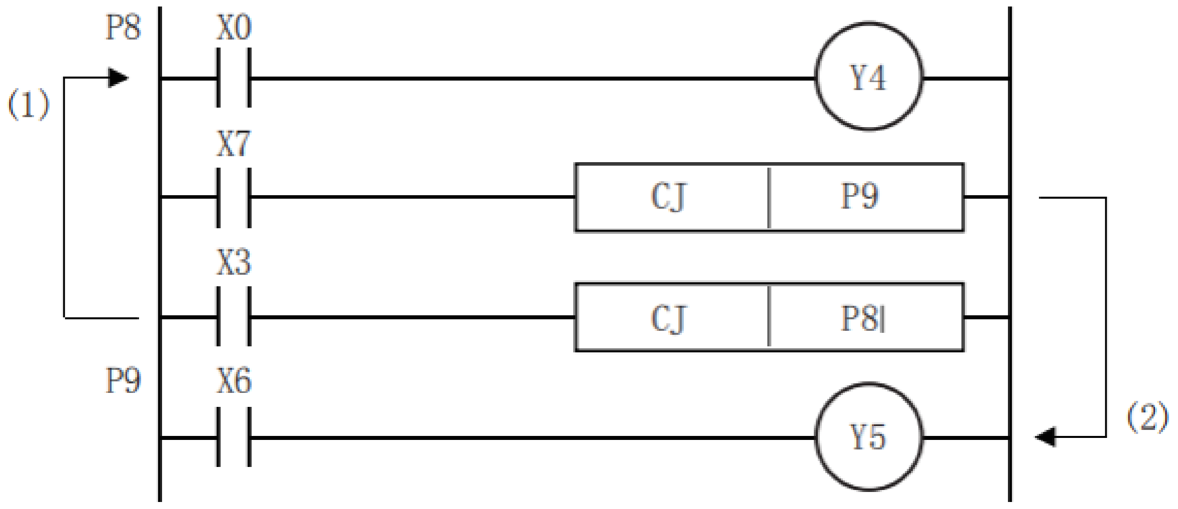
#Note:

After turning ON the coil of the timer, if the timer whose coil is ON is jumped by the CJ(P) instruction, the measurement will not be performed normally.

When the OUT instruction is jumped by the CJ(P) instruction, the scan time will be shorter.

When the CJ(P) instruction is used to jump backward, the scan time will be longer.

For the CJ(P) instruction, you can jump to a step smaller than the step number being executed. However, in order to avoid the time limit of the watchdog timer, a method of jumping out of the loop during this period should be considered.



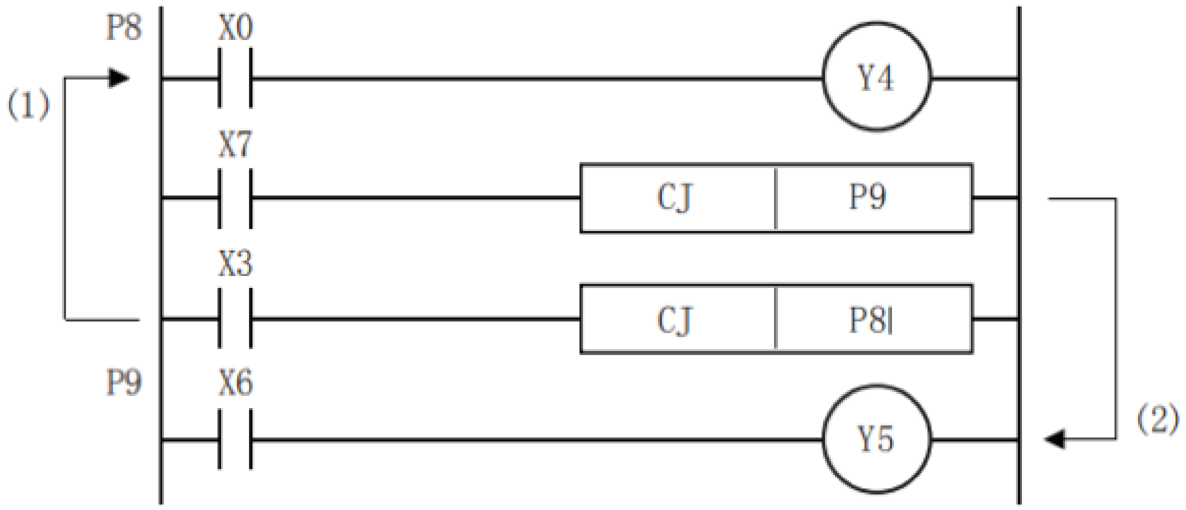
(1) While X3 is ON, the loop is executed.

(2) When X7 is set to ON, it jumps out of the loop.

• The device skipped by the CJ(P) instruction does not change.

When X2 is ON, jump to the label of P19.

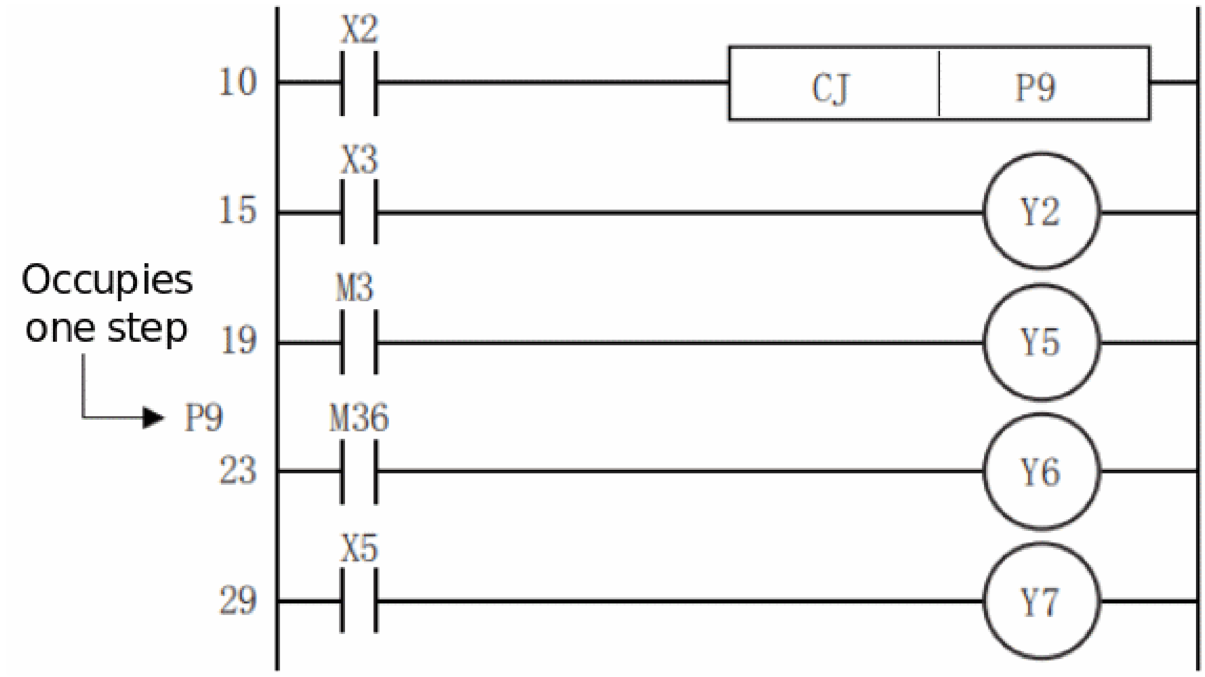
Even if X2 and X4 turn ON/OFF during CJ instruction execution, Y4 and Y5 will not change.



(1) When X2 is ON, jump to the label of P19.

(2) Even if X2 and X4 turn ON/OFF during CJ instruction execution, Y4 and Y5 will not change.

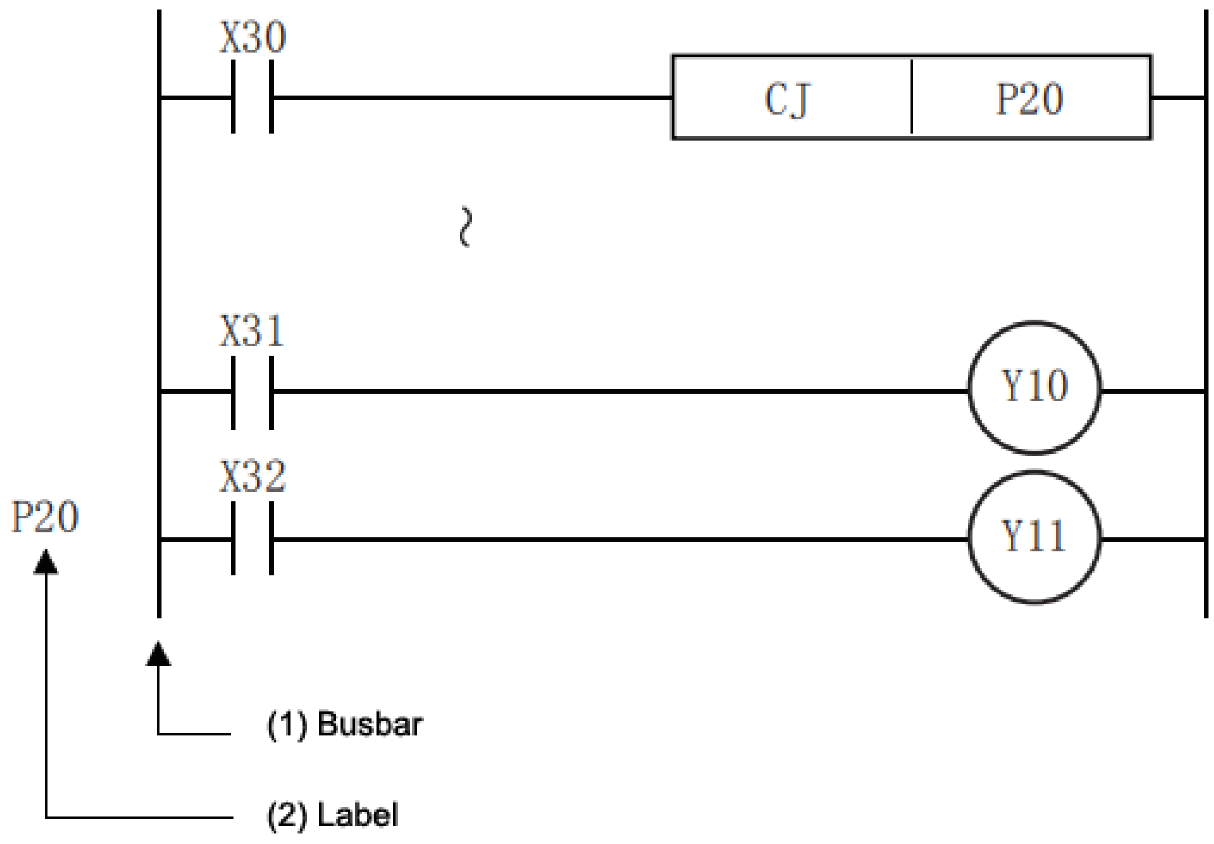
• The label (P□) occupies 1 step.



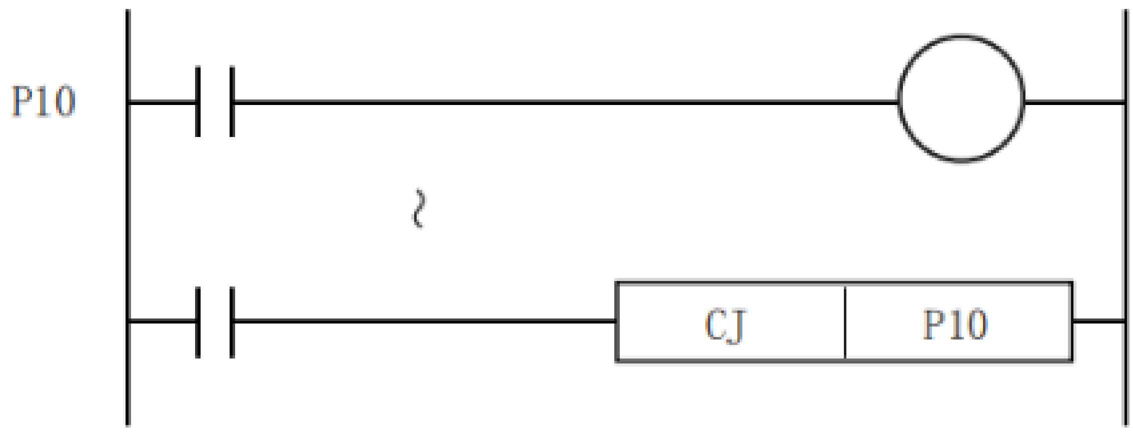
The jump instruction can only specify the pointer number in the same program file.

When jumping to the pointer number within the jump range during jump operation, the program after the jump destination pointer number is executed.

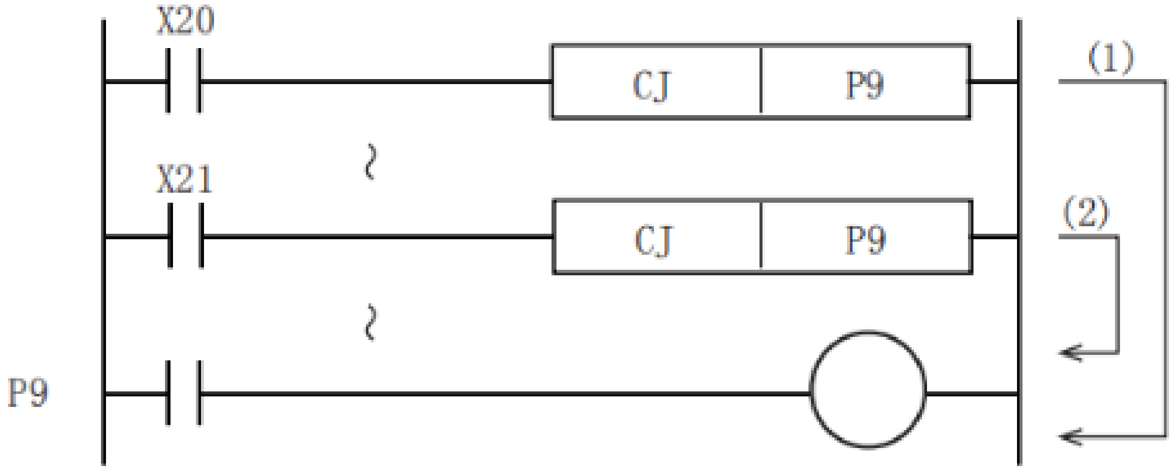
The label procedure is shown below. When creating a loop program, move the cursor to the left of the bus bar of the Circuit program, and enter the label (P) at the beginning of the loop block.



It is also possible to program the label at the position where the step number is less than the CJ instruction, but if the scan time becomes more than 200ms (default setting), a watchdog timer error will occur, which requires attention.



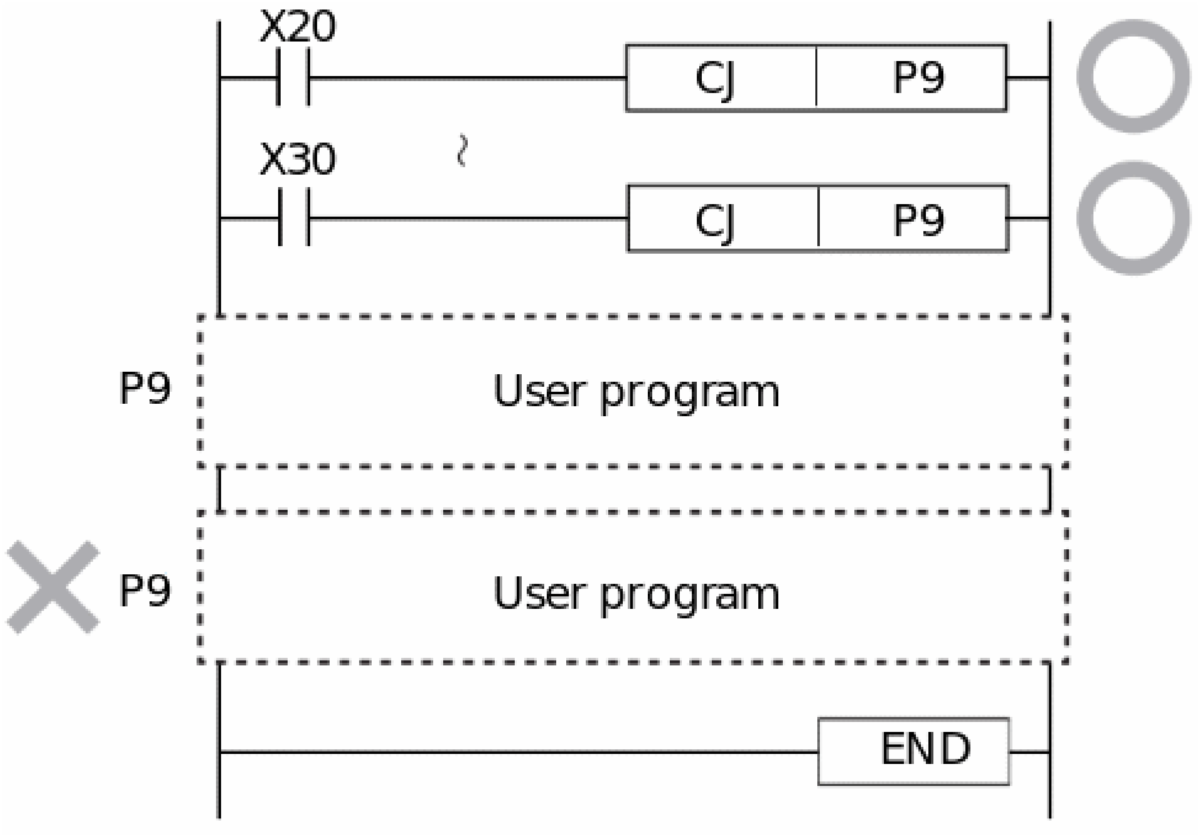
When the pointer number in the operand is the same and the label is one, the operation is as follows.



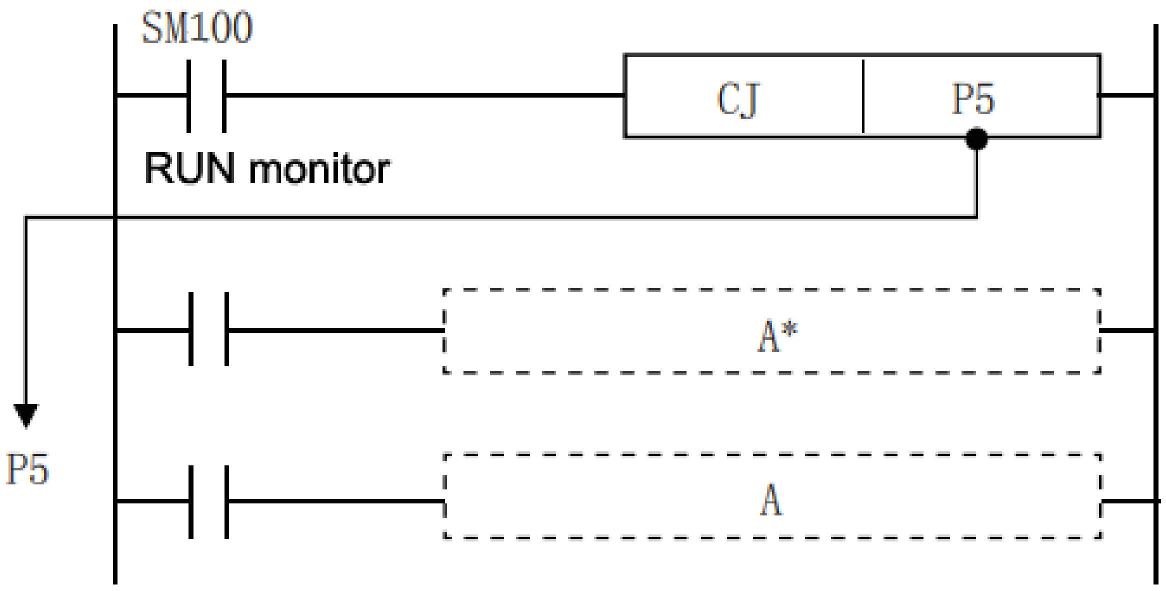
(1) When X20 is ON, jump from the CJ instruction of X20 to label P9.

(2) When X20 is OFF and X21 is ON, jump from the CJ instruction of X21 to label P9.

If the tag number is reused, it will become an error state.



SM100 is always ON during the operation of the CPU module, so the usage method shown below will jump unconditionally.



A*: User program (jump without operation)

A: User program

The pointer number P63 of LX3V represents the jump to the END instruction. The P63 pointer of LX5V no longer provides this function. If you need to use this function, please use the GOEND instruction.

Error code

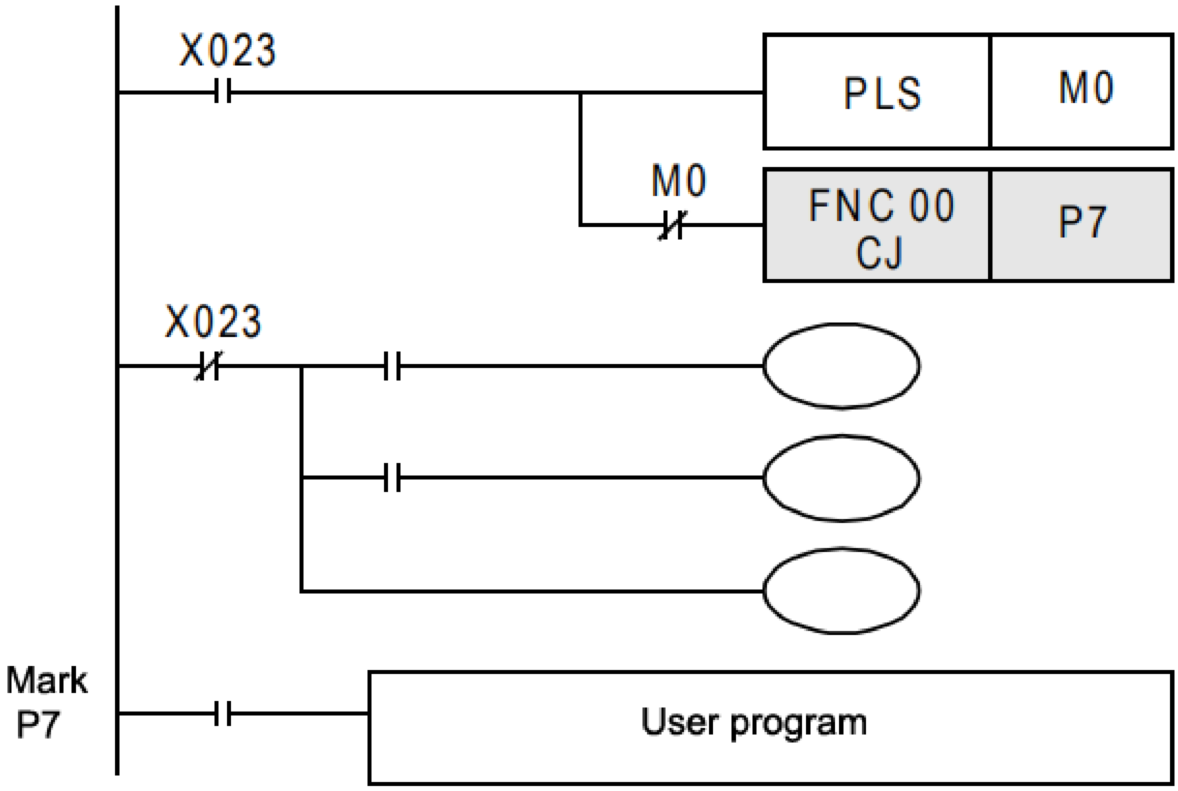
No error message

Example

(1) The situation to jump after OFF processing

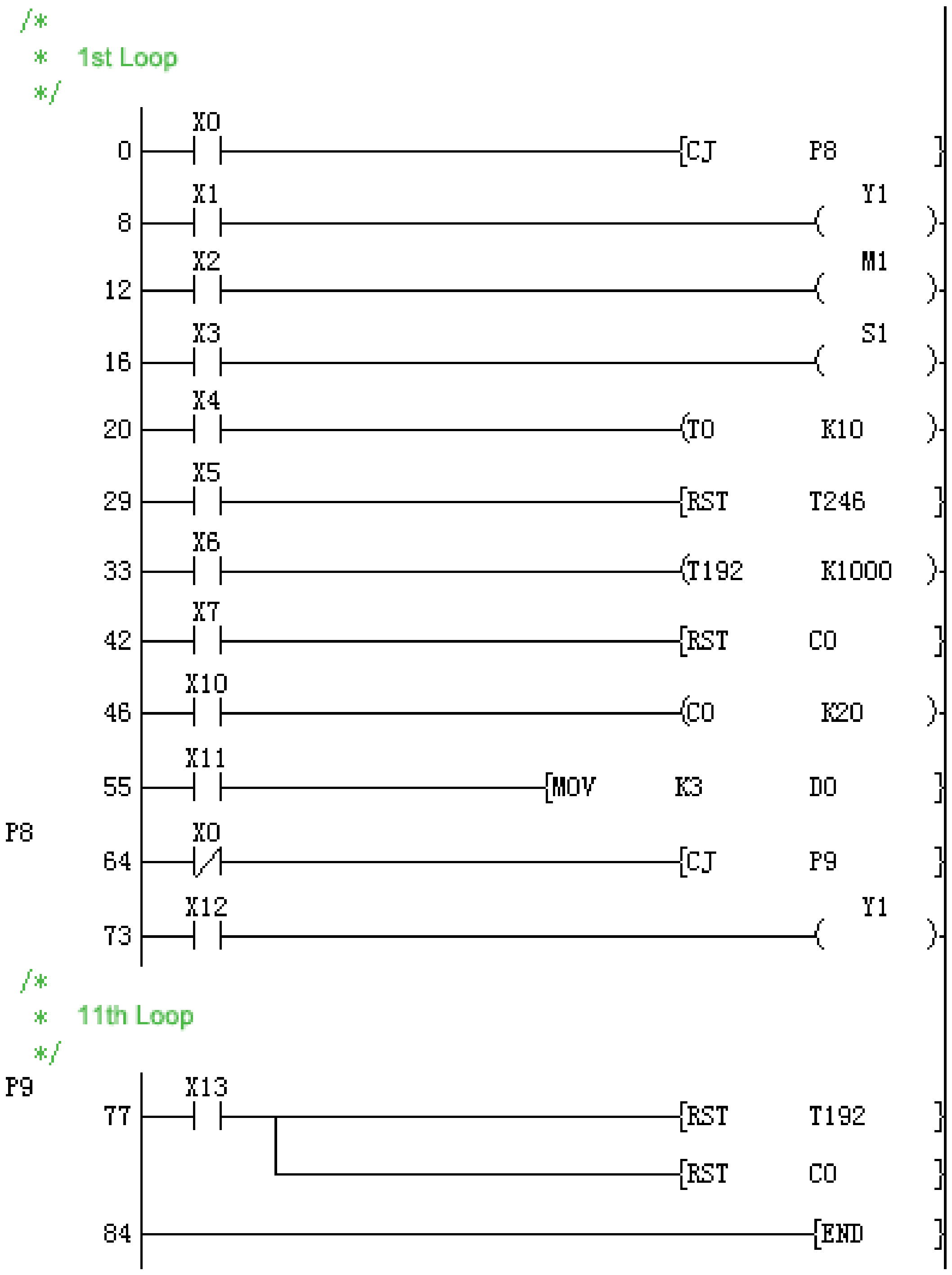
After one operation cycle when X023 changes from OFF to ON, the CJ P7 instruction is valid.

With this method, the output between CJ P7 instruction and mark P7 can be turned off before jumping.



(2) CJ instruction and action of contact coil

In the following program example, when X000 is ON, jump from the CJ instruction of the first loop to the mark P8. When X000 is OFF, no jump is performed, but the program is executed in order from step 1, and the CJ instruction in the 11th loop jumps to mark P9. The jumped instruction is not executed.



Double-coil action of Y001 output:

When X000=OFF, it will act through X001.

When X000=ON, it will act through X012.

Even if the program is distinguished by conditional jump, if the same coil (Y000) is programmed twice or more within or outside the jump, it will be treated as normal double coil processing.

The action of the subroutine timer (T192 to T199):

After the coil is driven, the action continues even if it jumps, and the output contact also operates.

If using the high-speed counter (HSC0 to HSC7) operation

After the coil is driven, the action continues even if it jumps, and the output contact also operates.

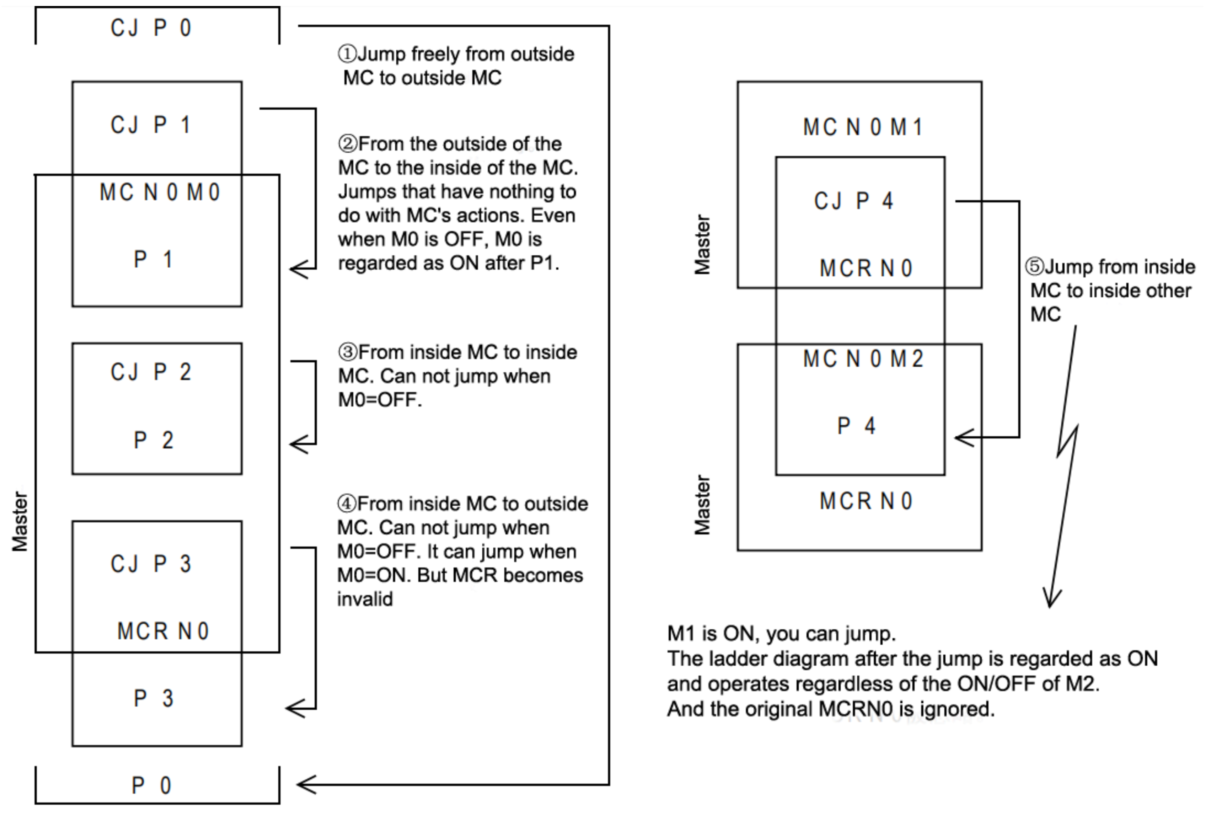
In the above program, if each input changes during the jump, the action of each coil is shown in the following table.

Content	**Contact state before jump**
Y,M,S (Y1, M1, S1)	X1, X2, X3 OFF
	X1, X2, X3 ON
1ms, 10ms, 100ms timer (T0)	X4 OFF
	X4 ON
Program timer (T192)	X5 OFF, X6 OFF
	X5 OFF, X6 ON
Counter (C0)	X7 OFF, X10 OFF
	X7 OFF, X10 ON
Application instructions	X11 OFF
(MOV)	X11 ON

(3) The relationship between CJ instruction and MC to MCR jump

The relationship between the main control instruction and the jump instruction and the action content are as follows.

However, since the operation of ②, ④, and ⑤ will become complicated, please avoid using them.


CJ P 0
①Jump freely from outside MC to outside MC
CJ P 1
②From the outside of the MC to the inside of the MC. Jumps that have nothing to do with MC's actions. Even when M0 is OFF, M0 is regarded as ON after P1.
MC N 0 M 0
P 1
CJ P 2
P 2
③From inside MC to inside MC. Can not jump when M0=OFF.
Master
CJ P 3
④From inside MC to outside MC. Can not jump when M0=OFF. It can jump when M0=ON. But MCR becomes invalid
MCR N 0
P 3
P 0
Master
MC N 0 M 1
CJ P 4
MCR N 0
⑤Jump from inside MC to inside other MC
Master
MC N 0 M 2
P 4
MCR N 0
M1 is ON, you can jump.
The ladder diagram after the jump is regarded as ON and operates regardless of the ON/OFF of M2.
And the original MCRN0 is ignored.

Subroutine jump

CALL/Subroutine call

When the jump instruction is ON, the program with the specified pointer number in the same program file is executed.

-[CALL (P) (P)]

Content, range and data type

Parameter	Content	Range	Data type	Data type (label)
(P)	Subroutine name	-	Pointer	POINTER

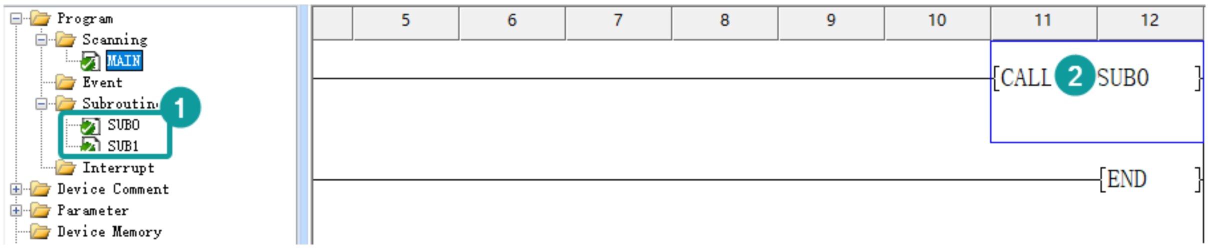
Device used

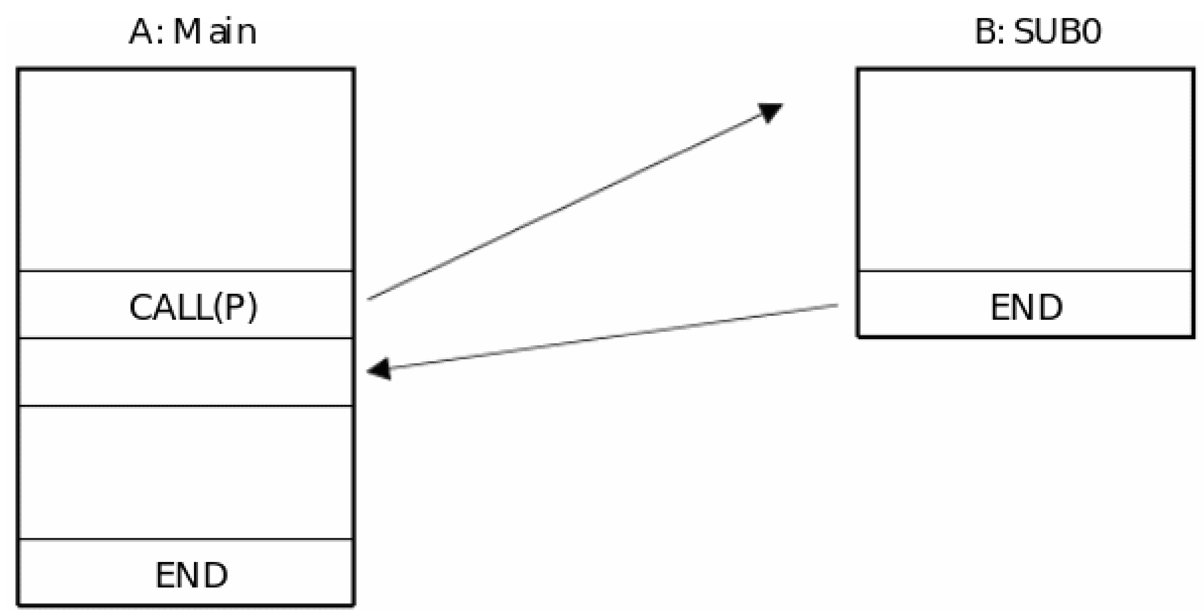
Instruction	Parameter	Offset modification	Pulse extension	other
			[D]XXP	Subroutine name
CALL	Parameter 1	●		●

Parameter 1 can only use the subroutine name.

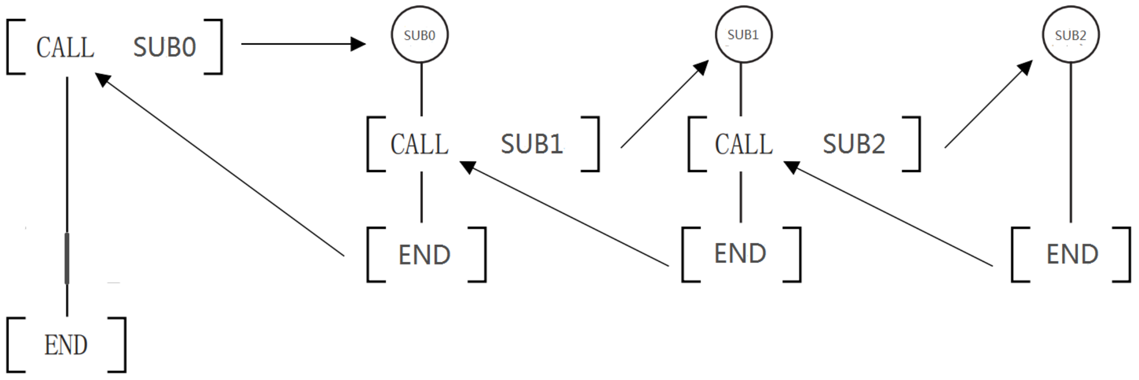
Features

When the CALL(P) instruction is executed, the subroutine of the pointer (P) will be executed. (P) can only write the name of the newly created subprogram, if the program name does not exist, the Circuit program compilation fails.


```
Program
  Scanning
    MAIN
  Event
  Subroutine
    SUB0
    SUB1
  Interrupt
Device Comment
Parameter
Device Memory
```



CALL(P) instructions can be nested up to 32 levels.



#Note:

• Multiple CALL(P) instructions can call the same subprogram, but subprograms with the same program name are not allowed.

• Use program timers in subroutines (the same applies to interrupt programs). This timer counts when the coil instruction or the END instruction is executed. If it reaches the timer setting value, the output contact will act when the coil instruction or END instruction is executed. Generally, the timer only counts when the coil instruction is executed, so if it is used in a subroutine that executes the coil instruction under certain conditions, it will not count.

• If the 1ms accumulative timer is used in a subroutine (the same in an interrupt program), when it reaches the set value, the output contact will act when the first coil instruction is executed (when the subroutine is executed), so be careful.

• The devices that are turned on in the subprogram (the same in the interrupt program) will be retained after the program ends. Therefore, these devices should be reset in the main program after the end of the program.

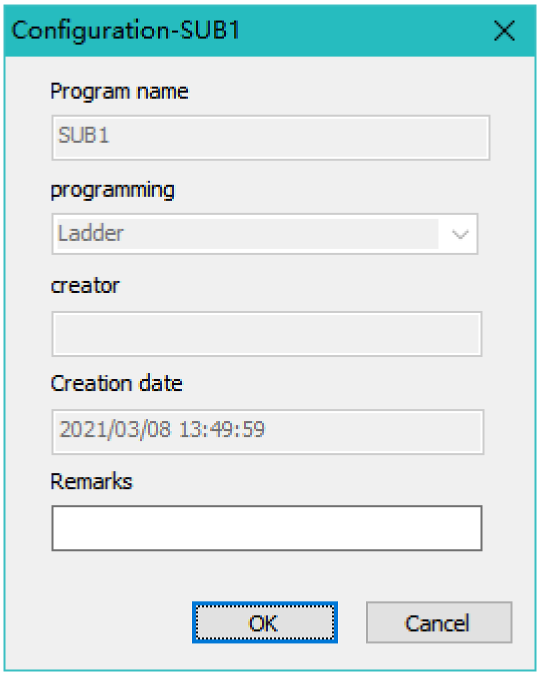
Error code

Error code	Content
4102H	CALL(P) instruction exceeds 32 levels of nesting structure

Example

(1) New subroutine

Project management→Subroutine→ Scan→Right click to create


Configuration-SUB1

Program name: SUB1

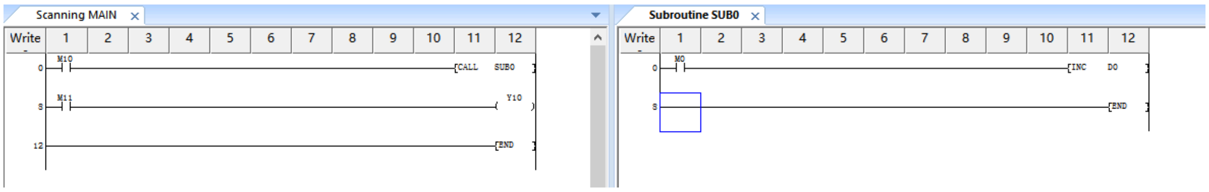
programming: Ladder

creator:

Creation date: 2021/03/08 13:49:59

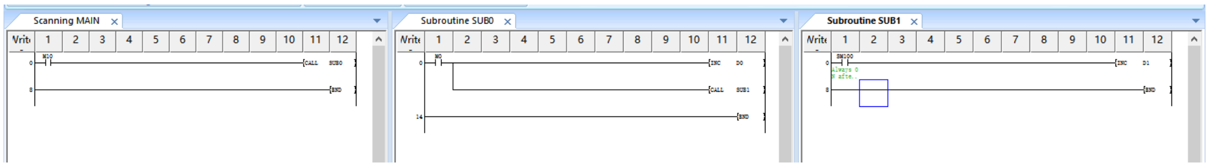
Remarks:

(2) Subroutine call



In the scan program, turn on M10 to call the subroutine SUB0, execute the Circuit program in the subroutine SUB0, until the END instruction of the subroutine is executed, return to the scan program MAIN to execute LD M11.

(3) Subroutine nesting



In the above figure, the subroutine SUB0 is called in the scan program, and the subroutine SUB1 is called in SUB0. So when the scan program M10 is turned on, after the CALL instruction is executed, the subroutine SUB0 will be executed first.And after the CALL instruction of SUB0 is executed, SUB1 will be executed first. After executing the END instruction of SUB1, return to SUB0 for execution. After executing the END instruction of SUB0, return to the scan program MAIN. The program has only 2 levels of nesting, and the number of nesting levels cannot be greater than 32.

Interrupt disable, interrupt enable

DI and EI/Interrupt prohibited and allowed

The CPU module is usually interrupt disabled. This instruction can make the CPU module into the interrupt enabled state (EI instruction), and then become disabled again (DI instruction).

• DI: It is forbidden to interrupt program execution.

• EI: Release the interrupt prohibition state.

-[DI (s)]

-[EI]

Content, range and data type

Parameter	Content	Range	Data type	Data type (label)
(P)	Subroutine name	-	Pointer	POINTER

Instruction	Parameter	Offset modification	Pulse extension
		[D]	XXP
DI	Parameter 1		●

Features

- DI

• Even if the execution interrupt condition is triggered in the program, prohibit the interrupt program execution before executing the EI instruction.

• When the PLC is powered on or after STOP, it will become the state after DI instruction is executed, and the interrupt program cannot be executed.

• The DI instruction can choose whether to use parameters. When there is no parameter, it means that all interrupt programs are prohibited. With parameters, according to the value in parameter s1, interrupt programs with this priority and lower priority are prohibited.

• The priority of the interrupt ranges from 0 to 2. The smaller the value, the higher the response priority of the interrupt. That is, the interrupt with priority 0 is the fastest to be responded.

• If there is no EI instruction before the DI instruction, the DI instruction is invalid.

- EI

• Release the interrupt prohibition state when DI instruction is executed, and allow interrupt program to run.

• When the EI and DI instructions are not enabled, they all maintain the original enabled or forbidden interrupt program execution status. The currently disabled interrupt priority can be viewed in SD151.

SD151	Currently disabled interrupt priority	According to the interrupt prohibition instruction (DI instruction), enable instruction (EI instruction), the priority of the interrupt prol 0: All priority interrupts are disabled (default); 1: Priority 1 and 2 interrupts are prohibited; 2: Priority 2 interrupt is prohibited; 3: All priority interrupts are allowed

A: Sequence control program

- DI, EI nested structure

(1) Interrupt allowable intervals of all priority levels;

(2) Interrupt forbidden zone below priority 2 (interrupt allowable zone above priority 1);

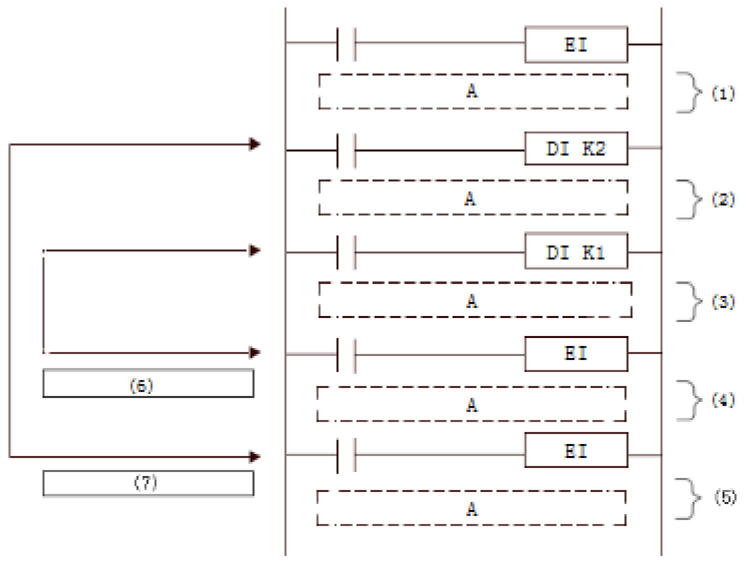
(3) Interrupt forbidden interval below priority 1 (interrupt allowable interval above priority 0);

(4) Interrupt prohibition zone below priority 2 (interrupt enable zone above priority 1);

(5) Interrupt allowable intervals of all priority levels;

(6) EI paired with [DI K1];

(7) EI paired with [DI K2].



• Interrupts (requests) that occur after the DI instruction are processed after the EI instruction is executed.

• When the DI instruction is executed multiple times and the priority of the argument is specified to be higher than the priority currently being prohibited, interrupts below the priority of the argument are disabled.

• When the DI instruction is executed multiple times and the priority of the argument is specified to be lower than the priority currently being disabled, the interrupt disable status will not be changed.

• The nesting of DI instructions can be up to 16 levels.

• The interrupt priority of the interrupt pointer can be set by the properties of the interrupt program. Refer to the description of the interrupt program for details.

• The interrupt prohibition interval when DI instruction and EI instruction are executed is as follows.

(1) When the DI instruction is executed multiple times (when the interrupt with priority higher than the currently prohibited interrupt priority is prohibited and specified)

Scan execution type program

1. Interrupt allowable intervals of all priority levels;
2. Interrupt prohibition interval below priority 2 (interrupt allowable interval above priority 1);
3. Interrupt prohibition section below priority 1 (interrupt enable section above priority 0).

(2) When the DI instruction is executed multiple times (when the interrupt priority is lower than the currently prohibited interrupt priority is prohibited and specified)

Scan execution type program

1. Interrupt allowable intervals of all priority levels;
2. Interrupt prohibited interval below priority 1 (interrupt allowable interval above priority 0);
3. The interrupts below priority 1 are already in the disabled state, so the interrupt disable priority will not be changed.

(3) When DI instruction is executed through interrupt program

A: Scan execution type program

B: interrupt program

1. Interrupt allowable intervals of all priority levels;
2. Interrupt prohibited interval below priority 3 (interrupt allowable interval above priority 1);
3. Interrupt prohibition section below priority 2 (interrupt enable section above priority 0).

(4) When only DI instructions without arguments are executed

A: Scan execution type program

1. Interrupt allowable intervals of all priority levels;
2. Interrupt prohibition interval below priority 1 (all interrupt prohibition intervals);
3. Because the DI instruction with no argument is set to interrupt prohibition, by executing the EI instruction once, all priority interrupts are set to allow.

(5) In the case of executing DI instructions with arguments and DI instructions without arguments (when executing in the order of DI instructions with arguments → DI instructions without arguments)

A: Scan execution type program

1. Interrupt allowable intervals of all priority levels;
2. Interrupt prohibition interval below priority 2 (interrupt allowable interval above priority 1);
3. Interrupt prohibition section below priority 1 (all interrupt prohibition sections).

(6) In the case of executing DI instructions with arguments and DI instructions without arguments (in the case of execution in the order of DI instructions with no arguments → DI instructions with arguments)

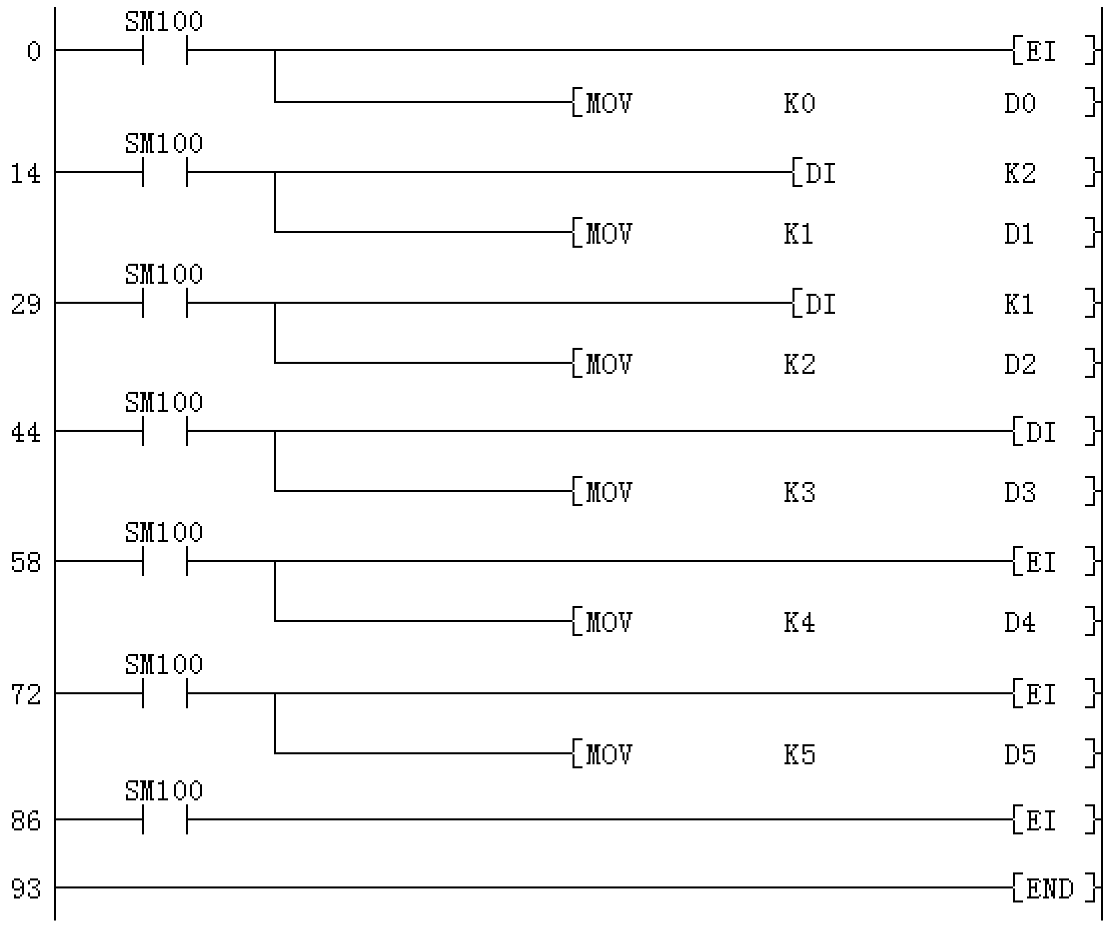
A: Scan execution type program

1. Interrupt allowable intervals of all priority levels;
2. Interrupt prohibition section below priority 1 (all interrupt prohibition sections).

Error code

Error code	Content
4085H	(S) read address exceeds the device range
4084H	The data set in (S) exceeds 0 to 2
4185H	When the nesting of DI instructions exceeds 16 levels

Example



All interrupt programs can be triggered

Can trigger interrupt programs of priority 0 and priority 1

Can trigger interrupts with priority 0

Cannot trigger any interrupts

Can trigger an interrupt program with a priority of 0

Can trigger interrupt programs with priority 0 and priority 1

SIMASK/Interrupt mask

Set interrupt pointer No. specified in (I) to the execution permission state/execution prohibition state according to the value of (s).

-[SIMASK (I) (s)]

Content, range and data type

Parameter	Content	Range	Data type	Data type (label)
(I)	Interrupt program name	-	Program name	POINTER
(s)	Specify the enable/ disable of interrupt	0: Allow. 1: Prohibited	Signed BIN 16 bit	ANY16

Device used

Instruction	Parameter	Devices	Offset	Pulse

														modification	extension
		KnX	**KnY**	**KnM**	**KnS**	**T**	**C**	**D**	**R**	**SD**	**K**	**H**	**E**	**[D]**	**XXP**
SIMASK	Parameter 1	Only support interrupt program name													
	Parameter 2	●	●	●	●	●	●	●	●	●	●	●	●	●	

Features

• The interrupt program of the interrupt program name specified in (I) is set to the execution permission state/ execution prohibited state according to the data specified in (s).

• When (s) is 0: Interrupt program execution permission status

• When (s) is 1, the execution of the interrupt program is prohibited

• Regarding the interrupt program when the power is turned on or after STOP→RUN, all interrupt programs will be executed.

• After setting interrupt prohibition, the prohibition state will be saved even if the instruction is disconnected. To restore it, write 0 to (S), turn on the instruction again, or execute STOP→RUN.

• The interrupted execution permission status/execution prohibition status will be stored in SM or SD, details as following:

(1) External interrupt

Register	Content	Register	Content	Register	Content	Register	Content
SM352	X0 rising edge interrupt	SM356	X2 rising edge interrupt	SM360	X4 rising edge interrupt	SM364	X6 rising edge interrupt
SM353	X0 falling edge interrupt	SM357	X2 falling edge interrupt	SM361	X4 falling edge interrupt	SM365	X6 falling edge interrupt
SM354	X1 rising edge interrupt	SM358	X3 rising edge interrupt	SM362	X5 rising edge interrupt	SM366	X7 rising edge interrupt
SM355	X1 falling edge interrupt	SM359	X3 falling edge interrupt	SM363	X5 falling edge interrupt	SM367	X7 falling edge interrupt

(2) Timer interrupt

Register	Content
SD350 to SD356	Timer interrupt mask, each bit represents an interrupt, a total of 100

(3) High-speed counter interrupt

Register	Content
SD382 to SD388	high-speed counter interrupt mask, each bit represents an interrupt, a total of 100

Error code

Error code	Content
4084H	Data beyond 0 and 1 is input in the application instruction(s)
4085H	(S) in the read application instruction exceeds the device range
4189H	The SIMASK instruction specifies an interrupt program name that is not set

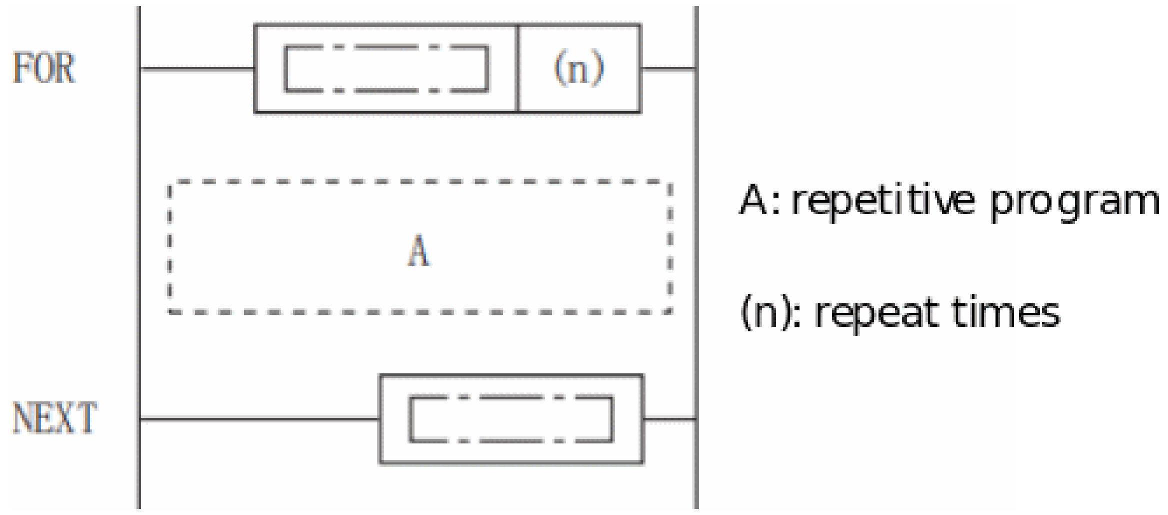
Example

As shown in the figure: when M10 is turned on, the three interrupt programs of INT10, INT91 and INT70 are prohibited from running.

Cycle instructions

FOR to NEXT/Cycle

When the processing between the FOR to NEXT instruction is executed unconditionally times, the next processing of the NEXT instruction will be performed.



Content, range and data type

Parameter	Content	Range	Data type	Data type (label)
	Number of repetitions between FOR to NEXT instructions	1 to 32767	Signed BIN 16 bit	ANY16

Instruction	Parameter	Devices												Offset modification	Pulse extension
		KnX	KnY	KnM	KnS	T	C	D	R	SD	K	H	E	[D]	XXP
FOR	Parameter 1	●	●	●	●	●	●	●	●	●	●	●	●	●	

Features

• When the processing between the FOR to NEXT instruction is executed unconditionally times, the next processing of the NEXT instruction will be performed.

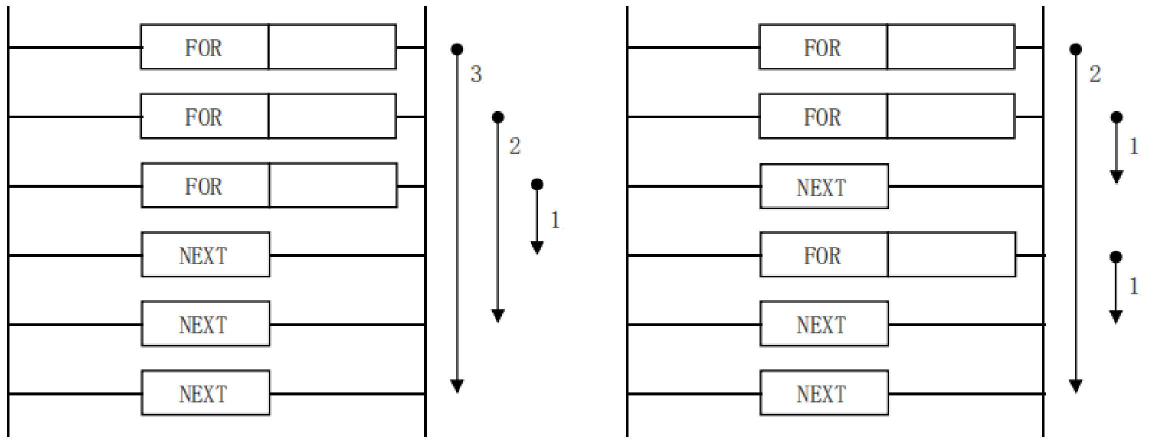
• can be specified in the range of 1 to 32767. When specifying -32768 to 0, the same processing as =1 will be performed.

• If you do not want to execute the processing between the FOR and NEXT instructions, use the CJ instruction to jump.

• The FOR instruction can be nested up to 5 levels.

#Note:

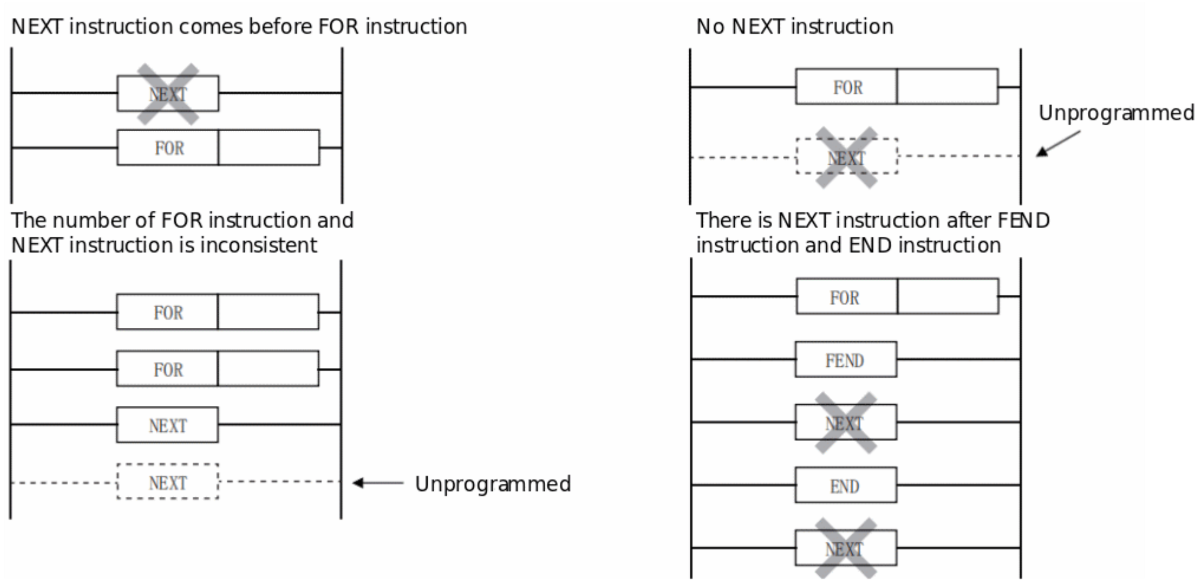
• In the case of FOR to NEXT instruction programming with nesting between FOR to NEXT instructions, up to 5 levels can be achieved.



• Do not use IRET, SRET, RET, FEND, END and other instructions to block between FOR to NEXT instructions.

• If the number of repetitions is too large, the cycle time (operation cycle) becomes longer and the watchdog timer error occurs, you need to change the watchdog timer time or reset the watchdog timer.

• The following program will become an error.


NEXT instruction comes before FOR instruction
NEXT
FOR
No NEXT instruction
FOR
NEXT
Unprogrammed
The number of FOR instruction and NEXT instruction is inconsistent
FOR
FOR
NEXT
NEXT
Unprogrammed
There is NEXT instruction after FEND instruction and END instruction
FOR
FEND
NEXT
END
NEXT

• If the FOR to NEXT instruction is repeatedly executed and ends midway, use the BREAK instruction.

Error code

Error code	Content
4085H	(s) read address exceeds the device range
4100H	When the nesting of FOR to NEXT instructions exceeds 5 levels or the number of FOR to NEXT does not correspond

Example

The program INC D0 will be executed 10 times, and INC D1 will be executed 100 times.

After execution, D0 will be equal to 10 and D1 will be equal to 100.

BREAK/Break cycle

When the processing between the FOR to NEXT instruction is executed unconditionally times, the next processing of the NEXT instruction will be performed.

-[BREAK]

Features

• Forcibly end the repeated processing by FOR to NEXT instructions.

• This instruction can only be between FOR to NEXT, otherwise an operation error will be reported.

• The BREAK instruction can only jump out of the loop nesting structure where the instruction itself is located.

• When the contact is connected, the loop structure of the FOR to NEXT instruction where it is located is forced to end, as shown in the figure below.

M0 turns ON, no matter how many cycles are left to execute, jump directly to step 35 to execute the program.

M4 turns ON, no matter how many loops are left to execute, jump directly to step 50 to execute the program.

Error code

Error code	Content
4186H	BREAK instruction is not used between FOR to NEXT instructions

Example

The program INC D0 will be executed 10 times, and INC D1 will be executed 100 times.

When M0 is OFF, D0 will be equal to 10 and D1 will be equal to 100 after execution.

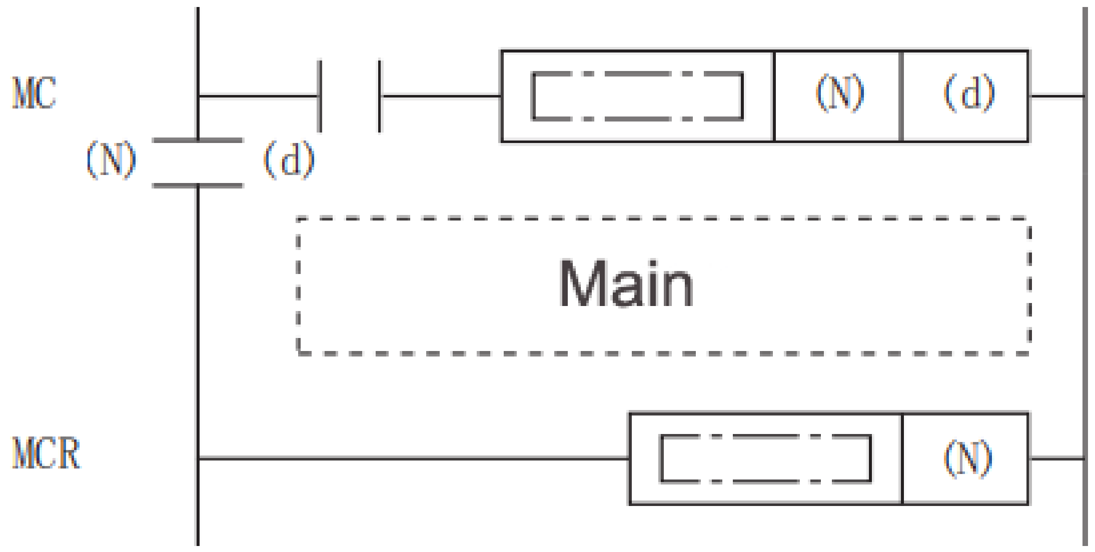
When M0 is ON, the BREAK instruction is executed, and the current loop is exited. The INC D1 instruction will not be executed, and the result D1=0.

Master Control Instructions

MC and MCR instructions

• MC: Start main control.

• MCR: End the main control.



Content, range and data type

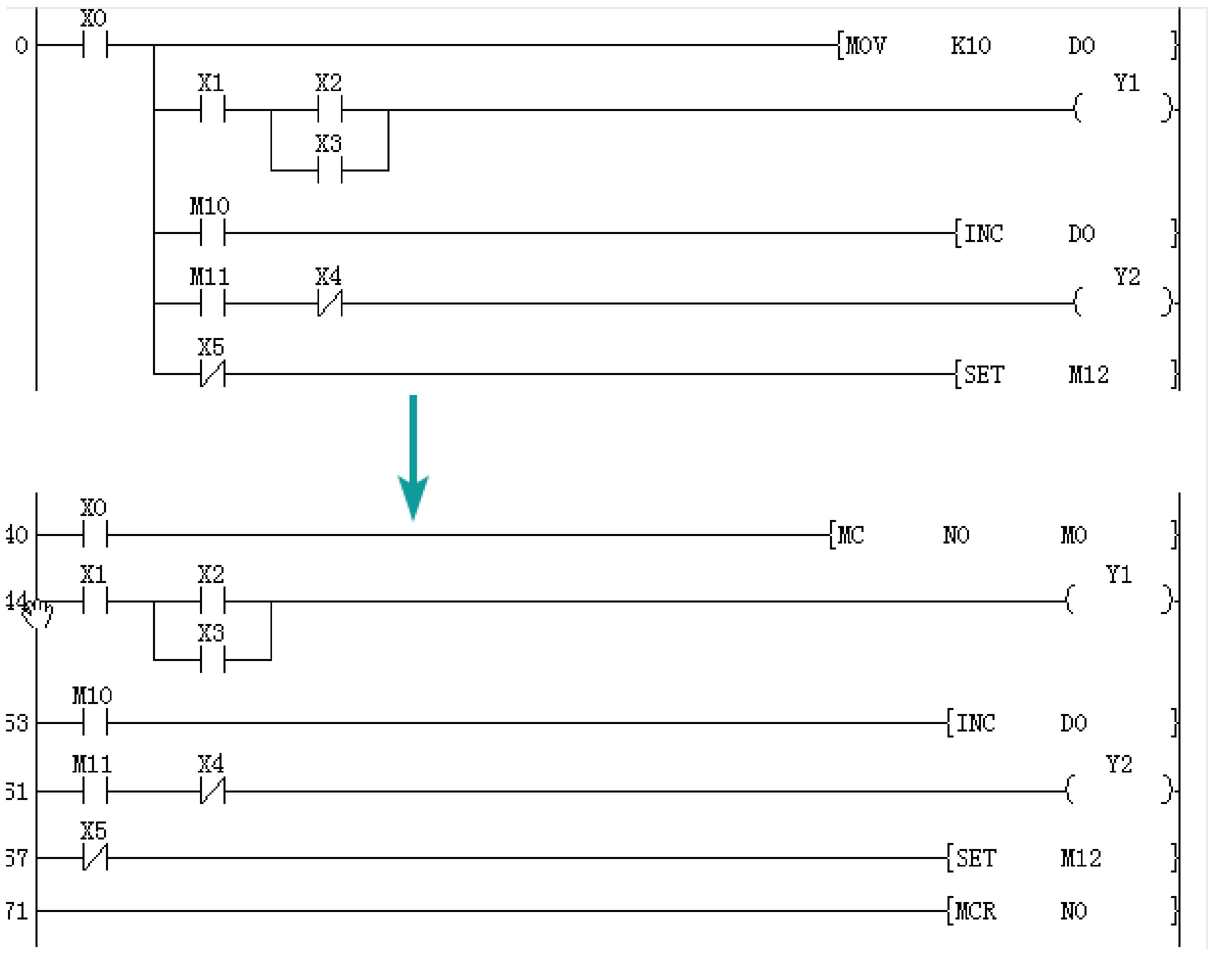
Parameter	Content	Range	Data type	Data type (label)
(N)	Nested ID N	0 to 7	Signed BIN 16 bit	ANY16
(d)	Device number that is turned ON	-	Bit	ANY_BOOL

Instruction	Parameter	Devices												Offset modification	Pulse extension
		KnX	KnY	KnM	KnS	T	C	D	R	SD	K	H	E	[D]	XXP
MC	Parameter1	Only use N0 to N7													
	Parameter2	●	●	●	●	●	●	●	●	●	●	●	●		
MCR	Parameter1	Only use N0 to N7													

Features

The main control instruction is used to create an efficient circuit program switching program by opening and closing the common bus of the circuit program.

The transition of ordinary Circuit program and master control Circuit program is as follows:



MC to MCR internal program, X0 must be open to execute

■MC

• When the execution instruction of the MC instruction is turned on by the start of main control, the operation result from the start of the MC instruction to the MCR instruction is the execution result of the instruction (loop). When the MC execution instruction is OFF, the calculation results from the MC instruction to the MCR instruction are as follows.

Devices	**Device status**
Timer	The count value becomes 0, and the coil and contact are all turned off.
Counter, cumulative timer	The coil turns off, but the count value and contact remain in the current state.
Devices in the OUT instruction	Forced to be OFF.
Devices in SET and RST instruction	Keep the current state

Devices in basic and application instructions

• For MC instructions, the same nesting (N) number can be used multiple times by changing the device of (d).

• When the MC instruction is ON, the coil of the device specified in (d) will turn ON. In addition, when the same device is used in an OUT instruction, etc., it becomes a double coil. Therefore, the device specified in (d) must not be used in other instructions.

Key points:

If there are instructions that do not require contact (such as, FOR to NEXT instructions).If the instruction after MC can not affect the main CPU module, the instruction will execute.

■MCR

• The release instruction of the main control indicates the end of the main control range.

• Do not add a contact instruction before the MCR instruction.

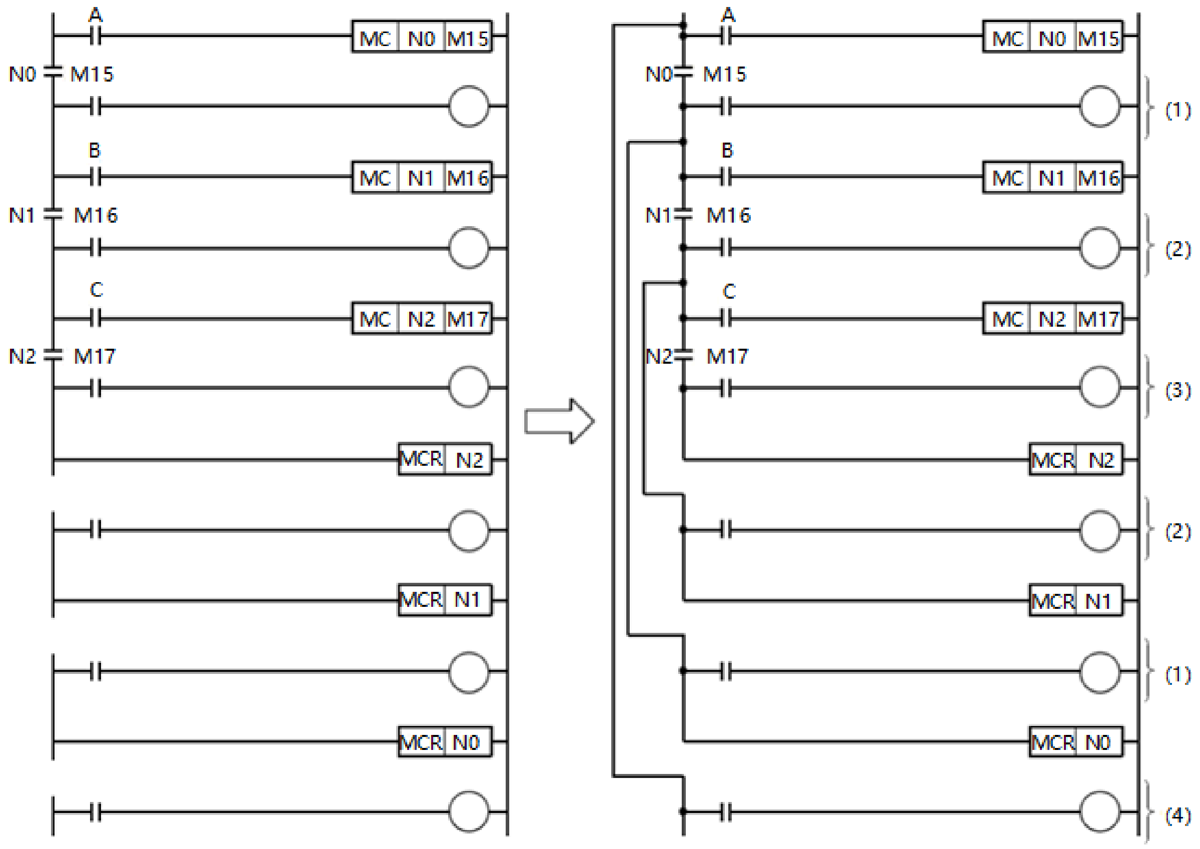
• When using, MC instruction and MCR instruction of the same nesting number should be used. However, when the MCR instruction has a nested structure concentrated in one position, all main controls can be terminated by the smallest number (N) number. (Refer to notes)

■Nested structure

The main control instruction can be used through a nested structure. Each main control section is distinguished by nesting (N). N0 to N7 can be used for nesting.

By using the nested structure, it is possible to create a Circuit program that sequentially restricts the execution conditions of the program. The Circuit program using the nested structure is shown below.

(Left: Display of engineering tools, Right: Actual action loop)



1. Execute when A is ON;
2. Execute when A and B are ON;
3. Execute when A, B, C are ON;
4. It has nothing to do with A, B,

#Note:

• If there is no instruction (LD, LDI, etc.) connected to the bus after the MC instruction, a program structure error occurs.

• MC to MCR instructions cannot be used in FOR to NEXT, STL to RET, subroutines, events, and interrupts. In addition, there cannot be instructions such as IRET, FEND, END, RET (SRET) inside MC to MCR to block.

• There can be up to 8 nests (N0 to N7). In the case of nesting, the MC instruction is used from the small number of nesting (N), while the MCR instruction is used from the old number. If the order is reversed, it does not become a nested structure, so the CPU module cannot operate normally.

• When the MCR instruction is a nested structure concentrated in one location, all main control can be ended by the smallest number (N) number.

Error code

No operation error

Example

(1) No nested structure

Main control program 1

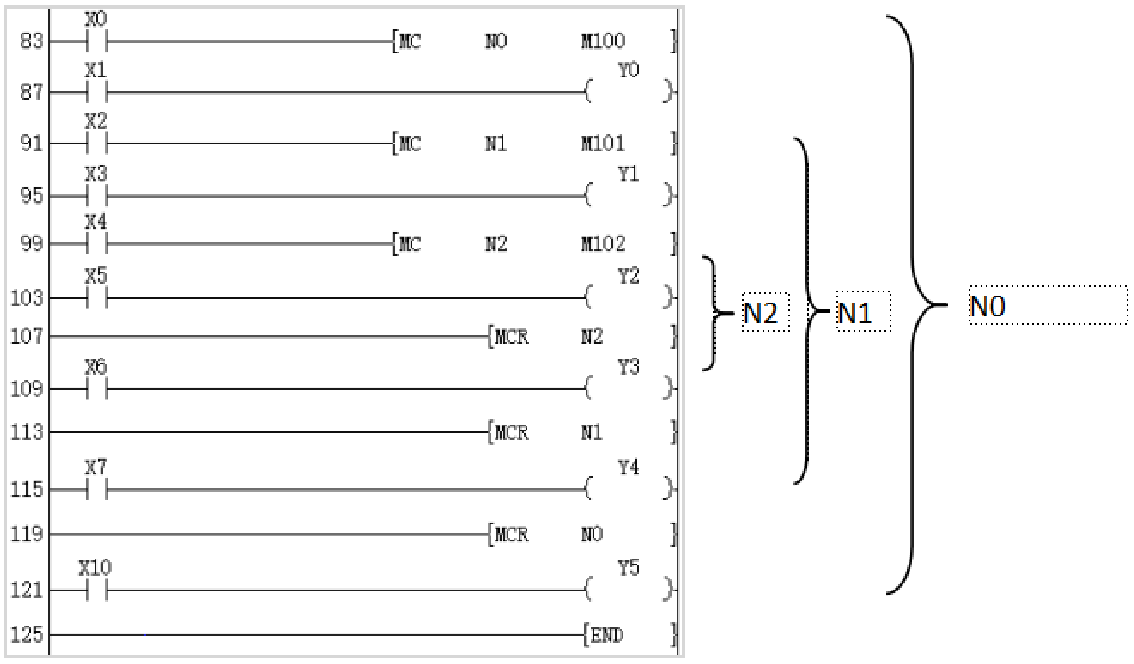
Main control program 2

The main control program 1 and the main control program 2 do not belong to the nested structure, so you can use N0 programming. There is no limit to the number of times N0 can be used in this case.

(2) Nested structure

When using the MC instruction, the number of nesting level N increases sequentially.(N0→ N1→ N2→ N3→ N4→ N5→ N6→ N7). When returning, use the MCR instruction to release from the larger nesting level. (N7→ N6→ N5→ N4→ N3→ N2→ N1 → N0).

For example, when MCR N6 and MCR N7 are not programmed, if MCR N5 is programmed, the nesting level will return to 5 at once.The nesting level can be programmed up to 8 levels (N7).



As shown above:

87 Walk: Level N0, Y0 will follow X1 state only when X0 is ON.

95 Walk: Level N1, Y1 will follow X3 state only when X0 and X2 are both ON.

103 Walk: Level N2, Y2 will follow X5 state only when X0, X2, and X4 are ON at the same time.

109 Walk: Level N1, use MCR N2 to return to level N1. Y3 will follow the state of X6 only when X0 and X2 are both ON.

115 walk: level N0, use MCR N1 to return to level N0. Y4 will follow the state of X7 only when X0 is ON.

121 Walk: Does not belong to the main control structure, has nothing to do with X0, X2, X4, Y5 follows the state change of X10.

Watchdog reset

WDT/watchdog timer

The watchdog timer is reset by the program.

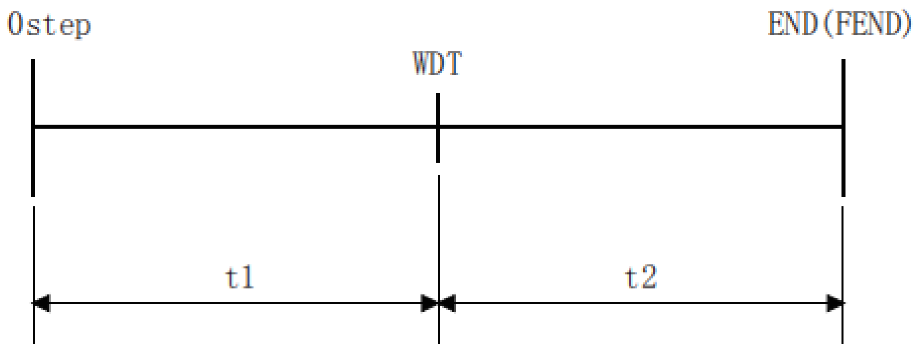
-[WDT]

Features

• Reset the watchdog timer through the program.

• Use when the scan time exceeds the set value of the watchdog timer depending on conditions.

• For t1 from step 0 to WDT instruction, and from WDT instruction to END instruction, do not exceed the set value of the watchdog timer.

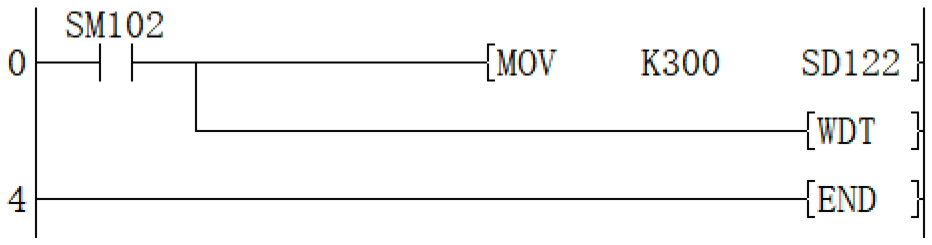


• The WDT instruction can be used more than twice in one scan.

#Note:

• The watchdog timeout time can be set in the special register SD122. The default is 200ms.

• Use the special relay SM122 to control whether to turn on the watchdog timer function. The WDT instruction will be invalid after closing.



(1) The watchdog timer time is set to 300ms;

(2) Refresh the watchdog timer.

Error code

There is no operation error.

Example

The FOR to NXET instruction loop takes a long scan period for many times, which may exceed the set watchdog timer 300ms, causing the PLC to report an error and cannot continue to run. After turning on M0, the WDT instruction will run, and the watchdog timer is updated every cycle , So that it will not report an error to execute the program normally.