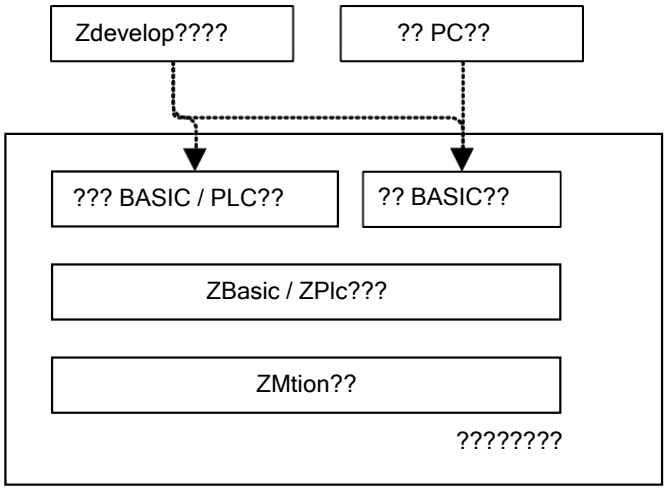
ZMotion Plc Programming Manual

Version 1.0

ZBasic is ZMotion motion controller supports the Basic programming language.

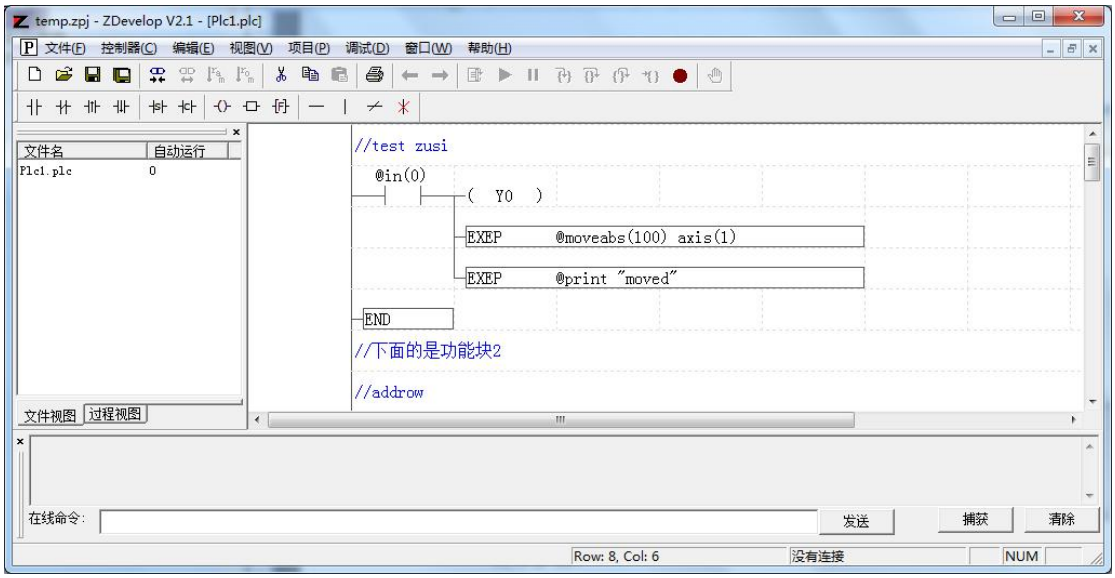
ZPlc is ZMotion motion controller support Plc language, you can use the ladder and statements table.



Chapter One ZPLC programming entry

Writing and debugging programs require ZPLC ZDevelopV2.0 or later software, ZMC motion controller (or emulator) V3.0 or later firmware, PC sends a command-line needs zmotion.dll dynamic libraries.

1.1 ZDevelop Development Software



Simple steps:

1, click on "File (File)" menu, New Project.
2. Click the "New (New)" option to create Plc file, write code, click "Save", the same directory as the project file and select Save.
3. Click on "Project (Project)" - "added to the project", select the file you just saved Plc.
4, double-click the file list window on the left correspond Plc file "auto-run (AutoRun)" option, enter the number 0.
5, connected to the controller or emulator, click on the "Download to RAM" menu, this time just downloaded to the controller and automatically run the program.

Not build project, only the program files can not be downloaded to the controller.

1.2 Setup program runs on power

It must be stored in the program ROM, and a corresponding set autorun file Autorun (Autorun) task number, see Introduction zdevelop software tools.

1.3 Debugging

After zdevelop connect the controller to select from the menu commissioning / start debugging, to ensure consistent program files and program files PC controller, it may cause the cursor position errors; only when debugging UNLOCK.

 Debug the machine pay attention to safety! Be sure to design effective safety devices in the machine, and add an error in the software

Handler, otherwise the losses caused by the positive movement technology company has no obligation or responsibility responsible.

1.4 Multitasking

PLC master task: PLC task to run automatically setting a first main task parameters related to status of the PLC cycle obtained from this task.

PLC controller supports multi-task to run at the same time, it can also run together Plc and Basic task.

When the interrupt task runs, PLC main task is halted, the other PLC task will not be affected, run concurrently with interrupt task.

1.5 Power-down storage

Each register is retentive part, similarly with VR.

SETCOM command by the configuration register D is held down.

1.6 PLC Step count

Program capacity different types of controllers are not the same, and ZBASIC program sharing space, there is no ZBASIC program, the majority of the total number of steps in 40K controller step down.

The number of steps occupied by each instruction is the number of register + 1, @ registers takes up more step.

1.7 Interrupt function

EI interrupt is enabled, the same BASIC commands: INT_ENABLE = 1
DI disable interrupts, with the BASIC command: INT_ENABLE = 0
ONPOWEROFF: break down
INT_ONn: active interrupt input variables, n- input port number
INT_OFFn: become invalid input interrupt, n- enter the port number
ONTIMERn: timer interrupt, n- timer number

1.8 Connect the display unit

The controller supports MODBUS-RTU protocol or MODBUS-TCP protocol connection man-machine

interface.

use ADDRESS protocol parameter setting station number, a default. Serial port connected configuration corresponding to the serial number, the configuration of the man-machine interface connected to the PLC Ethernet local IP address is the IP port number of the controller 502 (IP-based emulator machine 127.0.0.1).

Wei Lun touch screen Ethernet settings Figure

1.9 common problem

When the program motion error, ZDevelop software will show an error message, if you do not see an error message, you can command line input? * Task again see an error message, double-click the error messages can be automatically switched to the program error location.

problem	Possible Causes
stop of error: 2049: Line not ended.	1, Part of the command must occupy an entire row. 2, GSUB Call does not need parentheses()
stop of error: 2033: Unknown label is met.	1Undefined variable or array. 2Undefined SUB process 3, There is an array of definitions, but the definition of the statement is not executed,

Chapter register (Device) type

2.1 Basic register@

This register can execute the statement or expression BASIC embedded Plc, the format conversion will be performed in different instruction automatically to match.

For example:

LD @ (a + b)

EXEP @MOVEABS (100) AXIS (0) // absolute positioning

 @ Register a single number of characters should not exceed 1000 bytes.

2.2 Input Register X

This corresponds to a general purpose input register, the PLC master task, this register will be backed up at the beginning of each cycle, to avoid changes in the input scan phase, other tasks read current real time input state.

PLC Programming, X is octal (X0 ~ X7, X10 ~ X17, ...), while the input port IN to the controller 10 hex, paying particular attention to the preparation process, to make binary conversion, such as corresponding to X24 port IN20, IN8 correspond X10.

When you call or by decimal basic program PLC.

2.3 Output register Y

This corresponds to a general purpose output register, the PLC master task, the local hardware output is output to the output port at the end of the cycle.

PLC Programming, Y is octal (Y0 ~ Y7, Y10 ~ Y17, ...), and the output port OP of the controller 10 hex, paying particular attention to the preparation process, to make binary conversion, such OP20 port corresponding to Y24, OP8 corresponds to Y10.

When you call or by decimal basic program PLC.

2.4 Auxiliary bit register M

This corresponds to register MODBUS_BIT, (0xregister). Number: 0-4095

Retentive: 2048-2175

2.5 Status Register S

Number: 0-999
Retentive: 0-127

2.6 Data register D

This corresponds to register MODBUS_REG, (4xRegister), bit can be accessed by D.. Number: 0-2047
according to SETCOM the VR configured to determine a way to map (see ZBASIC SETCOM instruction manual). When you need to keep down, it is recommended variable = 3
variable parameters are global settings, a total of all ports.

	description
0	VRAt this time a VR Mapped to a MODBUS_REG.
1	TABLE At this time a TABLE Data is mapped to a MODBUS_REG. (Not recommended)
2(default)	system MODBUS Register, this time VR versus MODBUS send Memory is two independent sections.
3	VR_INT Mode, where a VR_INT Mapped to two MODBUS_REG.

2.7 counter C

Length register 32, when accessed by a 16-bit instruction automatically uses the lower 16 bits. Number: 0-127
Retentive: 100-107

2.8 Timer T

Length register 32, when accessed by a 16-bit instruction automatically uses the lower 16 bits. BASIC TIMER with the command
Number: 0-127
Retentive: 100-103

2.9 Index register V

16-bit length
register. No: 0-7

2.10 Index register Z

16-bit length
register. No: 0-7

2.11 Floating-point register DT

This register is 32-bit floating-point, will be automatically converted to other formats type when accessed, automatically remove rpm decimal part to an integer.

This registers TABLE ZBASIC is the same, may be used to transfer data between and ZBASIC. Number: 0-TSIZE-1

2.12 Local Register LV

Effective local function registers, each instance of each run using a different function data; each function 8
A LV, No. 0-7; this is a 32-bit floating-point register.

This register and the ZBASIC LOCAL similar, when ZBASIC call ZPLC, automatic call parameters passed in LV
register.

LV memory can be simultaneously used
as index registers. No: 0-7

2.13 Label register L

This register is defined by LBL instruction, registers a total of 64 L, L registers via CALL / JUMP
Instruction to
call or jump.
No: 0-63

2.14 Special register M

On the run flag M8000 =
Run flag M8001 = Off
Initial M8002 = ON
Initial M8003 = OFF
M8004 = error
flag M8011 =
10MS M8012 =

100MS M8013 =
1S M8014 = 60S
M8020 = ZeroFlag

M8021 = borrow
M8022 = Carry
M8023 = decimal point mark

M8100-M8199 axis IDLE flags 0-99
BUFFER remaining shaft M8200-M8299 flag 0-99

2.15 Special register D

D8000 = scan time monitoring
D8001 = version
D8004 = Error
D8010 = minimum
scan time scan time
D8011 =
D8012 = maximum scan time
D8013 = sec
D8014 = min
When D8015 =
D8016 = Day
D8017 = month
In D8018 =
D8019 = week

D10000-D10198 = DPOS, floating-point manner, each axis representing two REG. D11000-D11198 = MPOS, floating-point manner, each axis representing two REG. D12000-D12198 = VPSPEED, floating-point manner, each axis representing two REG.
D13000-D13128 = AOUT, analog output
D14000-D14256 = AIN, the analog input

third chapter ZPLC Instruction List

3.1 Common commands

3.1.1 // comment

// This behavior represents the comments, which always occupy an entire row.

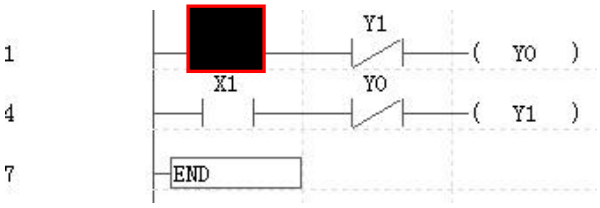
3.1.2 LD

Instructions:

LD instruction for the start of the left bus bar A contact by a contact or a contact point A of the circuit blocks. Operands:

S: X, Y, M, S, T, C, D, @ a.

Programming showsexample: LD



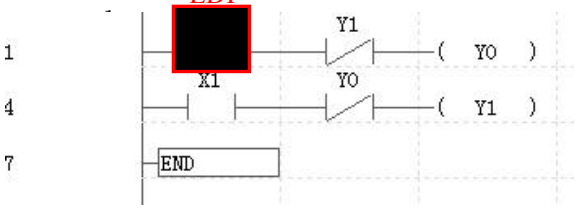
3.1.3 LDI

Instructions:

LD instruction for the start of the left bus bar B contact point B or a start point of the circuit blocks. Operands:

S: X, Y, M, S, T, C, D, @ a.

Programmin LDI



3.1.4 LDP

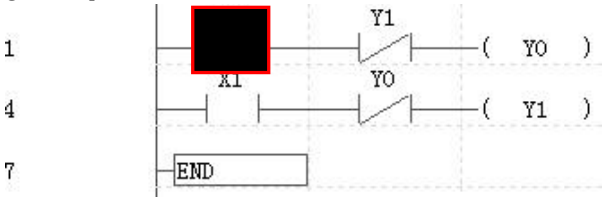
Instructions:

And use the same LDP instruction LD, LDP but only at the rising edge when the specified device (changed from OFF to ON time), an ON operation period.

Operands:

S: X, Y, M, S, T, C, D, @ a.

Programmin LDP
g example:



3.1.5 LDF

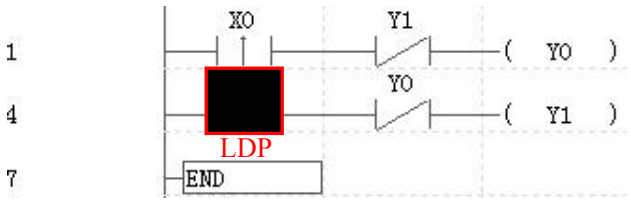
Instructions:

Usage and LD LDF same instruction, but only on the falling edge LDF when the specified device (changed from ON to OFF time), an ON operation period.

Operands:

S: X, Y, M, S, T, C, D, @ a.

Programming example:



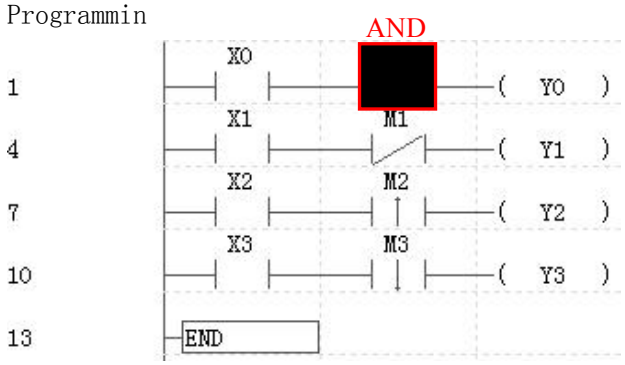
3.1.6 AND

Instructions:

A contact for the AND instruction connected in series. Operands:

S: X, Y, M, S, T, C, D, @ a.

Programmin



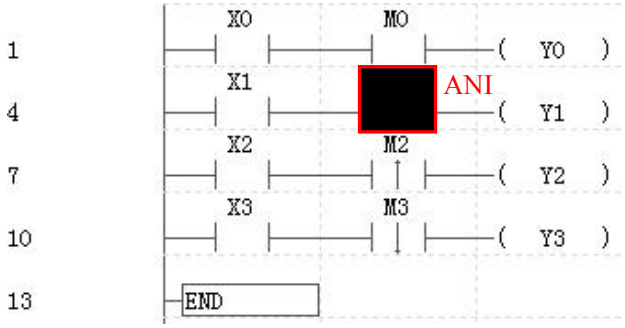
3.1.7 ANI

Instructions:

ANI instructions for contact B is connected in series. Operands:

S: X, Y, M, S, T, C, D, @ a.

Programming example:



3.1.8 ANDP

Instructions:

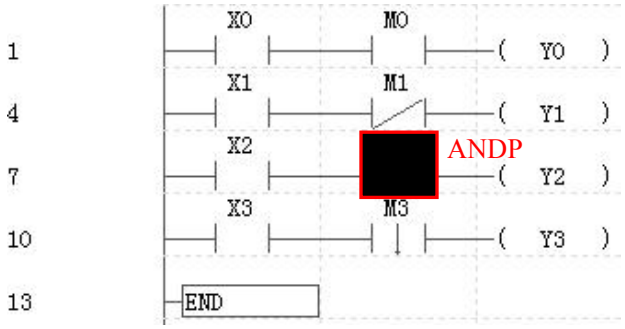
Usage instructions AND ANDP and the same, but only at the rising edge ANDP specified device (changed from OFF to ON

When the time), an ON

operation period. Operands:

S: X, Y, M, S, T, C, D, @ a.

Programming example:



3.1.9 ANDF

Instructions:

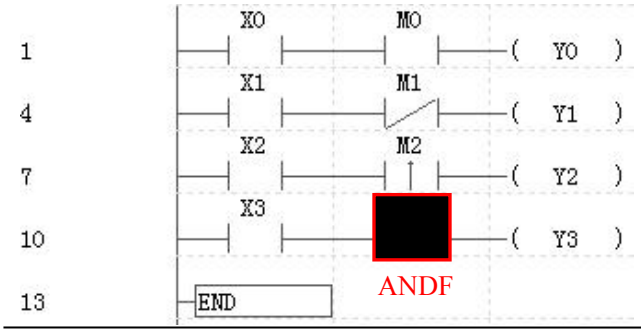
Usage instructions and ANDF AND same, but only on the falling edge ANDF specified device (changed from ON to OFF

When the time), an ON

operation period. Operands:

S: X, Y, M, S, T, C, D, @ a.

Programming example:



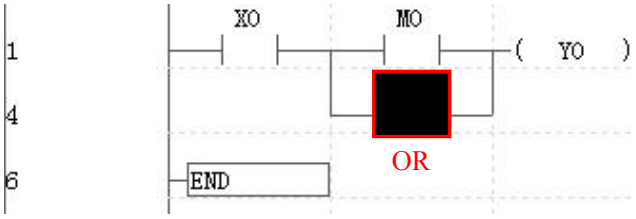
3.1.10 OR

Instructions:

OR instruction for a parallel connection of A contact. Operands:

S: X, Y, M, S, T, C, D, @ a.

Programming example:



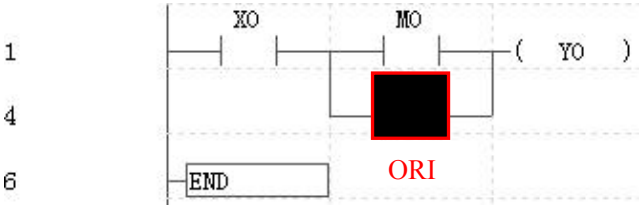
3.1.11 ORI

Instructions:

ORI instructions for parallel connection a B contact. Operands:

S: X, Y, M, S, T, C, D, @ a.

Programming example:



3.1.12 ORP

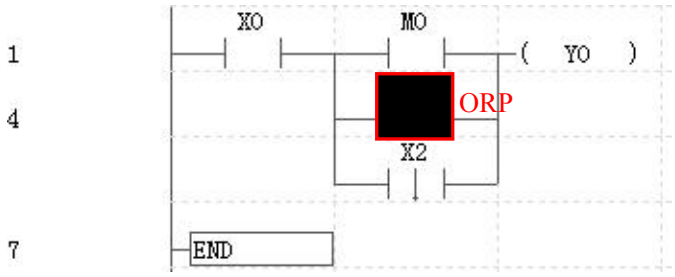
Instructions:

ORP OR instruction usage and the same, but only at the rising edge ORP when the specified device (changed from OFF to ON time), an ON operation period.

Operands:

S: X, Y, M, S, T, C, D, @ a.

Programming example:



3.1.13 ORF

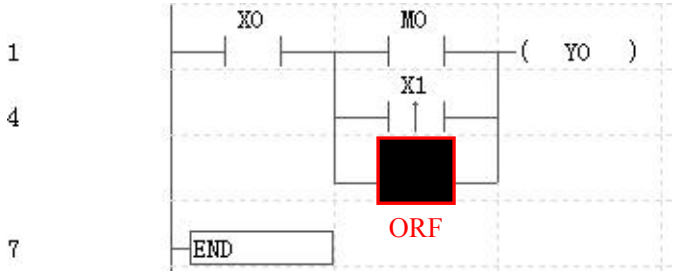
Instructions:

ORF OR instruction usage and the same, but the rising edge of ORF only when the specified device (changed from ON to OFF time), an ON operation period.

Operands:

S: X, Y, M, S, T, C, D, @ a.

Programming example:



3.1.14 PLS

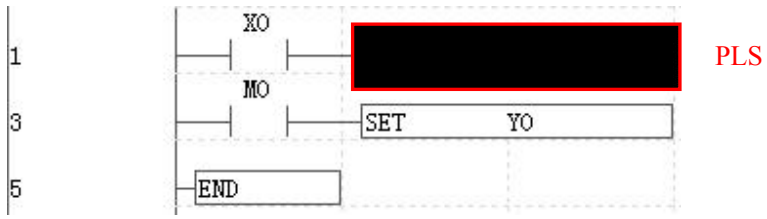
Instructions:

PLS rising edge detection instruction. When the PLS instruction is executed, only the operation of the device turns ON in a period of the drive input from OFF to ON later.

Operands:

S: Y, M

Programming example:



3.1.15 PLF

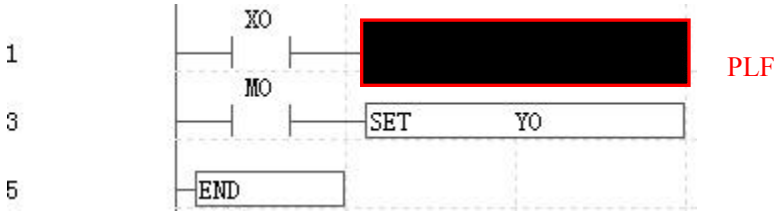
Instructions:

PLF rising edge detection instruction. When the PLF instruction is executed, only the operation of the device turns ON in a period from ON to OFF driving input later.

Operands:

S: Y, M

Programming example:



3.1.16 STL

Instructions:

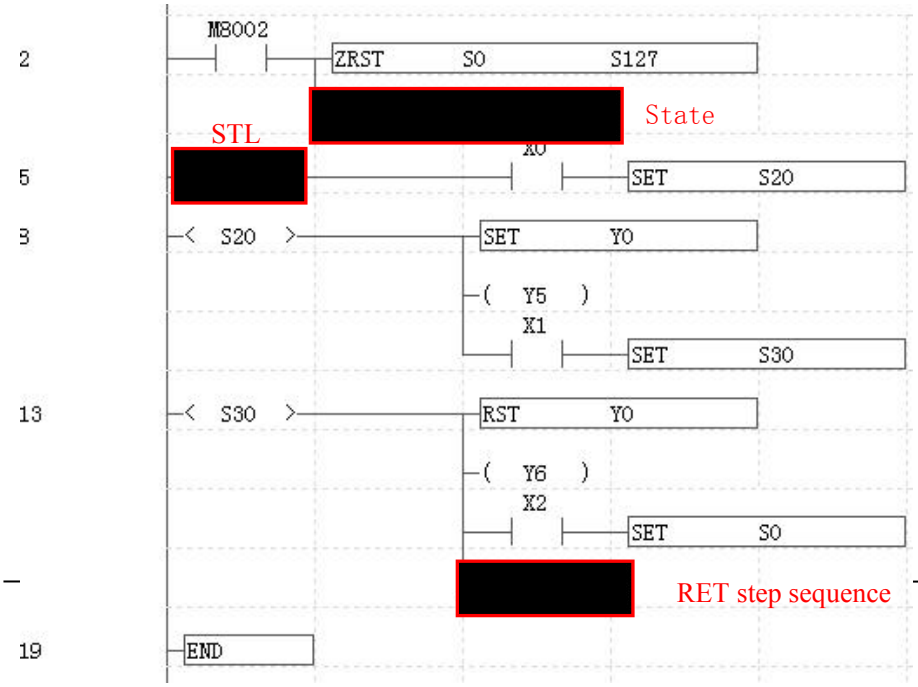
Use stepladder program of instructions to the mechanical action, based on the distribution state of each of the steps S, the state as the connection contacts(STL contacts) of the circuit, the order of input conditions and output control program.

When the current state (S0) to a next state (S1) transfer operation in the scanning period of two states are implemented; next scan cycle when executed, a current state (S0) is the next state (S1) is reset, operation in the current state (S0) is not executed, the input elements are turned off OUT.

Operands:

S: Sn

Programming example:



shapeTransfer state can only use SETSn, Wh en used out, the use of Sn as an ordinary auxiliary relay; required stepping of a

program is
completed
RET end of the program.

3.1.17 RET

Instructions:

RET instruction represents the end of the step procedure, so in the end there must be a series of step point RET instruction. Operands:

no

Programming example:

See STL instruction.

3.1.18 SET

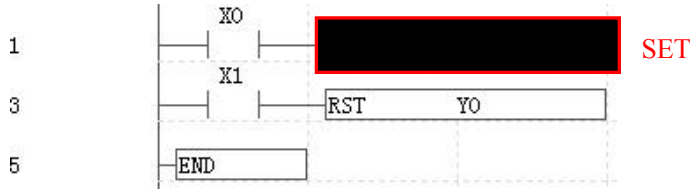
Instructions:

When the SET instruction is driven, its designated elements are set to ON, and the element is provided will remain ON, regardless of the SET

Whether the instruction is still driven. usableRST instruction element to OFF. Operands:

S: Y, M, S

Programming example:



3.1.19 RST

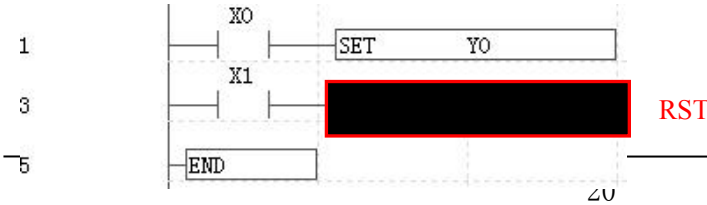
Instructions:

when RST instruction is driven, the element that specifies the bit is set to OFF, and regardless of whether the RST instruction is still driven, the bit is set element will remain OFF state. The element may utilize the SET instruction bit to ON.

When the RST instruction is driven, the current value of the word device designated operands are cleared:

S: Y, M, T, C, D, S,V, Z, DT, LV

Programming example:



3.1.20 NOP

Instructions:

NOP instruction in the program without any operation, and therefore will remain after the implementation of the original logic operation, use the following: Replace want to delete a command, but do not want to change the length of the program, you can NOP.

3.1.21 EU

Instructions:

EU (NP) instruction when the cumulative register from 0 to 1, this instruction will accumulate register sustain period of a first scan, and after the second scanning period, the accumulation register to 0 automatically.

Operands:

no

Programming example:

3.1.22 ED

Instructions:

ED (PN) register by the instruction when the cumulative time becomes 0, the instruction register 1 will maintain a cumulative period of one scan, and after the second scanning period, the accumulation register to 0 automatically.

Operands:

no

Programming example:

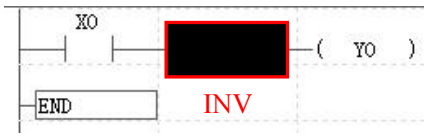
3.1.23 INV

Instructions:

INV INV instruction is an instruction prior to invert the result of logical operation instructions, without the specified device number. Operands:

no

Programming example:



3.1.24 EXE

Instructions:

PLC can call the number by the standard instruction operation Basic EXE command:

Basic standard programming instructions Example:

Basic call the print function, when X0 changes from OFF to ON, print out 55.

3.1.25 EXEP

Instructions:

EXEP usage and EXE same command, but only after the drive input from OFF to ON, only called Basic standard instruction

Operands:

Basic standard programming instructions Example:

Basic call the print function, when X0 is turned on, print out 55.

3.1.26 MPS

Instructions:

MPS, MRD, MPP instructions have no operands, the three instructions are occupied by a number of program steps.

Embedded PLC has 11 stack space, that can push the maximum depth is 11. Each use MPS

The current result onto the first segment is stored, the result of a previously pushed sequentially into the next stage. The first segment MPP instructions read out and remove it, while the following means sequentially to move forward. MRD instruction read the first paragraph, but does not delete it. Other units remain unchanged. Use these three instructions can facilitate multi-drop programming.

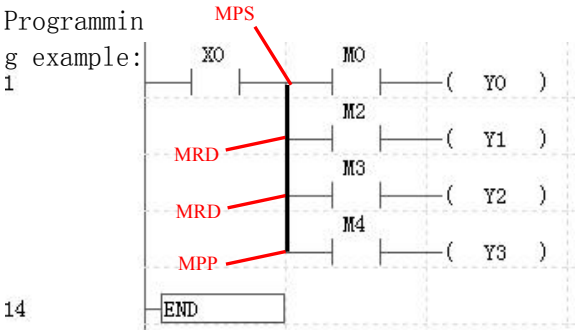
When a multi-drop programming, theMPS preceding calculated result is stored, beyond the branch may be utilized MRD, MPP calculation result of the preceding read out from the stack, and then calculated later. The last branch must use MPP, guarantee MPS, MPP use the same number of times. Note that after the use of MPP, you can not use MRD reads the result of the operation, which is to be placed at the end of a branch MPP use.

MRD instruction can be used multiple times, there is no limit. MPS maximum number of continuous use is 11, but can be used multiple times. Each instruction has a MPS instruction corresponding MPP, MPP number of no more than the number of MPS.

Operands:

no

Programming example:




```
1   ld x0
2   MPS
3   AND     M0
4   OUT Y0
5   MRD
6   AND     M2
7   OUT Y1
8   MRD
9   AND     M3
10  OUT Y2
11  MPP
12  AND     M4
13  OUT Y3
14  end
15
```

3.1.27 MRD

See MPS instruction.

3.1.28 MPP

See MPS instruction.

3.1.29 TMR

Instructions:

when When TMR instruction is executed, its specified power receiving coil of the timer, a timer is started, when the contact operation of the timer reaches the value specified by the timing, the corresponding timer.

Length register 32 when accessed by a 16-bit instruction automatically uses the lower 16

bits.

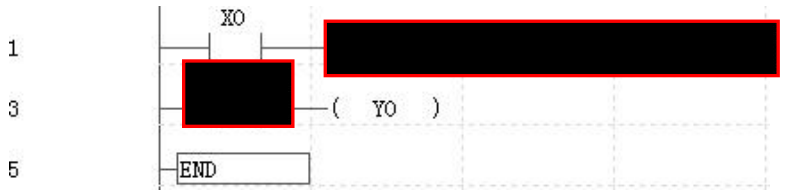
A total timer 128, respectively T0 ~ T127, T100 ~ T103 which is retentive type. Operands:

S1: T0 ~ T 127

S2: K, D

Programming example:

X0 is turned on after the timer starts timing T0, T0 2 seconds normally open contacts closed, Y0 conducting coil.



3.1.30 CNT

Instructions:

The command is 16-bit count instructions that, when executed by a CNT OFF to ON, the counter designated by the energized coil to the power, the count value is incremented by 1 when the count reaches the specified value, the operation of the counter contact.

Length register 32 when accessed by a 16-bit instruction automatically uses the lower 16 bits.

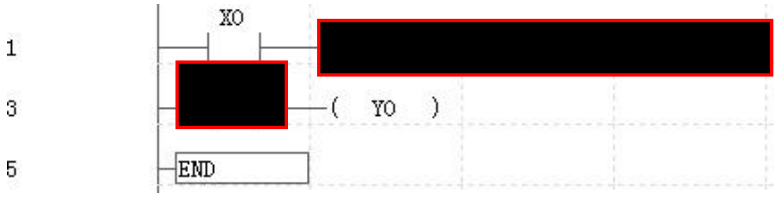
Counter is a total of 128, namely C0 ~ C127, C100 ~ C107 which is a hold-type. Operands:

S1: C0 ~ C 127

S2: K, D

Programming example:

X0 from OFF to ON, the counter value by 1 C0, C0 when the value is 10, C0 normally open contacts closed, Y0 is turned on.



3.1.31 DCNT

Instructions:

The command is 32-bit count instructions that, when executed by a CNT OFF to ON, the counter designated by the energized coil to the power, the count value is incremented by 1 when the count reaches the specified value, the operation of the counter contact.

Length register 32 when accessed by a 16-bit instruction automatically uses the lower 16 bits.

Counter is a total of 128, namely C0 ~ C127, C100 ~ C107 which is a hold-type. Operands:

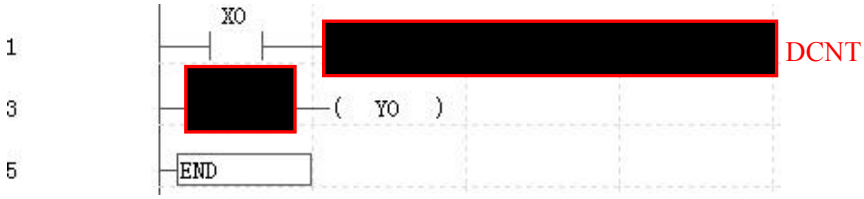
S1: C0 ~ C 127

S2: K, D

Programming example:

X0 from OFF to ON, the counter value by 1 C0, C0 when the value is 10, C0 normally open contact closure, Y0

Turned on.



3.1.32 MC

Instructions:

1. Is master start command, when MC instruction execution, instructions are located between the MC and MCR instructions will be executed normally. When MC instruction Off, positioned between the MC and MCR instructions instruction operation is as follows:

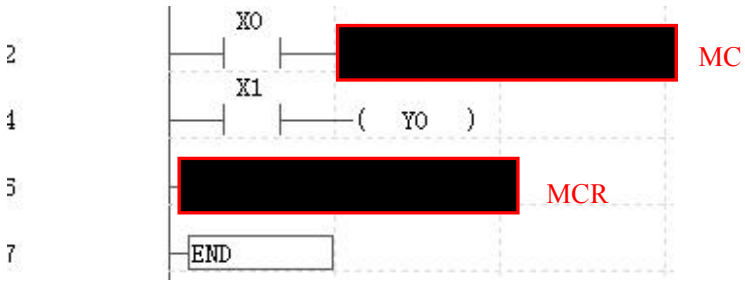
General timer: timer value to zero, the coil energized, the contact does not operate the timer subroutine: timer value to zero, the coil energized, the contact operation is not accumulative timer: a coil de-energized, the current count value and the contact holding state counter: ring loss of power, to maintain the current count and the contact state SET, RST instruction driven elements: current state holding

By instruction: all without operation

2.MCR end command for the master, the master program is placed Finally, there is not point instruction before MCR instruction. Operands:

S: N

Programming example:



3.1.33 MCR

See MC instruction.

3.2 Compare instruction contacts

3.2.1 LD * / LDD * / FLD * AND * / ANDD * / FAND * OR * / ORD * / FOR *

Instructions:

When the content of the contacts S1 and S2 compare instruction of the instruction for comparison, the comparison results meet the determination condition, the conduction instruction, do not meet the determination condition, the instruction is non-conductive.

LD * / LDD * / FLD * Instruction List:

16-bit instruction	32-bit instruction	32-bit floating-point instructions	Conducting condition	Nonconducting conditions
LD =	**D**LD =	**F**LD =	**S1**= S2	**S1**≠ S2
LD>	**D**LD>	**F**LD>	**S1**> S2	**S1**≦ S2
LD <	**D**LD <	**F**LD <	**S1**<S2	**S1**≧ S2
LD <>	**D**LD <>	**F**LD <>	**S1**≠ S2	**S1**= S2
LD <=	**D**LD <=	**F**LD <=	**S1**≦ S2	**S1**> S2
LD> =	**D**LD> =	**F**LD <=	**S1**≧ S2	**S1**<S2

AND * / ANDD * / FAND * Instruction List:

16-bit instruction	32-bit instruction	32-bit floating-point instructions	Conducting condition	Nonconducting conditions
AND =	**D**AND =	**F**AND =	**S1**= S2	**S1**≠ S2
AND>	**D**AND>	**F**AND>	**S1**> S2	**S1**≦ S2
AND <	**D**AND <	**F**AND <	**S1**<S2	**S1**≧ S2
AND <>	**D**AND <>	**F**AND <>	**S1**≠ S2	**S1**= S2
AND <=	**D**AND <=	**F**AND <=	**S1**≦ S2	**S1**> S2
AND> =	**D**AND> =	**F**AND> =	**S1**≧ S2	**S1**<S2

OR * / ORD * / FOR * Instruction List:

16-bit instruction	32-bit instruction	32-bit floating-point instructions	Conducting condition	Nonconducting conditions
OR =	**D**OR =	**F**OR =	**S1**= S2	**S1**≠ S2
OR>	**D**OR>	**F**OR>	**S1**> S2	**S1**≦ S2
OR <	**D**OR <	**F**OR <	**S1**<S2	**S1**≧ S2

OR <>	**D**OR <>	**F**OR <>	**S1**≠ S2	**S1**= S2
OR <=	**D**OR <=	**F**OR <=	**S1**≦ S2	**S1**> S2
OR> =	**D**OR> =	**F**OR> =	**S1**≧ S2	**S1**<S2

Operands:

S1: K, H, KnX, KnY, KnM, T, C, D S2:
K, H, KnX, KnY, KnM, T, C, D

3.2.2 LD =

See 3.2.1.

3.2.3 LDD =

See 3.2.1.

3.2.4 LD>

See 3.2.1.

3.2.5 LDD>

See 3.2.1.

3.2.6 LD <

See 3.2.1.

3.2.7 LDD <

See 3.2.1.

3.2.8 LD <>

See 3.2.1.

3.2.9 LDD <>

See 3.2.1.

3.2.10 LD <=

See 3.2.1.

3.2.11 LDD <=

See 3.2.1.

3.2.12 LD> =

See 3.2.1.

3.2.13 LDD> =

See 3.2.1.

3.2.14 AND =

See 3.2.1.

3.2.15 ANDD =

See 3.2.1.

3.2.16 AND>

See 3.2.1.

3.2.17 ANDD>

See 3.2.1.

3.2.18 AND <

See 3.2.1.

3.2.19 ANDD <

See 3.2.1.

3.2.20 AND <>

See 3.2.1.

3.2.21 ANDD <>

See 3.2.1.

3.2.22 AND <=

See 3.2.1.

3.2.23 ANDD <=

See 3.2.1.

3.2.24 AND> =

See 3.2.1.

3.2.25 ANDD> =

See 3.2.1.

3.2.26 OR =

See 3.2.1.

3.2.27 ORD =

See 3.2.1.

3.2.28 OR>

See 3.2.1.

3.2.29 ORD>

See 3.2.1.

3.2.30 OR <

See 3.2.1.

3.2.31 ORD <

See 3.2.1.

3.2.32 OR <>

See 3.2.1.

3.2.33 ORD <>

See 3.2.1.

3.2.34 OR <=

See 3.2.1.

3.2.35 ORD <=

See 3.2.1.

3.2.36 OR> =

See 3.2.1.

3.2.37 ORD> =

See 3.2.1.

3.2.38 LD

Instructions:

Contact point log mmnjhic operation instruction LD &, LD |, LD ^, LDD &, LDD |, LDD ^, instruction S1 operation content for bit S2 of the result, the instruction conduction is not 0, the result is zero then the instruction non-conductive.

16-bit instruction	32-bit instruction	Conducting condition	Nonconducting conditions
LD &	DLD &	S1 & S2 ≠ 0	S1 & S2 = 0
LD |	DLD |	S1 | S2 ≠ 0	S1 | S2 = 0
LD ^	DLD ^	S1 ^ S2 ≠ 0	S1 ^ S2 = 0

Operands:

S1: K, H, KnX, KnY, KnM, T, C, D S2:
K, H, KnX, KnY, KnM, T, C, D

3.2.39 LD &

See 3.2.38 LD #.

3.2.40 LD |

See 3.2.38 LD #.

3.2.41 LD ^

See 3.2.38 LD #.

3.2.42 LDD &

See 3.2.38 LD #.

3.2.43 LDD |

See 3.2.38 LD #.

3.2.44 LDD ^

See 3.2.38 LD #.

3.2.45 AND

Instructions:

Logical operation instructions contact AND &, AND |, AND ^, ANDD &, ANDD |, ANDD ^, instruction S1 operation content for bit S2 of the comparison result is not 0 then the instruction is turned, the comparison result is the instruction is not turned to 0

through.

16-bit instruction	32-bit instruction	Conducting condition	Nonconducting conditions
LD &	**D**LD &	**S1** & **S2** ≠ 0	**S1** & **S2** = 0
LD |	**D**LD |	**S1** | **S2** ≠ 0	**S1** | **S2** = 0
LD ^	**D**LD ^	**S1** ^ **S2** ≠ 0	**S1** ^ **S2** = 0

Operands:

S1: K, H, KnX, KnY, KnM, T, C, D S2:
K, H, KnX, KnY, KnM, T, C, D

3.2.46 AND &

See 3.2.45 AND #.

3.2.47 AND |

See 3.2.45 AND #.

3.2.48 AND ^

See 3.2.45 AND #.

3.2.49 ANDD &

See 3.2.45 AND #.

3.2.50 ANDD |

See 3.2.45 AND #.

3.2.51 ANDD ^

See 3.2.45 AND #.

3.2.52 OR

Instructions:

Contact point logic operation instructions OR &, OR |, OR ^, ORD &, ORD |, ORD ^, S1 operation content for bit S2 of the instruction, the comparison result is not 0 then the instruction 0 is turned on, the comparison result is then this instruction does not conduct.

16-bit instruction	32-bit instruction	Conducting condition	Nonconducting conditions
OR &	DOR &	S1 & S2 ≠ 0	S1 & S2 = 0
OR |	DOR |	S1 | S2 ≠ 0	S1 | S2 = 0
OR ^	DOR ^	S1 ^ S2 ≠ 0	S1 ^ S2 = 0

Operands:

S1: K, H, KnX, KnY, KnM, T, C, D S2:
K, H, KnX, KnY, KnM, T, C, D

3.2.53 OR &

See 3.2.52 OR #.

3.2.54 OR |

See 3.2.45 AND #.

3.2.55 OR ^

See 3.2.45 AND #.

3.2.56 ORD &

See 3.2.45 AND #.

3.2.57 ORD |

See 3.2.45 AND #.

3.2.58 ORD ^

See 3.2.45 AND #.

3.2.59 FLD =

See 3.2.1.

3.2.60 FLD>

See 3.2.1.

3.2.61 FLD <

See 3.2.1.

3.2.62 FLD <>

See 3.2.1.

3.2.63 FLD <=

See 3.2.1.

3.2.64 FLD> =

See 3.2.1.

3.2.65 FAND =

See 3.2.1.

3.2.66 FAND>

See 3.2.1.

3.2.67 FAND <

See 3.2.1.

3.2.68 FAND <>

See 3.2.1.

3.2.69 FAND <=

See 3.2.1.

3.2.70 FAND> =

See 3.2.1.

3.2.71 FOR =

See 3.2.1.

3.2.72 FOR>

See 3.2.1.

3.2.73 FOR <

See 3.2.1.

3.2.74 FOR <>

See 3.2.1.

3.2.75 FOR <=

See 3.2.1.

3.2.76 FOR> =

See 3.2.1.

3.3 Compare instruction and transfer

3.3.1 CMP

Instructions:

CMP is a 16-bit instruction, two compare value, the result (large, equal, small) to the bit soft element. To clear the result of the comparison, to be manually cleared.

Operands:

S1: KnX, KnY, KnM, T, C, D, K,

H S2: KnX, KnY, KnM, T, C, D,

K, HD: Y, M, S

16-bit instruction	32-bit instruction	Whether the pulse type
CMP	**D**CMP	no
CMPP	**D**CMPP	Yes

3.3.2 CMPP

CMP is 16-bit pulse instruction, see CMP instruction.

3.3.3 DCMP

CMP 32-bit instructions, see CMP instruction.

3.3.4 DCMPP

CMP is 32-bit pulse instruction, see CMP instruction.

3.3.5 ZCP

Instructions:

ZCP 16-bit instructions, interval comparison, the result (large, equal, small) to the bit soft element. To clear the result of the comparison, to be manually cleared.

Operands:

S1: KnX, KnY, KnM, T, C, D, K,

H S2: KnX, KnY, KnM, T, C, D,

K, HD: Y, M, S

16-bit instruction	32-bit instruction	Whether the pulse type
ZCP	DZCP	no
ZCPP	DZCPP	Yes

3.3.6 ZCPP

ZCP 16-bit pulse instruction, see ZCP instruction.

3.3.7 DZCP

ZCP 32-bit instructions, see ZCP instruction.

3.3.8 DZCPP

ZCP 32-bit pulse instruction, see ZCP instruction.

3.3.9 MOV

Instructions:

MOV 16-bit instructions, the device is transmitting the content to other devices instructions.

Operands:

S1: KnX, KnY, KnM, KnS, T, C, D, K, H, V, Z, LV, DT, @

D: KnY, KnM, KnS, T, C, D, K, H, V, Z, LV, DT

16-bit instruction	32-bit instruction	Whether the pulse type
MOV	DMOV	no
MOVP	DMOVP	Yes

3.3.10 MOVP

MOV's 16-bit pulse instruction, see MOV instruction.

3.3.11 DMOV

MOV 32-bit instructions, see MOV instruction.

3.3.12 DMOVP

MOV's 32-bit pulse instruction, see MOV instruction.

3.3.13 SMOV

Instructions:

SMOV instruction in number of bits (4 bits) allocated synthesized data.

Transmission source S and the transfer destination is first converted to the BCD of 4 bits, then the number of bits in S m1 m2 lower portion (bits starting from 1) incorporated into the starting position at D, n, and finally converts the data into the BIN and the combined D to save in.

Operands:

S: KnX, KnY, KnM, KnS, T, C, D, K, H, V, Z, LV, DT, @

m1: K, H

m2: K, H

D: KnY, KnM, KnS, T, C, D, V, Z, LV, DT

n : K, H

3.3.14 SMOVP

SMOV pulse execution command, see SMOV instruction.

3.3.15 CML

Instructions:

CML is a 16-bit instruction, based on the data in units of bits and inverts the copy storage. Operands:

S: KnX, KnY, KnM, KnS, T, C, D, K, H, V, Z, LV, DT, @

D: KnY, KnM, KnS, T, C, D, K, H, V, Z, LV, DT

16-bit instruction	32-bit instruction	Whether the pulse type
CML	DCML	no
CMLP	DCMLP	Yes

3.3.16 CMLP

CML, 16 pulse instruction, see CMP instruction.

3.3.17 DCML

CML 32-bit instructions, see instruction CML.

3.3.18 DCMLP

CML, 32 pulse instruction, see CMP instruction.

3.3.19 BMOV

Instructions:

BMOV instruction data specifying a plurality of points bulk copy (many). Operands:

S: KnX, KnY, KnM, KnS, T, C, D, K, H, V, Z, LV, DT, @

D: KnY, KnM, KnS, T, C, D, K, H, V, Z, LV, DT

n: K, H

3.3.20 BMOVP

BMOV pulse execution command, see BMOV instruction.

3.3.21 FMOV

Instructions:

FMOV 16-bit instructions, transmitting the same data to the destination device specified in points (one to many). Operands:

S: KnX, KnY, KnM, KnS, T, C, D, K, H, V, Z, LV, DT, @

D: KnY, KnM, KnS, T, C, D, K, H, V, Z, LV, DT

n: K, H

16-bit instruction	32-bit instruction	Whether the pulse type
FMOV	DFMOV	no
FMOVP	DFMOVP	Yes

3.3.22 FMOVP

FMOV 16-bit pulse instruction, see FMOV instruction.

3.3.23 DFMOV

FMOV 32-bit instructions, see FMOV instruction.

3.3.24 DFMOVP

FMOV 32-bit pulse instruction, see FMOV instruction.

3.3.25 XCH

Instructions:

XCH instruction data exchanged two devices. Operands:

S: KnX, KnY, KnM, KnS, T, C, D, K, H, V, Z, LV, DT, @

D: KnY, KnM, KnS, T, C, D, K, H, V, Z, LV, DT

n: K, H

16-bit instruction	32-bit instruction	Whether the pulse type
XCH	DXCH	no
XCHP	DXCHP	Yes

3.3.26 XCHP

See 3.3.25 XCH instruction.

3.3.27 DXCH

See 3.3.25 XCH instruction.

3.3.28 DXCHP

See 3.3.25 XCH instruction.

3.3.29 BCD

Instructions:

BCD instruction is transmitted after converting BIN (2 binary number) data into BCD (10 decimal) data. (Mainly used in tape

Output digital display 7-segment BCD

code) Operands:

S: KnX, KnY, KnM, KnS, T, C, D, K, H, V, Z, LV, DT, @

D: KnY, KnM, KnS, T, C, D, K, H, V, Z, LV, DT

16-bit instruction	32-bit instruction	Whether the pulse type
BCD	DBCD	no
BCDP	DBCDP	Yes

3.3.30 BCDP

See 3.3.29 BCD instruction.

3.3.31 DBCD

See 3.3.29 BCD instruction.

3.3.32 DBCDP

See 3.3.29 BCD instruction.

3.3.33 BIN

Instructions:

BIN instruction is transmitted after converting the BCD (10 decimal) data into BIN (hexadecimal number 2) data. (Mainly used in a digital switch input)

Operands:

S: KnX, KnY, KnM, KnS, T, C, D, K, H, V, Z, LV, DT, @

D: KnY, KnM, KnS, T, C, D, K, H, V, Z, LV, DT

16-bit instruction	32-bit instruction	Whether the pulse type
BIN	DBIN	no
BINP	DBINP	Yes

3.3.34 BINP

See 3.3.33 BIN command.

3.3.35 DBIN

See 3.3.33 BIN command.

3.3.36 DBINP

See 3.3.33 BIN command.

3.4 Jump circulation category

3.4.1 LBL

Instructions:

Here is a subroutine labeled and indicate the name of the subroutine.

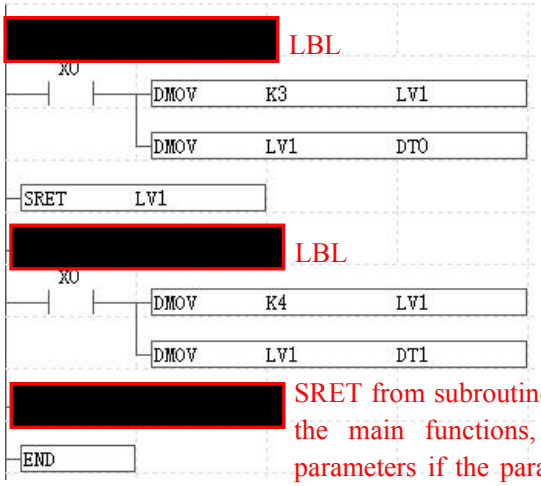
Operands:

S: @ sub_func, ln

Programming example:

1, Basic PLC subroutine call

```
1   runtask 1,SUB_Func
2   while 1
3       local a
4       delay(1000)
5       call SUB_CAL
6       a = return
7       trace "a = " a
8   wend
9   END
10  SUB SUB_Func()
11      WHILE 1
12          local b
13          delay(1000)
14          call SUB_CAL1
15          b = return
16          trace "b = " b
17      WEND
18  END SUB
```



SRET from subroutine returns to the main functions, additional parameters if the parameter as a return value.

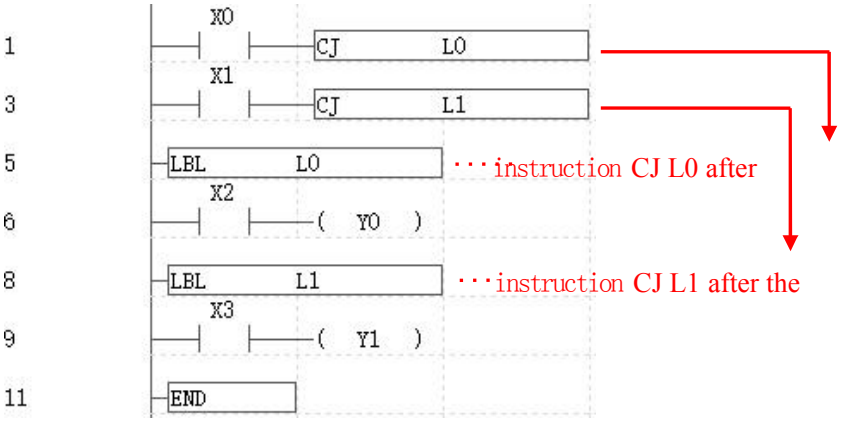
3.4.2 CJCJP

Instructions:

Make CJ, CJP start instruction to the program order of instructions is not executed until the flag LBL. Operands:

S: Ln

Programming example:



3.4.3 CJEND CJPEND

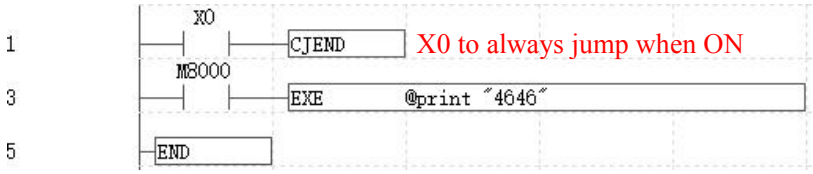
Instructions:

Make CJEND, CJPEND sequential program instruction to an instruction to start up a program END flag is not performed. Operands:

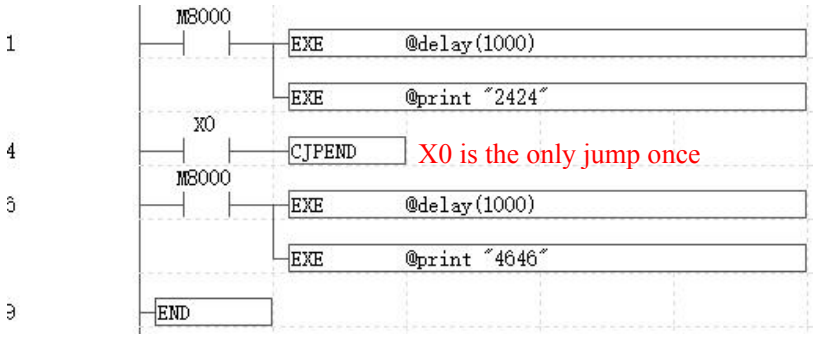
no

Programming example:

1, CJEND



2, CJPEND



3.4.4 CALL CALLP

Instructions:

Call instruction, you can call the global process of BASIC, or LBL-defined procedures. Operands:

This command can pass parameters(Parameter must be 32 Bit floating-point numbers, such as: global variables, LV, DT), Parameter or parameters are automatically passed LV register of the called procedure.

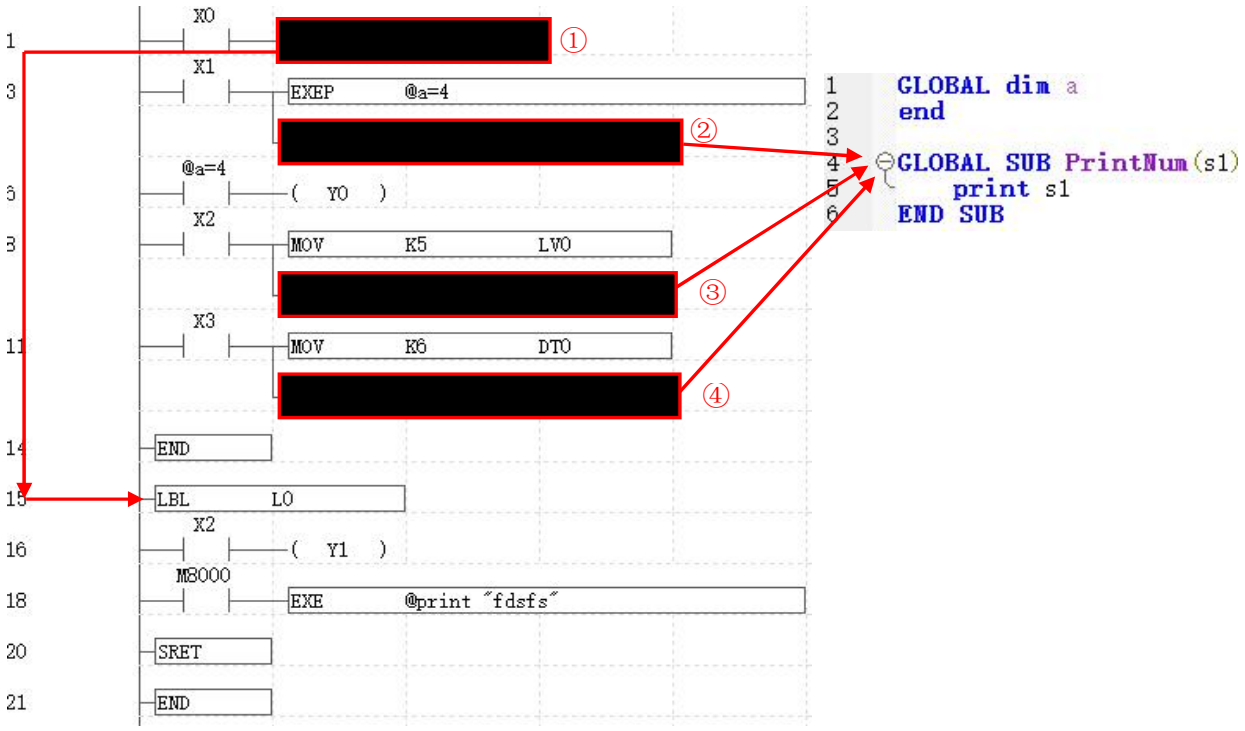
Parameters are automatically passed in 32-bit floating-point format. Programming example:

① PLC Functions for the call;

② BASIC is calling a global process parameters as global variables;

③ invoking BASIC global process parameter is the LV;

④ to call the global process of BASIC, parameters DT;



3.4.5 SRET

Instructions:

Execution of the main program After the CALL instruction, a jump to a subroutine, then returns to the main SRET instruction. Operands:

Can bring the return value, the return value is a floating point format, the caller acquired return value RETURN instruction. Programming example:

See LBL instruction.

3.4.6 IRET

Instructions:

IRET instruction the interrupt subroutine returns to the main program. Operands:

no.

3.4.7 EI

Instructions:

EI instructions cause the programmable interrupt controller to permission state, the state of the programmable controller typically disable interrupts. Operands:

no.

3.4.8 DI

Instructions:

DI instructions cause a programmable controller to a state allowing the interrupt disable interrupts. Operands:

no.

3.4.9 FEND

Instructions:

FEND instruction indicates the end of the main program of instruction.

carried out After FEND instruction, executes the same instruction END output processing, input processing, refresh the watchdog timer, and then returns to the program step 0. You need to use this command when writing subroutines and interrupt routines.

Operands:

no.

3.4.10 WDT

Instructions:

WDT instruction can refresh the watchdog timer.

You can set the watchdog timer time by setting D8000. Operands:

no.

3.4.11 FOR

Instructions:

From FOR instruction to the program run repeatedly between the NEXT instruction specified number of times. Operands:

S: KnX, KnY, KnM, KnS, T, C, D, K, H, V, Z, LV, DT, @

Programming example:

3.4.12 NEXT

Instructions:

From FOR instruction to the program run repeatedly between the NEXT instruction specified number of times. Operands:

no.

3.4.13 END

Instructions:

END instruction indicates the end of the instructions of the program.

Operands:

no.

3.5 Operation Instructions

3.5.1 ADD

Instructions:

ADD instruction for adding two data, stores the result in the destination register. Operands:

S1: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

S2: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

D: KnY, KnM, KnS, T, C, D, Z, V, LV, DT

16-bit instruction	32-bit instruction	Whether the pulse type
ADD	DADD	no
ADDP	DADDP	Yes

3.5.2 ADDP

See 3.5.1 ADD instruction.

3.5.3 DADD

See 3.5.1 ADD instruction.

3.5.4 DADDP

See 3.5.1 ADD instruction.

3.5.5 SUB

Instructions:

The SUB instruction subtracts two data, stores the result in the destination register. Operands:

S1: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

S2: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

D: KnY, KnM, KnS, T, C, D, Z, V, LV, DT

16-bit instruction	32-bit instruction	Whether the pulse type
SUB	DSUB	no
SUBP	DSUBP	Yes

3.5.6 SUBP

See 3.5.5 SUB instruction.

3.5.7 DSUB

See 3.5.5 SUB instruction.

3.5.8 DSUBP

See 3.5.5 SUB instruction.

3.5.9 MUL

Instructions:

MUL instruction multiplies two data, stores the result in the destination register. Operands:

S1: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

S2: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

D: KnY, KnM, KnS, T, C, D, Z, V, LV, DT

16-bit instruction	32-bit instruction	Whether the pulse type
MUL	DMUL	no
MULP	DMULP	Y

3.5.10 MULP

See 3.5.10 MUL instruction.

3.5.11 DMUL

See 3.5.10 MUL instruction.

3.5.12 DMULP

See 3.5.10 MUL instruction.

3.5.13 DIV

Instructions:

DIV instruction two division data, stores the result in the destination register.

When the 16-bit instruction, representing 32-bit result, the low 16-bit storage provider, high I 16-bit memory.

When the 32-bit instruction, representing 64-bit result, the low 32-bit storage business, the high 32-bit remainder is stored. Operands:

S1: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

S2: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

D: KnY, KnM, KnS, T, C, D, Z, V, LV, DT

16-bit instruction	32-bit instruction	Whether the pulse type
DIV	DDIV	no
DIVP	DDIVP	Yes

3.5.14 DIVP

See 3.5.13 DIV instruction.

3.5.15 DDIV

See 3.5.13 DIV instruction.

3.5.16 DDIVP

See 3.5.13 DIV instruction.

3.5.17 INC

Instructions:

INC instructions cause the device numerical data plus 1.

Continuous operation instruction is executed every calculation cycle plus 1 operation.

The rising edge of the pulse-type instruction execution plus 1 for each trigger operation. Operands:

D: KnY, KnM, KnS, T, C, D, Z, V, LV, DT

16-bit instruction	32-bit instruction	Whether the pulse type
INC	DINC	no
INCP	DINCP	Yes

3.5.18 INCP

See 3.5.17 INC instruction.

3.5.19 DINC

See 3.5.17 INC instruction.

3.5.20 DINCP

See 3.5.17 INC instruction.

3.5.21 DEC

Instructions:

DEC instructions cause the device data value minus 1.

Continuous computation cycle calculation is executed for each instruction arithmetic subtraction.

Each trigger edge pulse type instruction execution operation decremented. Operands:

D: KnY, KnM, KnS, T, C, D, Z, V, LV, DT

16-bit instruction	32-bit instruction	Whether the pulse type
DEC	DDEC	no

DECP	**D**DECP	**Yes**

3.5.22 DECP

See 3.5.21 DEC instruction.

3.5.23 DDEC

See 3.5.21 DEC instruction.

3.5.24 DDECP

See 3.5.21 DEC instruction.

3.5.25 WAND

Instructions:

WAND instruction logically ANDs two values and stores the result in the destination register. Operands:

S1: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

S2: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

D: KnY, KnM, KnS, T, C, D, Z, V, LV, DT

16-bit instruction	32-bit instruction	Whether the pulse type
WAND	DWAND	no
WANDP	DWANDP	Yes

3.5.26 WANDP

See 3.5.25 WAND instruction.

3.5.27 DWAND

See 3.5.25 WAND instruction.

3.5.28 DWANDP

See 3.5.25 WAND instruction.

3.5.29 WOR

Instructions:

WOR instruction OR'ed two values and stores the result in the destination register. Operands:

S1: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

S2: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

D: KnY, KnM, KnS, T, C, D, Z, V, LV, DT

16-bit instruction	32-bit instruction	Whether the pulse type
WOR	DOR	no
WORP	DORP	Yes

3.5.30 WORP

See 3.5.25 WOR instruction.

3.5.31 DOR

See 3.5.25 WOR instruction.

3.5.32 DORP

See 3.5.25 WOR instruction.

3.5.33 WXOR

Instructions:

WXOR two instruction values logical exclusive OR operation, and stores the result in the destination register. Operands:

S1: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

S2: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

D: KnY, KnM, KnS, T, C, D, Z, V, LV, DT

16-bit instruction	32-bit instruction	Whether the pulse type
WXOR	DXOR	no
WXORP	DXORP	Y

3.5.34 WXORP

See 3.5.33 WXOR instruction.

3.5.35 DXOR

See 3.5.33 WXOR instruction.

3.5.36 DXORP

See 3.5.33 WXOR instruction.

3.5.37 NEG

Instructions:

NEG instruction obtaining a number of complement (the inverse value of the sign). Operands:

D: KnY, KnM, KnS, T, C, D, Z, V, LV, DT

16-bit instruction	32-bit instruction	Whether the pulse type
NEG	DNEG	no
NEGP	DNEGP	Yes

3.5.38 NEGP

See 3.5.37 NEG instruction.

3.5.39 DNEG

See 3.5.37 NEG instruction.

3.5.40 DNEGP

See 3.5.37 NEG instruction.

3.6 Shift instruction

3.6.1 ROR

Instructions:

ROR instruction so that the data does not include the carry flag bit including Rotate Right. Operands:

D: KnY, KnM, KnS, T, C, D, Z, V, LV, DT

n: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

16-bit instruction	32-bit instruction	Whether the pulse type
ROR	DROR	no
RORP	DRORP	Yes

3.6.2 RORP

See 3.6.1 ROR instruction.

3.6.3 DROR

See 3.6.1 ROR instruction.

3.6.4 DRORP

See 3.6.1 ROR instruction.

3.6.5 ROL

Instructions:

ROL instruction so that the data does not include the carry flag including bitwise to the left. Operands:

D: KnY, KnM, KnS, T, C, D, Z, V, LV, DT

n: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

16-bit instruction	32-bit instruction	Whether the pulse type
ROL	DROL	no
ROLP	DROLP	Yes

3.6.6 ROLP

See 3.6.1 ROL instructions.

3.6.7 DROL

See 3.6.1 ROL instructions.

3.6.8 DROLP

See 3.6.1 ROL instructions.

3.6.9 RCR

Instructions:

ROL instruction so that the data comprises the carry flag bit including Rotate Right. Operands:

D: KnY, KnM, KnS, T, C, D, Z, V, LV, DT

n: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

16-bit instruction	32-bit instruction	Whether the pulse type
RCR	DRCR	no
RCRP	DRCRP	Yes

3.6.10 RCRP

See 3.6.9 RCR command.

3.6.11 DRCR

See 3.6.9 RCR command.

3.6.12 DRCRP

See 3.6.9 RCR command.

3.6.13 RCL

Instructions:

RCR comprising instructions cause the data including the carry flag bit to the left. Operands:

D: KnY, KnM, KnS, T, C, D, Z, V, LV, DT

n: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

16-bit instruction	32-bit instruction	Whether the pulse type
RCL	DRCL	no
RCLP	DRCLP	Yes

3.6.14 RCLP

See 3.6.13 RCL instruction.

3.6.15 DRCL

See 3.6.13 RCL instruction.

3.6.16 DRCLP

See 3.6.13 RCL instruction.

3.6.17 SFTR

Instructions:

SFTR starting instruction D n1 n2 bits right bit elements, start copying and n2 bits S bits starting from the upper element of D.

Operands:

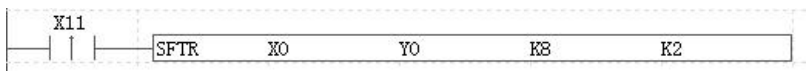
S: X, Y, M, S, T, C, D, @.

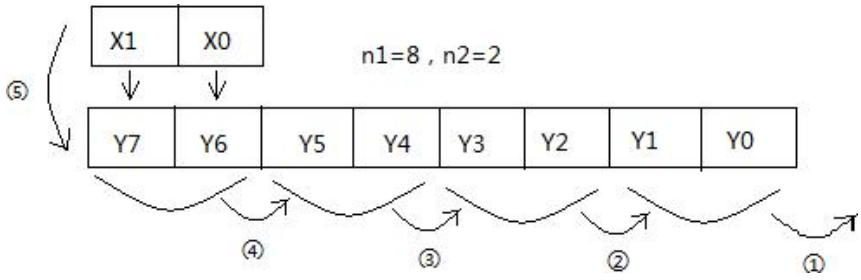
D: X, Y, M, S, T, C, D, @.

n1: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

n2: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

Programming example:





3.6.18 SFTRP

Instructions:

SFTR SFTRP is pulse type instructions.

3.6.19 SFTL

Instructions:

SFTR starting instruction D n1 n2 bits left bit elements, start copying and n2 bits S bits starting from the element D is low.

Operands:

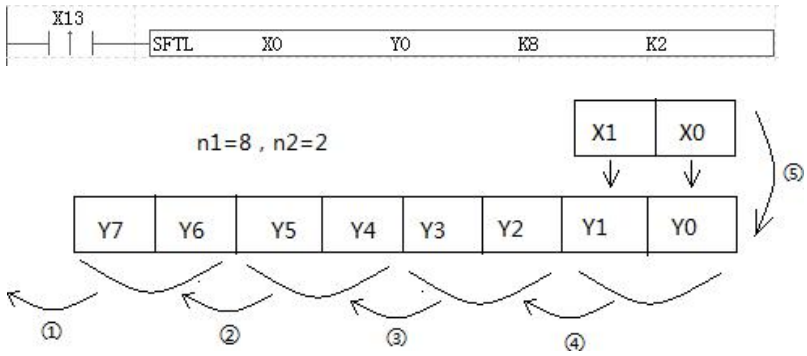
S: X, Y, M, S, T, C, D, @.

D: X, Y, M, S, T, C, D, @.

n1: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

n2: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

Programming example:



3.6.20 SFTLP

Instructions:

SFTLP is SFTL pulse type instructions.

3.6.21 WSFR

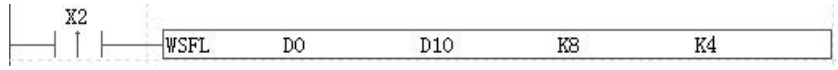
Instructions:

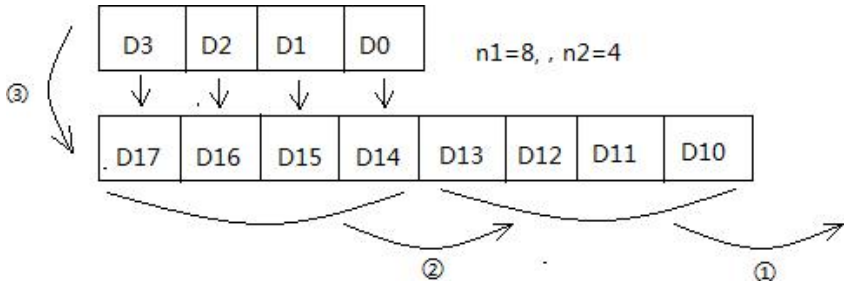
WSFR D starting instruction word elements n1 n2 bits right, and from the D n2-n1 n2-bit word to begin copying initiation + S elements.

Operands:

S: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV,
DT, @ D: KnY, KnM, KnS, T, C, D, Z, V, LV, DT
n1: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV,
DT, @
n2: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

Programming example:





3.6.22 WSFRP

Instructions:

WSFRP is WSFR pulse type instructions.

3.6.23 WSFL

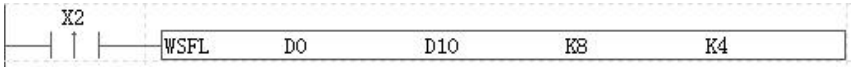
Instructions:

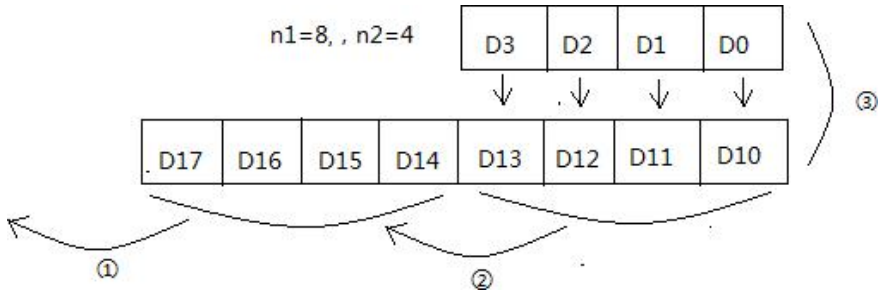
WSFR D starting instruction word elements left n1 n2 bits and n2 bits word begins copying starting from the elements S D.

Operands:

S: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV,
DT, @ D: KnY, KnM, KnS, T, C, D, Z, V, LV, DT
n1: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV,
DT, @
n2: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

Programming example:





3.6.24 WSFLP

Instructions:

WSFLP is WSFL pulse type instructions.

3.6.25 SFWR

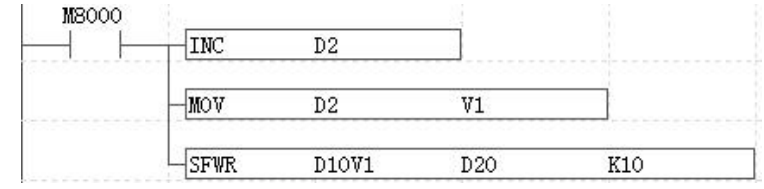
Instructions:

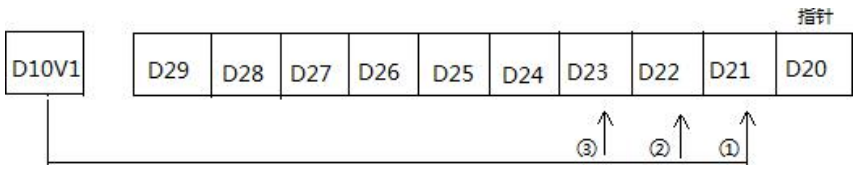
will The writing starting from the value S D address, the number n of "FIFO" queue, to a device as the first pointer, when executed, before adding the content of the pointer 1, after the designated S value means the contents of the write FIFO data series D by the pointer position specified. If the data queue is full, no subsequent data processing, and the flag M8022 turns ON.

Operands:

S: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV,
DT, @ D: KnY, KnM, KnS, T, C, D, Z, V, LV, DT
n1: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV,
DT, @

Programming example:





3.6.26 SFWRP

Instructions:

SFWRP is SFWR pulse type instructions.

3.6.27 SFRD

Instructions:

Is read out from the first item "FIFO" S to D of the queue, then the queue S literally right one word, the queue pointer is decremented. In a device as the first pointer, when executed, the first content of the pointer is decremented by 1, S means after the content of the designated value is written to FIFO data series D by the pointer position specified. If the pointer is already zero, the process is not the instruction operation, while the 0 flag M8020 is set.

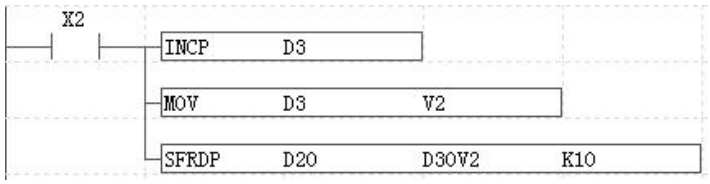
Operands:

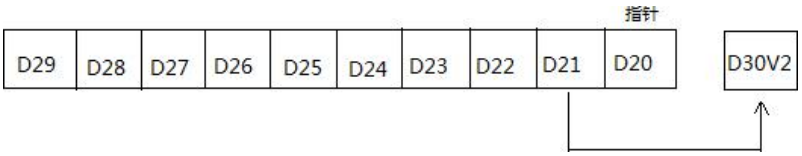
S: KnY, KnM, KnS, T, C, D, Z, V, LV, DT

D: KnY, KnM, KnS, T, C, D, Z, V, LV, DT n1: KnX,

KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

Programming example:





3.6.28 SFRDP

Instructions:

SFRDP is SFRD pulse type instructions.

3.7 Data processing instructions

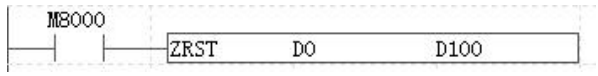
3.7.1 ZRST

Instructions:

ZRST instructions cause the device to specify command execution between two bulk reset. Operands:

S1: Y, M, S, T, C, D, Z, V, LV, DT
S2: Y, M, S, T, C, D, Z, V, LV, DT

Programming example:



3.7.2 ZRSTP

Instructions:

ZRSTP is ZRST pulse type instructions.

3.7.3 DECO

Instructions:

DECO command according to the value of the source operand to the destination operand of the number of the corresponding bit.

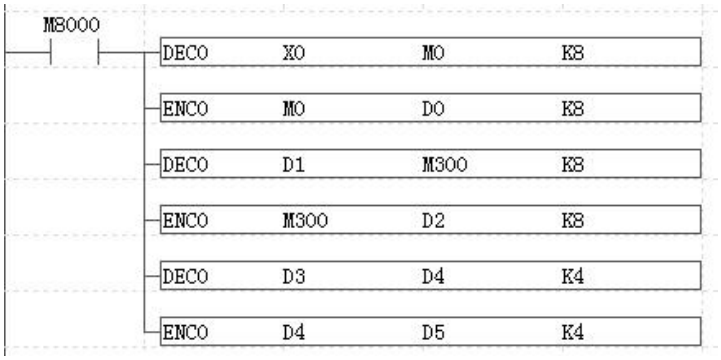
The destination operand is a word device or bit device, n ranges from 1-4 or 1-8, respectively, are not treated as 0. Operands:

S: X, Y, M, S, T, C, D, Z, V, LV, DT, K, H

D: Y, M, S, T, C, D, Z, V, LV, DT

n: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

Programming example:



3.7.4 DECOP

Instructions:

DECOP DECO is pulse type instructions.

3.7.5 ENCO

Instructions:

ENCO instruction decoding result DECO encoding.

The source operand is a word device or bit device, n ranges from 1-4 or 1-8, respectively, are not treated as 0. Operands:

S: X, Y, M, S, T, C, D, Z, V, LV, DT, K, H

D: Y, M, S, T, C, D, Z, V, LV, DT

n: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

Programming example:

See instructions 3.7.3DECO

3.7.6 ENCOP

Instructions:

ENCOP is ENCO pulse type instructions.

3.7.7 SUM

Instructions:

SUM is calculated as the total number of instruction "1" bits in the data specified in the device.

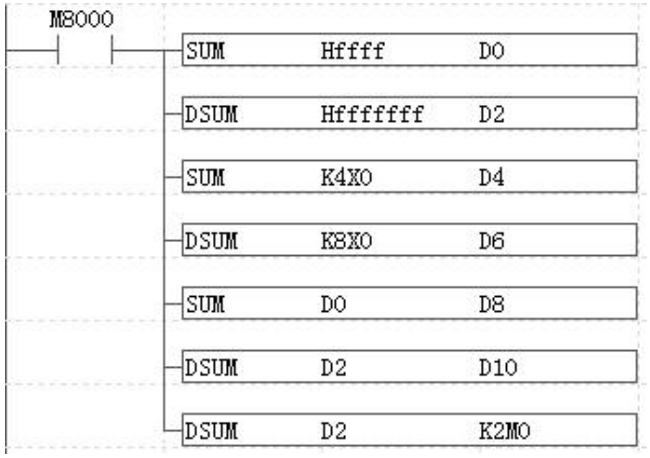
Operands:

S: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT,

@ D: KnY, KnM, KnS, T, C, D, Z, V, LV, DT

16-bit instruction	32-bit instruction	Whether the pulse type
SUM	**D**SUM	no
SUMP	**D**SUMP	Yes

Programming example:



3.7.8 SUMP

See 3.7.7 SUM command.

3.7.9 DSUM

See 3.7.7 SUM command.

3.7.10 DSUMP

See 3.7.7 SUM command.

3.7.11 BON

Instructions:

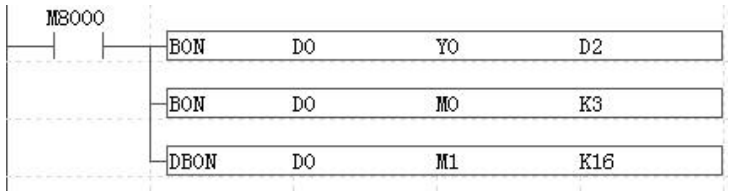
SON instruction checks the device specified bit position or ON to OFF.

Operands:

S: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV,
DT, @ D: Y, M, S, T, C, D, @.
n: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

16-bit instruction	32-bit instruction	Whether the pulse type
SON	**D**SON	no
SONP	**D**SONP	Yes

Programming example:



3.7.12 BONP

See 3.7.11 SON instructions.

3.7.13 DBON

See 3.7.11 SON instructions.

3.7.14 DBONP

See 3.7.11 SON instructions.

3.7.15 MEAN

Instructions:

MEAN instruction averaging the data. Operands:

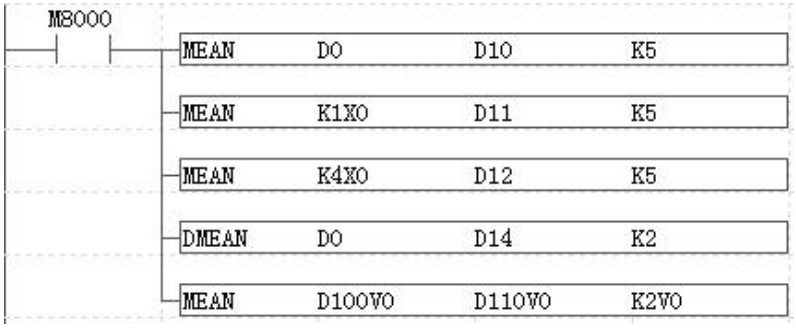
S: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

D: KnY, KnM, KnS, T, C, D, Z, V, LV, DT,

n: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

16-bit instruction	32-bit instruction	Whether the pulse type
MEAN	DMEAN	no
MEANP	DMEANP	Yes

Programming example:



3.7.16 MEANP

See 3.7.15 MEAN instruction.

3.7.17 DMEAN

See 3.7.15 MEAN instruction.

3.7.18 DMEANP

See 3.7.15 MEAN instruction.

3.7.19 ANS

3.7.20 ANR

3.7.21 ANRP

3.7.22 SQR

Instructions:

Computing the square root

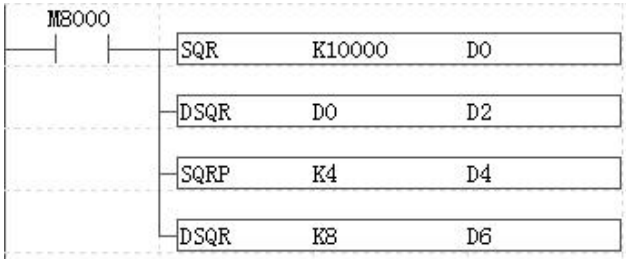
SQR instruction. Operands:

S: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

D: KnY, KnM, KnS, T, C, D, Z, V, LV, DT,

16-bit instruction	32-bit instruction	Whether the pulse type
SQR	**D**SQR	no
SQRP	**D**SQRP	Yes

Programming example:



3.7.23 SQRP

See 3.7.22 MEAN instruction.

3.7.24 DSQR

See 3.7.22 MEAN instruction.

3.7.25 DSQRP

See 3.7.22 MEAN instruction.

3.7.26 FLT

Instructions:

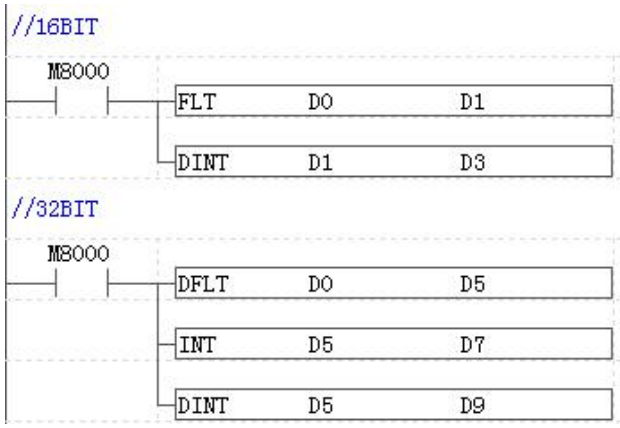
FLT instruction BIN integer value to a binary floating-point (real) instruction. Operands:

S: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

D: KnY, KnM, KnS, T, C, D, Z, V, LV, DT,

16-bit instruction	32-bit instruction	Whether the pulse type
FLT	DFLT	no
FLTP	DFLTP	Yes

Programming example:



3.7.27 FLTP

See 3.7.26 FLT instruction.

3.7.28 DFLT

See 3.7.26 FLT instruction.

3.7.29 DFLTP

See 3.7.26 FLT instruction.

3.7.30 SWAP

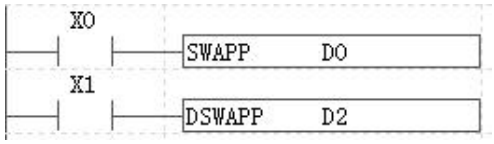
Instructions:

SWAP instruction swaps the high data word 8 and a lower 8-bit instruction. Operands:

S: KnY, KnM, KnS, T, C, D, Z, V, LV, DT

16-bit instruction	32-bit instruction	Whether the pulse type
SWAP	DSWAP	no
SWAPP	DSWAPP	Yes

Programming example:



3.7.31 SWAPP

See 3.7.30 SWAP instruction.

3.7.32 DSWAP

See 3.7.30 SWAP instruction.

3.7.33 DSWAPP

See 3.7.30 SWAP instruction.

3.8 Floating point instructions

3.8.1 DECMP

Instructions:

DECM instruction for comparing two data, the output bit soft element (3 points) in. Operands:

S1: KnX, KnY, KnM, KnS, T, C, D, K, H, LV, DT, @ S2: KnY, KnM, KnS, T, C, D, Z, V, LV, DT

D: Y, M, S, T, C, D, @.

16-bit instruction	32-bit instruction	Whether the pulse type
-	DECMP	no
-	DECMPP	Yes

3.8.2 DECMPP

See 3.8.1 DECM instruction.

3.8.3 DEZCP

Instructions:

The upper and lower DEZCP instruction point range and comparison data (binary floating-point number), and outputs (3:00) in place in the device according to the result of the instruction.

Operands:

S1: KnX, KnY, KnM, KnS, T, C, D, K, H, LV, DT, @

S2: KnX, KnY, KnM, KnS, T, C, D, K, H, LV, DT, @

S: KnY, KnM, KnS, T, C, D, Z, V, LV, DT

D: Y, M, S, T, C, D, @.

16-bit instruction	32-bit instruction	Whether the pulse type
-	DEZCP	no
-	DEZCPP	Yes

3.8.4 DEZCPP

See 3.8.3 DEZCP instruction.

3.8.5 DEBCD

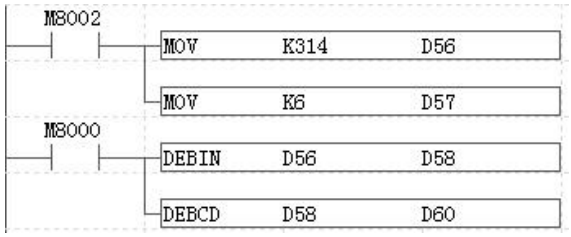
Instructions:

DEBCD instruction converts the soft element 10 into a binary floating-point decimal floating-point number. Operands:

S: KnX, KnY, KnM, KnS, T, C, D, K, H, LV, DT, @

D: KnX, KnY, KnM, KnS, T, C, D, K, H, LV, DT

Programming example:



3.8.6 DEBCDP

Instructions:

DEBCD is DEBCDP pulse type instructions.

3.8.7 DEBIN

Instructions:

DEBIN instruction converting device 10 in decimal floating-point number to a binary floating-point number. Operands:

S: KnX, KnY, KnM, KnS, T, C, D, K, H, LV, DT, @

D: KnX, KnY, KnM, KnS, T, C, D, K, H, LV, DT

Programming example:

See DEBCD instructions.

3.8.8 DEBINP

Instructions:

DEBIN is DEBINP pulse type instructions.

3.8.9 DEADD / DEADDP / DESUB / DESUBP / DEMUL / DEMULP / DEDIV / DEDIVP / DESQR / DESQ RP

Instructions:

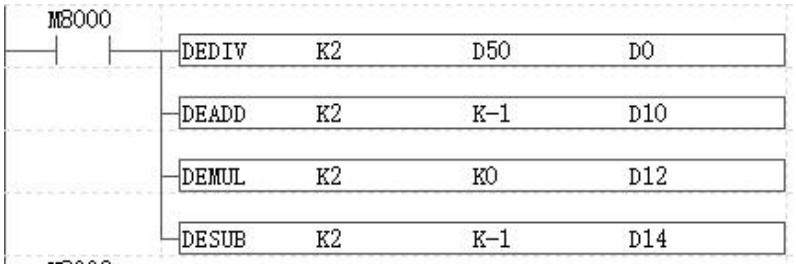
The set of instructions to four floating point arithmetic.

32- floating-point instructions	Whether the pulse type	Operand species	Instruction Name
DEADD	no	S1: T, C, D, K, H, LV, DT, @ S2: T, C, D, K, H, LV, DT, @ D: T, C, D, LV, DT	Floating point adder
DEADDP	Yes		
DESUB	no		Floating-point subtraction
DESUBP	Yes		
DEMUL	no		Floating-point multiply
DEMULP	Yes		
DEDIV	no		Floating-point division
DEDIVP	Yes		
DESQR	no		Float Evolution
DESQRP	Yes		

Note: The parameters are floating point instructions to the floating point conversion constant (all included in the mentioned Mitsubishi
RAD / DEG / EXP / LOGE).

When floating point data registers used in the instruction, the data value in the floating-point register or may be automatically converted into values of the constants.

Programming example:



3.8.10 DEADD

See 3.8.9.

3.8.11 DEADDP

See 3.8.9.

3.8.12 DESUB

See 3.8.9.

3.8.13 DESUBP

See 3.8.9.

3.8.14 DEMUL

See 3.8.9.

3.8.15 DEMULP

See 3.8.9.

3.8.16 DEDIV

See 3.8.9.

3.8.17 DEDIVP

See 3.8.9.

3.8.18 DESQR

See 3.8.9.

3.8.19 DESQRP

See 3.8.9.

3.8.20 INT

Instructions:

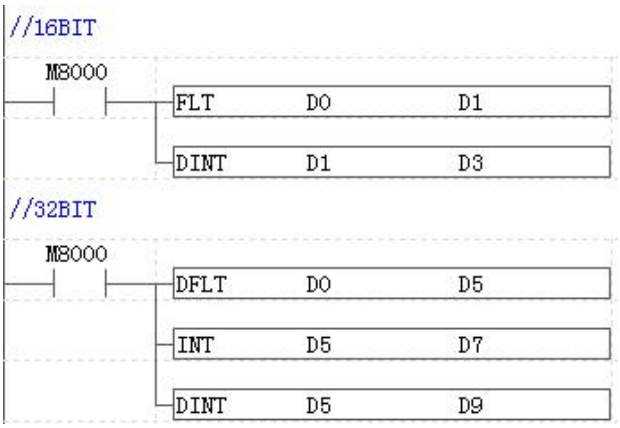
The INT instruction binary floating-point number, is converted into integer instructions BIN general data in the form of a programmable controller. Operands:

S: T, C, D, K, H, LV, DT, @

D: KnY, KnM, KnS, T, C, D, K, H, LV, DT ,, Z

16-bit instruction	32-bit instruction	Whether the pulse type
INT	DINT	no
INTP	DINTP	Yes

Programming example:



3.8.21 INTP

See 3.8.20 INT.

3.8.22 DINT

See 3.8.20 INT.

3.8.23 DINTP

See 3.8.20 INT.

3.8.24 DSIN

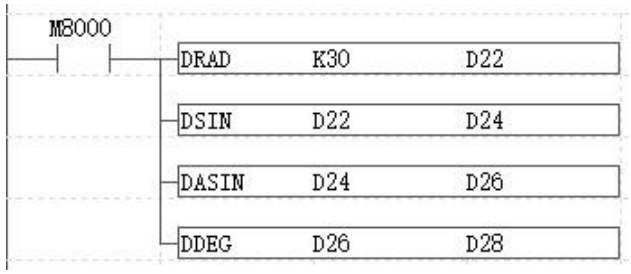
Instructions:

DSIN seek command angle (RAD) SIN worth of instructions. Operands:

S1: T, C, D, LV, DT,

@ S2: T, C, D, LV, DT

Programming example:



3.8.25 DSINP

Instructions:

DSIN instruction pulse form.

3.8.26 DCOS

Instructions:

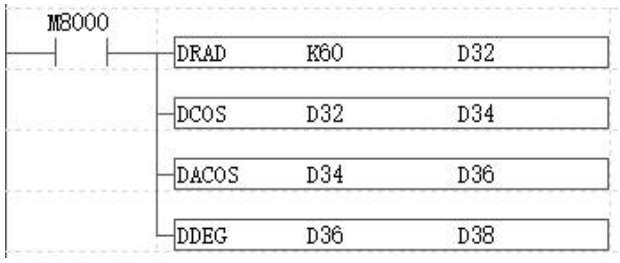
DCOS seek command angle (RAD) COS worth of instructions. Operands:

S1: T, C, D, LV, DT,

@ S2: T, C, D, LV,

DT

Programming example:



3.8.27 DCOSP

Instructions:

DCOS command pulse form.

3.8.28 DTAN

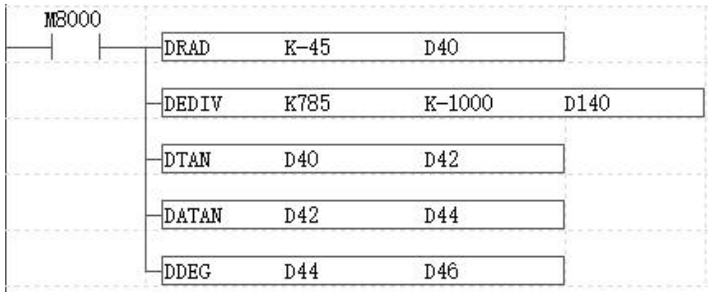
Instructions:

DTAN seek command angle (RAD) worth of TAN instruction. Operands:

S1: T, C, D, LV, DT,

@ S2: T, C, D, LV, DT

Programming example:



3.8.29 DTANP

Instructions:

DTAN instruction pulse form.

3.8.30 DEMOV

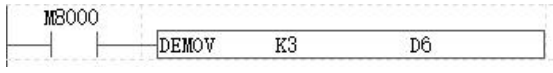
Instructions:

DEMOV transmit binary floating point instruction. Operands:

S1: T, C, D, LV, DT, @, K,

H S2: T, C, D, LV, DT

Programming example:



3.8.31 DEMOVP

Instructions:

DEMOV instruction pulse form.

3.8.32 DEXP

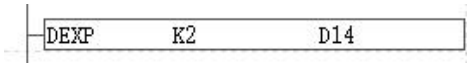
Instructions:

DEXP instructions are exponentiation e (2.71828) as the base. Operands:

S1: T, C, D, LV, DT, @, K,

H S2: T, C, D, LV, DT

Programming example:



3.8.33 DEXPP

Instructions:

DEXPP instruction pulse form.

3.8.34 DLOGE

Instructions:

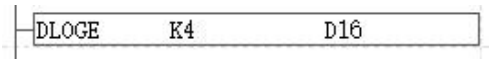
DLOGE natural logarithmic operation instruction execution.

Operands:

S1: T, C, D, LV, DT, @, K,

H S2: T, C, D, LV, DT

Programming example:



3.8.35 DLOGEP

Instructions:

DLOGEP instruction pulse form.

3.8.36 DLOG

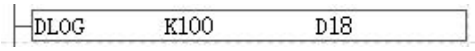
Instructions:

DLOG instruction execution common logarithm (base 10 logarithm) operation. Operands:

S1: T, C, D, LV, DT, @, K, H

S2: T, C, D, LV, DT

Programming example:



3.8.37 DLOGP

Instructions:

DLOG command pulse form.

3.8.38 DASIN

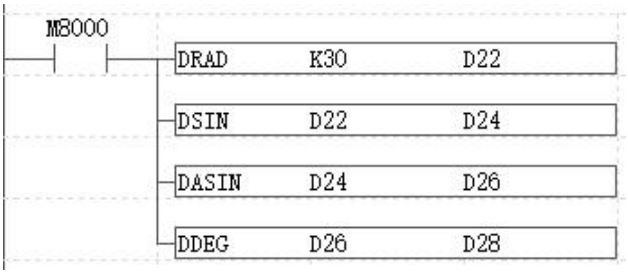
Instructions:

SIN-1 DASIN instruction execution operation. Operands:

S1: T, C, D, LV, DT, K,

@ S2: T, C, D, LV, DT

Programming example:



3.8.39 DASINP

Instructions:

DASIN instruction pulse form.

3.8.40 DACOS

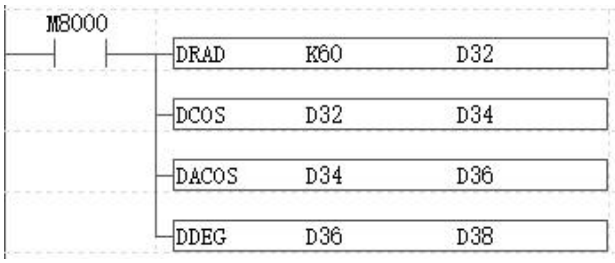
Instructions:

COS-1 DACOS instruction execution operation. Operands:

S1: T, C, D, LV, DT, K,

@ S2: T, C, D, LV, DT

Programming example:



3.8.41 DACOSP

Instructions:

DACOS instruction pulse form.

3.8.42 DATAN

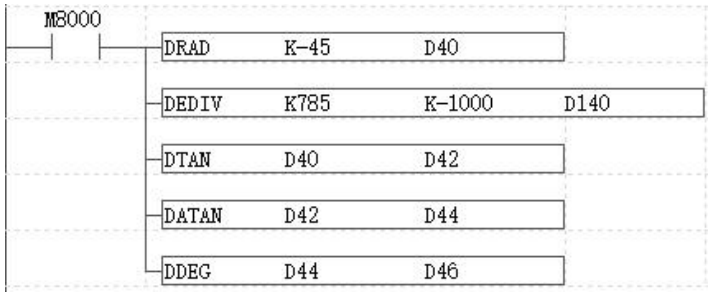
Instructions:

TAN-1 DATAN instruction execution operation. Operands:

S1: T, C, D, LV, DT, K,

@ S2: T, C, D, LV, DT

Programming example:



3.8.43 DATANP

Instructions:

DATAN command pulse form.

3.8.44 DRAD

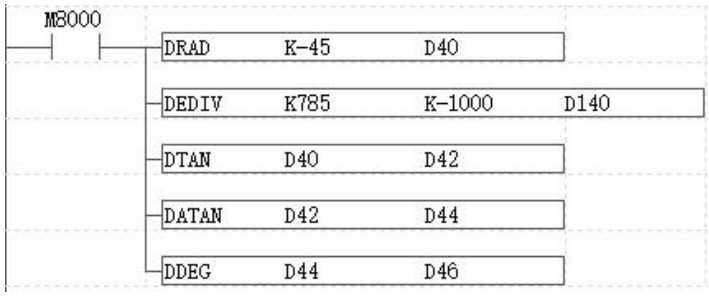
Instructions:

DRAD angle instruction arithmetic converted to radians. Operands:

S1: T, C, D, LV, DT, K, @

S2: T, C, D, LV, DT

Programming example:



3.8.45 DRADP

Instructions:

DRAD command pulse form.

3.8.46 DDEG

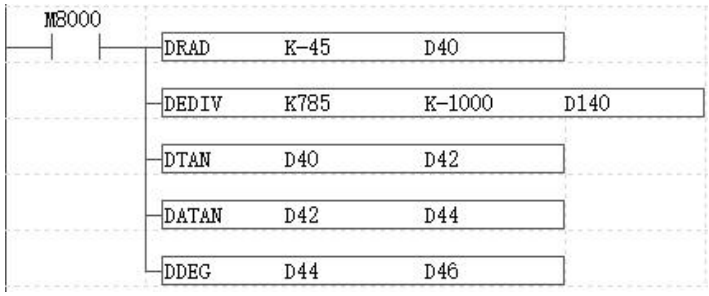
Instructions:

DDEG operation instruction is converted into an angle in radians. Operands:

S1: T, C, D, LV, DT, K,

@ S2: T, C, D, LV, DT

Programming example:



3.8.47 DDEGP

Instructions:

DDEG instruction pulse form.

3.8.48 DSINH

Instructions:

DSINH hyperbolic sine operation instructions. Operands:

S1: T, C, D, LV, DT, K,

@ S2: T, C, D, LV, DT

Programming example:

3.8.49 DSINHP

Instructions:

DSINH instruction pulse form.

3.8.50 DCOSH

Instructions:

DCOSH cosh computation instruction. Operands:

S1: T, C, D, LV, DT, K,

@ S2: T, C, D, LV, DT

Programming example:

3.8.51 DCOSHP

Instructions:

DCOSH instruction pulse form.

3.8.52 DTANH

Instructions:

DTANH hyperbolic tangent operation instructions. Operands:

S1: T, C, D, LV, DT, K,

@ S2: T, C, D, LV, DT

Programming example:

3.8.53 DTANHP

Instructions:

DTANH instruction pulse form.

3.9 Other instructions

3.9.1 BASE

Instructions:

BASE instruction specifies a movement of the axes involved.

Operands:

S1: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

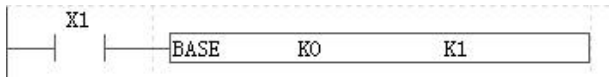
S2: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

.....

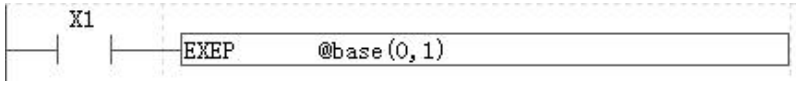
S8: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

Programming example:

1, directly BASE directive PLC instruction:



2, call BASE Basic instruction in the PLC:



3.9.2 MOVE

Instructions:

MOVE instruction for the relative movement of the linear interpolation command.

Operands:

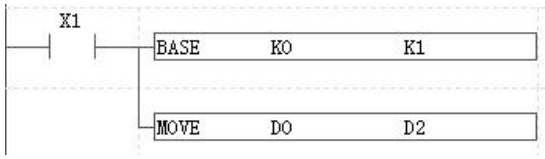
S1: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

S2: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

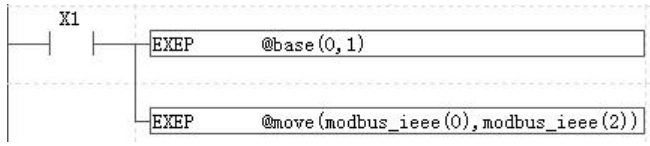
.....

S8: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

When using a word operand registers, floating point registers required value assignment instruction. Example Programming: 1, directlyPLC instruction MOVE instruction:



2, call the MOVE command in the PLC in the Basic:



Note: Both of Call the MOVE command there is a difference. When using the MOVE command in the PLC, if not completed MOVE motion is disconnected X1, the MOVE movement to stop immediately; when using the MOVE command Basic, even if the movement did not complete MOVE is disconnected X1, the movement will continue until MOVE carry out.

3.9.3 MOVEABS

Instructions:

MOVEABS instructions linear interpolation absolute motion commands. Operands:

S1: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

S2: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

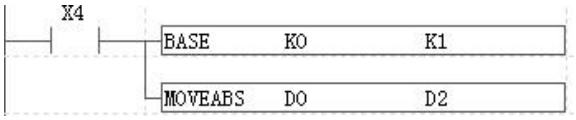
.....

S8: KnX, KnY, KnM, KnS, T, C, D, K, H, Z, V, LV, DT, @

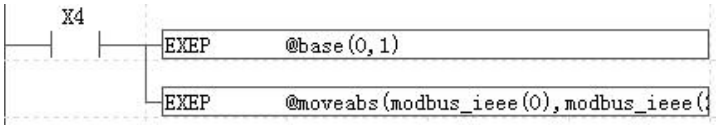
When using a word operand registers, floating point registers required value assignment instruction.

Programming example:

1, PLC instruction directly MOVEABS instructions:



2, call MOVEABS Basic instruction in the PLC:



Note: Both the call to MOVEABS instruction there is a difference. With the MOVE command.

3.9.4 VMOVE

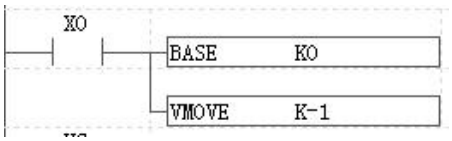
Instructions:

VMOVE instruction is continuous movement instruction, i.e., continuous motion in one direction. Operands:

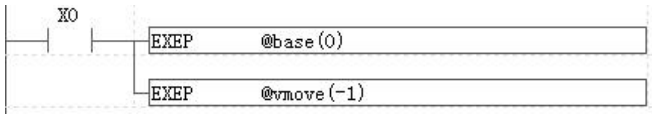
S: K (-1 negative direction / a positive direction) Programming

example:

1, PLC instruction directly VMOVE instructions:



2, call VMOVE Basic instruction in the PLC:



Note: Both the call to VMOVE instruction there is a difference. With the MOVE command.

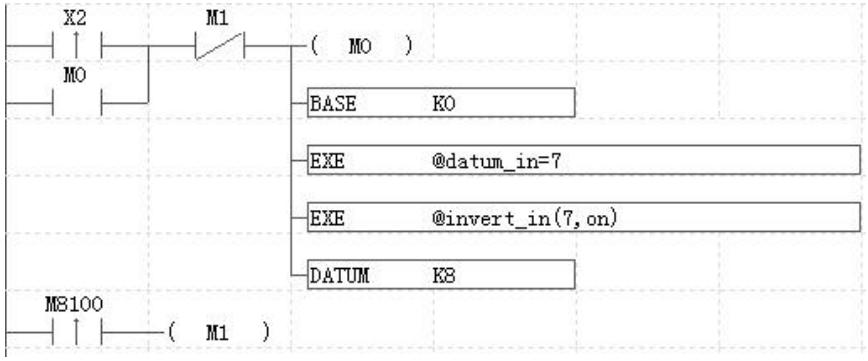
3.9.5 DATUM

Instructions:

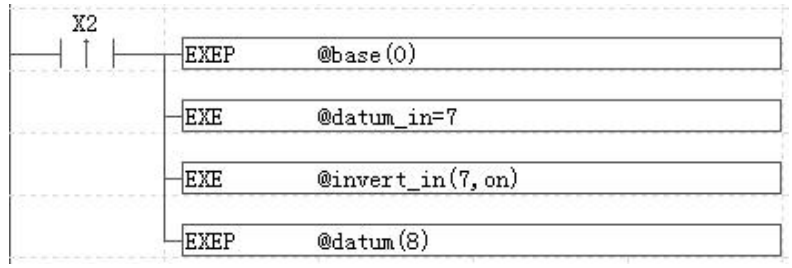
DATUM instruction is continuous movement instruction, i.e., continuous motion in one direction. Operands:

S: K (-1 negative direction / a positive direction) Programming example:

1, directly to the DATUM instruction PLC instruction:



2, call the DATUM Basic instruction in the PLC:



Note: Both the call to DATUM command there is a difference. With the MOVE command.

Chapter IV use PLC and BASIC

4.1 PLC versus BASIC Shared variables

Related register:

X Corresponding to IN

Y correspond OP

M correspond

MODBUS_BIT D

TABLE MODBUS_REG

DT corresponding to

the corresponding

@BASIC expression

4.2 PLC use BASIC command

Related instructions: EXE EXEP @MOVE ()

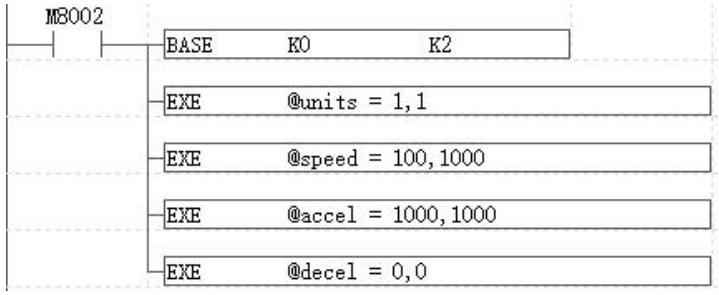
See EXE / EXEP instructions.

4.3 PLC use BASIC expression

Related register: @basicexpr

Programming example:

The following example demonstrates the use of PLC It defines the basic parameters axis.



4.4 PLC transfer BASIC Subroutine

Related command: CALL @basicfunc

See CALL / CALLP instruction.

4.5 BASIC transfer PLC Subroutine

Related instructions: LBL

@plcfunc CALL plcfunc

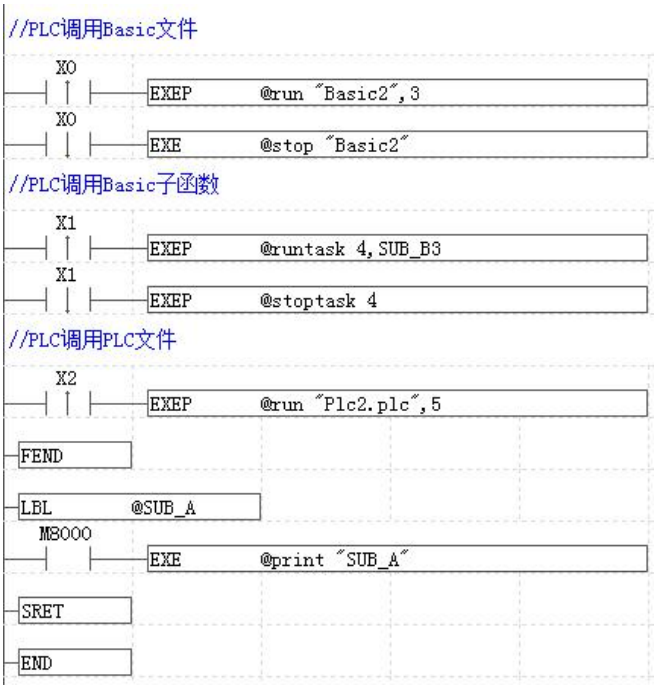
See LBL instruction.

4.6 BASIC start up PLC task

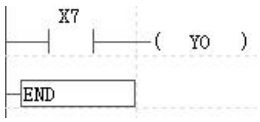
Related command: RUN, RUNTASK

can be realised Subroutine call each other and Basic Plc, and start each file. Programming example:

Plc1.plc file, set to run automatically



Plc2.plc file



Basic1.bas file

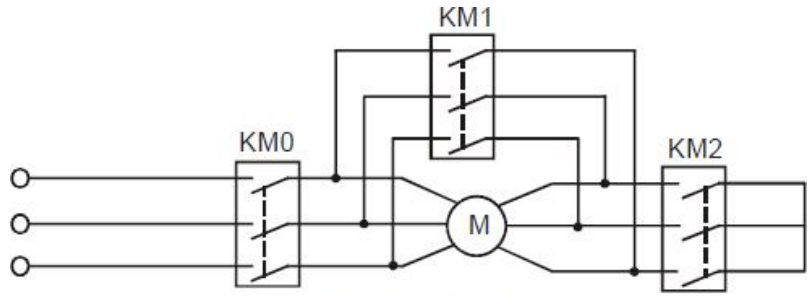
```
1   'Basic调用PLC文件
2   run "Plc1.plc",6
3   'Basic调用PLC函数
4   runtask 2,SUB_A
5   END
6
7   GLOBAL SUB SUB_B1()
8       print "SIB_B1"
9   END SUB
```

Basic2.bas file

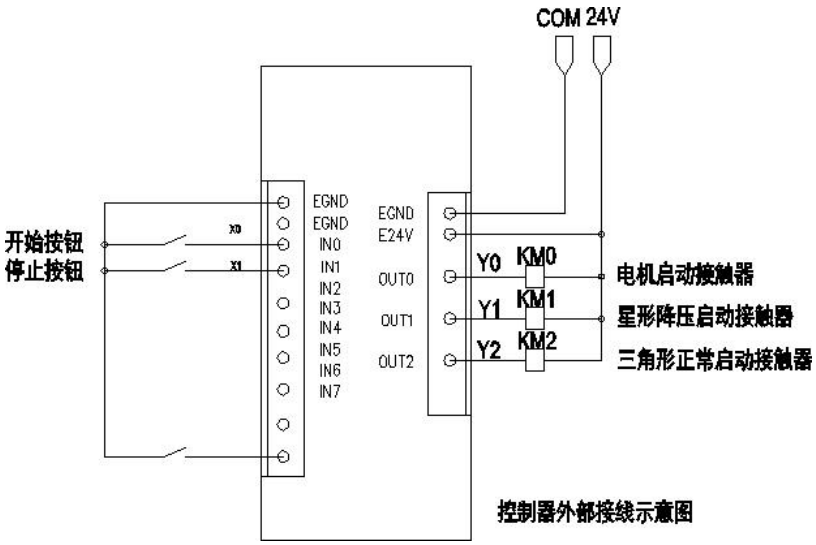
```
GLOBAL SUB SUB_B2()
    while 1
        print "SIB_B2"
        delay(1000)
    wend
END SUB
GLOBAL SUB SUB_B3()
    while 1
        print "SIB_B3"
        delay(1000)
    wend
END SUB
```

Chapter V Routine

5.1 Star Triangle buck starts



降压启动主电路示意图



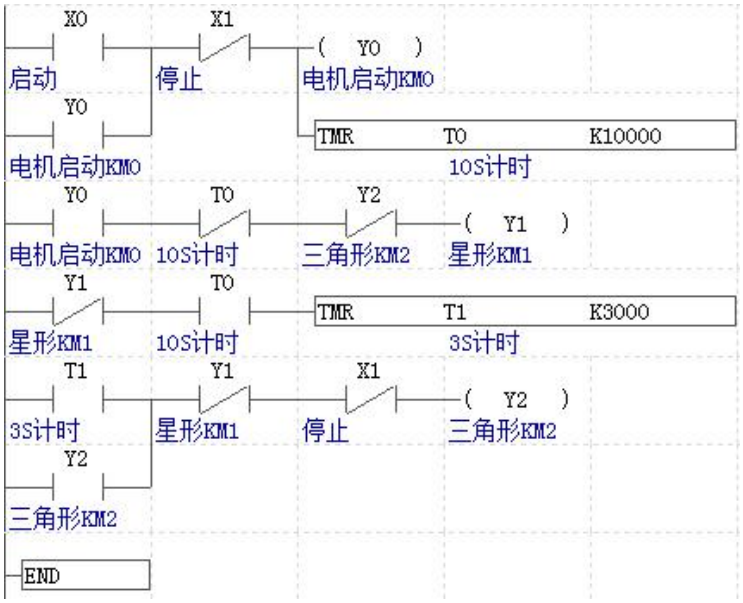
控制器外部接线示意图

Action required:

1. Three-phase AC induction motor starting current is high, usually 5 to 7 times the rated current, the starting current to reduce the effect on the grid, a star - delta reduced voltage start mode.

2. Star - delta reduced voltage start-up process: After the switch, the motor starts and star Contactor down manner starting contactor starting first. After the 10-second delay, the step-down star-shaped manner starting contactor is opened, and then after a lapse of three seconds delay triangle running contactor energized, the main motor is delta-connected circuit is running. Using two delay in order to ensure a star-voltage mode after starting to fully open the contactor is turned triangle running contactor.

Program control:



Program description:

1. Press the start button, X0 = On, Y0 = On and self-protection, motor starting contactors KM0 turned on while the timer T0, due to Y0 = On, T0 = Off, Y2 = Off, so Y1 = On, star-voltage mode is turned on to start contactor KM1.

2.T0 10 seconds after the timer reaches the predetermined value, T0 = On, Y1 = Off, T1 timer is started, 3s reaches a preset value, T1 = On, so Y2 = On, normal operation of the triangular guide contactor KM2 through.

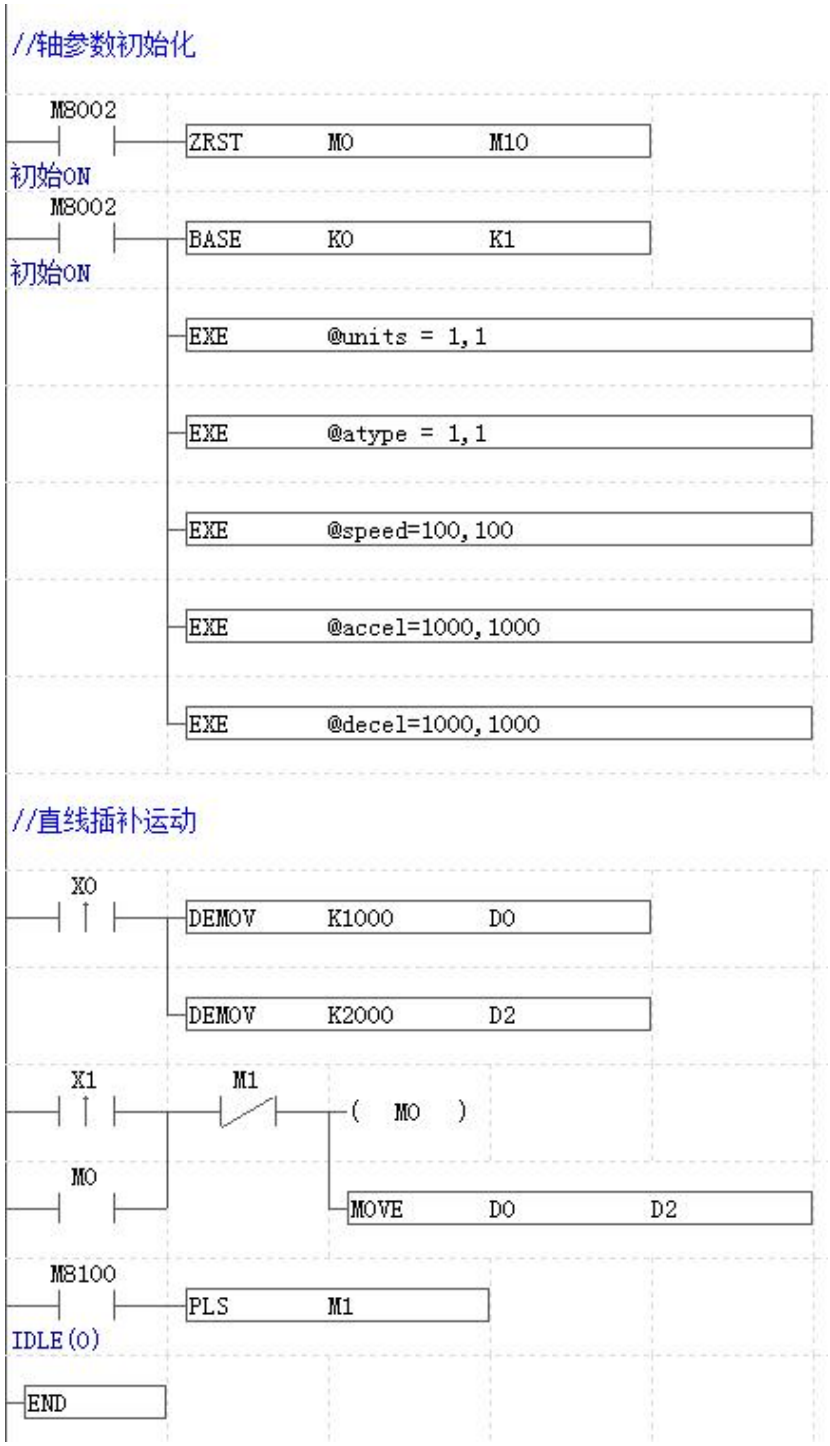
3. When you press the stop button, X1 = On, whether the motor is in start state or run state, Y0, Y1, Y2 are changed to Off, the motor stops running.

5.2 Linear interpolation motion

Action required:

Complete a simple linear interpolation of the two axes.

Program control:



Program description:

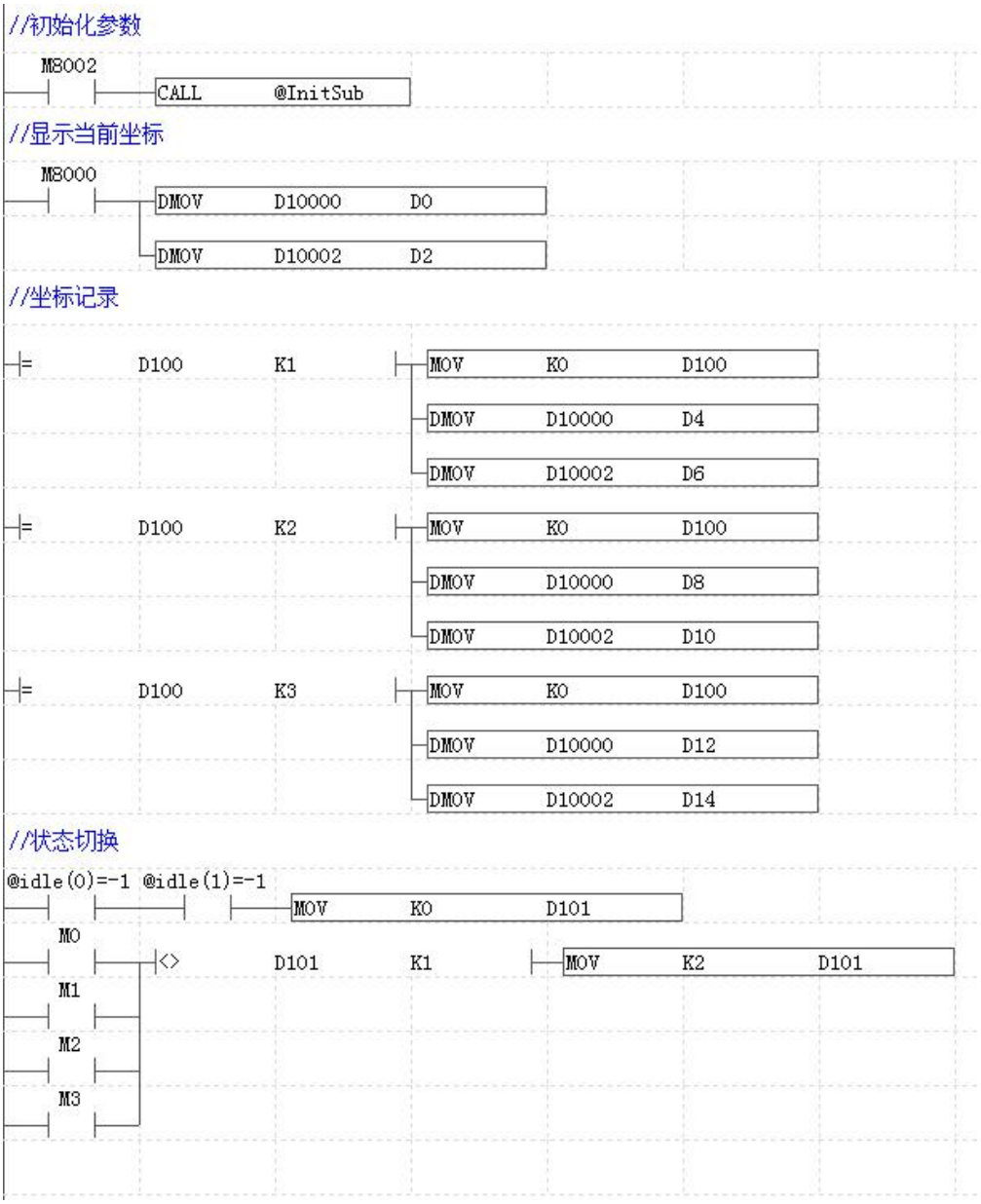
Power-on initialization program, various parameters are initialized shaft, when X0 is triggered when a rising edge X0 of the two axes of motion from the assignment registers D0, D2, X1 is triggered when a rising edge of the opening X1 and MOVE M0 self-locking motion when the motion is completed, a period is set Ml , M0 disconnected.

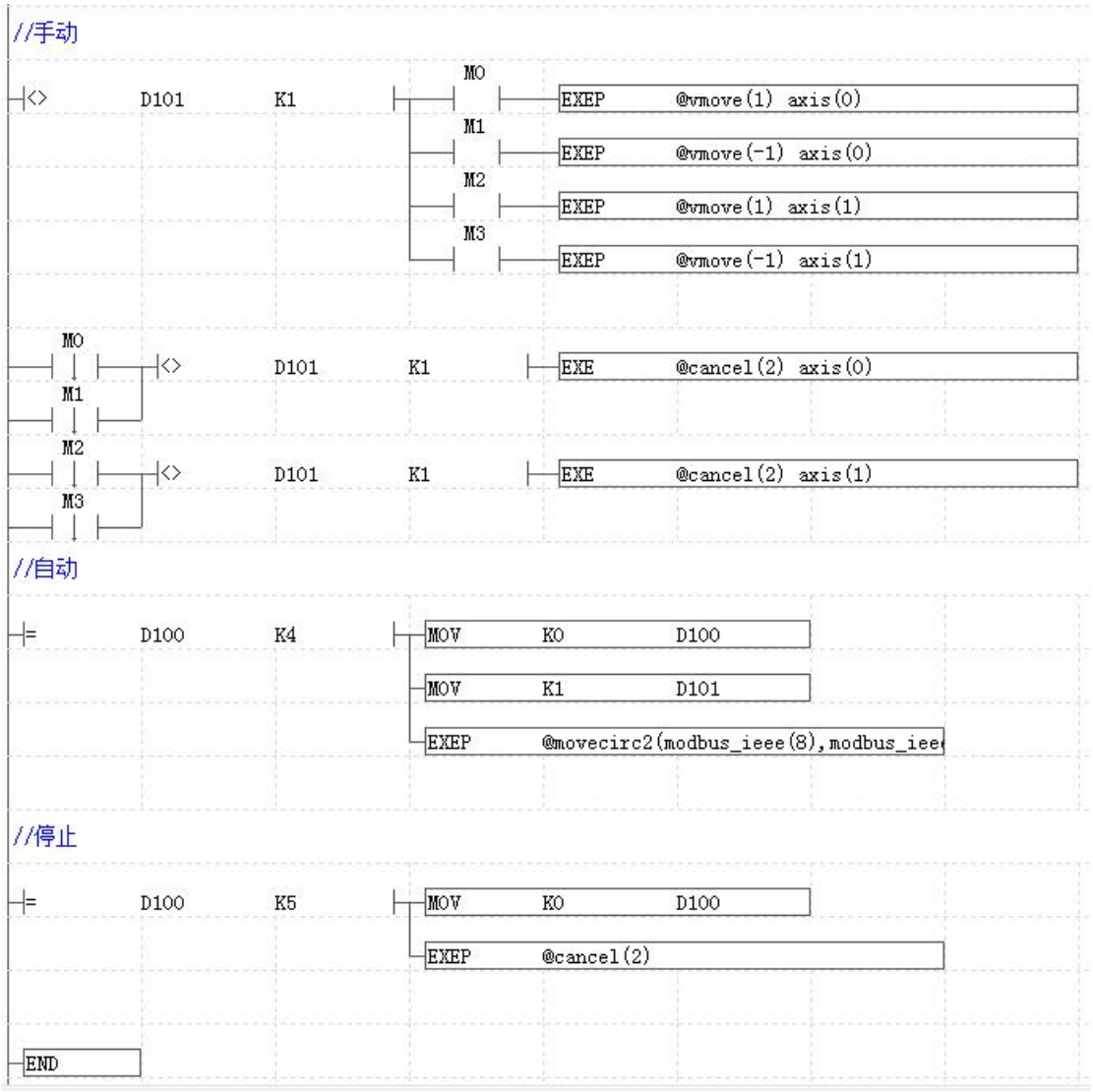
5.3 Three Draw an arc

Action required:

By moving a touchscreen keypad X, Y-axis coordinate position to the right, and positioning the arc and the end point coordinates of the intermediate point. After the positioning is completed, press the start key to implementing the specified arc operation, a stop key to stop the operation of the arc. The status bar displays the current system status in real time.

Program control:


//初始化参数
M8002
CALL @InitSub
//显示当前坐标
M8000
DMOV D10000 D0
DMOV D10002 D2
//坐标记录
= D100 K1
MOV K0 D100
DMOV D10000 D4
DMOV D10002 D6
= D100 K2
MOV K0 D100
DMOV D10000 D8
DMOV D10002 D10
= D100 K3
MOV K0 D100
DMOV D10000 D12
DMOV D10002 D14
//状态切换
@idle(0)=-1 @idle(1)=-1
MOV K0 D101
M0
<> D101 K1
MOV K2 D101
M1
M2
M3




```
1   GLOBAL SUB InitSub()
2       base(0,1)
3       atype = 1,1
4       units = 1,1
5       accel = 10000,10000
6       decel = 0,0
7       speed = 1000,1000
8       dpos = 0,0
9       modbus_reg(101)=0
10  END SUB
```