

GSK983M(T) CNC System

PLC Program Manual



GSK

广州数控设备有限公司
GSK CNC EQUIPMENT CO., LTD.



The operating manual describes all matters concerning the operation of the system in detail as much as possible. However, it is impractical to give particular descriptions of all unnecessary and/or unavailable works on the system due to the length limit of the manual, specific operations of the product and other causes. Therefore, the matters not specified herein may be considered impractical or unavailable.



This operating manual is the property of GSK CNC Equipment Co., Ltd. All rights reserved. It is against the law for any organization or single to publish or reprint this manual without the express written permission of GSK and the latter reserves the right to ascertain their legal liability.

Company Profile

GSK CNC EQUIPMENT CO., LTD(GSK) , CNC Industry Base of South China, is responsible for the National High Technology Research and Development Program of China (863 Program): Moderate CNC Industrialization Key Technology. For ten years, we are exclusively engaged in research, Development, manufacture, sale, training and popularization of Machine Tool CNC system, Servo Motor and driver, and other mechanical products. Today, GSK has already expressed into a large-scale new high-tech enterprise that deals with research, teaching, working and trading. Our products support more than 60 domestic manufacturers of machine tools with after-sales service network through the country. With a yield in the lead in China for four years in succession, GSK series products are in great demand in the domestic demand and have a ready sale in Southeast Asia at high performance-to-price ratio.

Field technical support services

Field support services are available when you encounter a problem insolvable through telephone. GSK CNC Equipment Company Limited will designate a technical support engineer to the field to solve technical problems for you.

Chinese version of all technical documents in Chinese and English languages is regarded as final.

Table of Contents

Section I	GSK983M/T CNC System	1
	Programmable Logic Controller Introduction	1
1	GSK983M/ T CNC System's PLC Control Mode	1
1.1	Executing Mode.....	1
1.2	High-Level and Low-Level	1
1.3	Input Signal Synchronous.....	2
1.4	Sequence Program Maximum Steps	4
2	NC Signal Address PLC	4
2.1	Signals Class.....	4
2.2	Address Symbol Meaning in The Ladder	4
3	Instruction System.....	4
3.1	Logic Operation Result.....	4
3.2	Instruction Type.....	5
3.3	Basic Instructions Illustration.....	7
3.4	Function Instructions Illustration	12
4	Nonvolatile Memory	37
5	PLC Data Displaying and Setting.....	38
5.1	PLC Data Displaying Page Selection.....	38
5.2	Signal Status Displaying.....	38
5.3	Manual Control PLC Signal status.....	38
5.4	Setting and Displaying Timer Values.....	39
5.5	Setting and Displaying Counter Preset Values and Current Values.....	39
5.6	Setting and Displaying The Sequence Memory	40
5.7	Setting and Displaying Data Table.....	40
6	Address Table	41
Section II	GSK983M/T CNC System	55
	PLC Programmer Software Introduction.....	55
1	Summary	55
2	System Requirements	55
3	The interface description.....	56
3.1	The total interface	56
3.2	Menu command.....	58
3.2.1	[File] Menu	58
3.2.2	[Edit]Menu.....	62
3.2.3	[CNC]Menu	65
3.2.4	[Help] Menu	66
3.3	Toolbar buttons	66

3.3.1	Toolbar.....	66
3.3.2	Toolbar for the ladder program	67
4	The limit of the Ladder program edit.....	69
5	GSK983M/T P L C Programmer Procedure	70

Section I GSK983M/T CNC System

Programmable Logic Controller Introduction

GSK983M/ T CNC system use the built-in programmable logic controller (PLC). So it is high security and the structure is close, because there has no connected line between the CNC and PLC.

The PLC Edit soft support ladder program.

1 GSK983M/ T CNC System's PLC Control Mode

1.1 Executing Mode

Since PLC sequence control handled by software, so operates on principles different from a general relay circuit.

The differences between them:

- 1) In a general relay sequence circuit, each relay operates at approximately the same time. In the PLC control, each relay operates sequentially. (the sequence can be written as a ladder diagram)
- 2) The sequence program is executed from the beginning of coding to the end of coding of the ladder diagram in the sequence written. When the sequence program ends, the program starts over from the beginning, This is called continuous operation.

1.2 High-Level and Low-Level

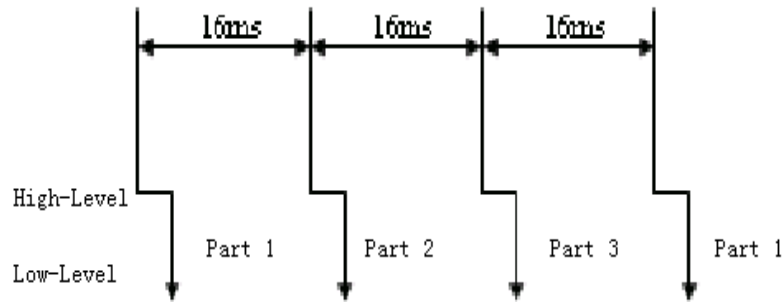
A sequence program consists of two parts; HIGH-level sequence and LOW-level sequence.

The HIGH-level sequence part operates every 16 ms.

The LOW-level sequence part must be divided in order to execute the HIGH-level sequence part. For example a sequence program is executed in the following sequence when the dividing number is n.

After the last LOW-level sequence part is executed, the sequence program is executed again from the beginning. Thus, when the dividing number is n, the cycle of execution is $16 \times n$ ms.

For example: The LOW-level sequence part is divided by 3, following show the sequence:



In general, the signals that must be executed immediately are deal with in the HIGH level. For example:

- 1) Emergency signal
- 2) Decelerate signal of return zero
- 3) Over travel
- 4) External decelerate signal
- 5) Feed hold

But, if the HIGH level program is too long, then the cycle time is very long. So, the HIGH level program is shorter and it is better.

1.3 Input Signal Synchronous

The input signals include the signals that transferred from CNC and machine tool.

PLC deal with the signals in the following mode:

1) The signals in the LOW level must be transferred into the synchronous registers first, then be deal with by the ladder program in the same time. So the signals are fixed during a LOW level cycle.

2) The signals in the HIGH level need not be transferred into the synchronous registers first. They are transferred directly in order to be deal with immediately.

Note: The status of the same input signal may be different in the HIGH level and LOW

level sequences. That is, at HIGH level, processing is performed using the asynchronous input signal memory and at LOW level processing is performed using the synchronous input signal memory. Therefore, it is possible for synchronous input signal to have a longer period at the LOW level sequence than the asynchronous input signal.

This must be kept in mind when writing the sequence program.

The following ladder program is an example:

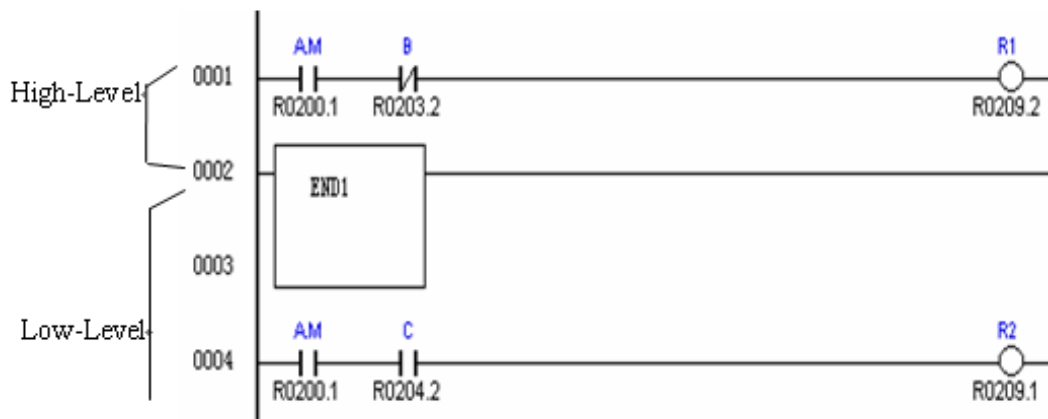


Fig1.

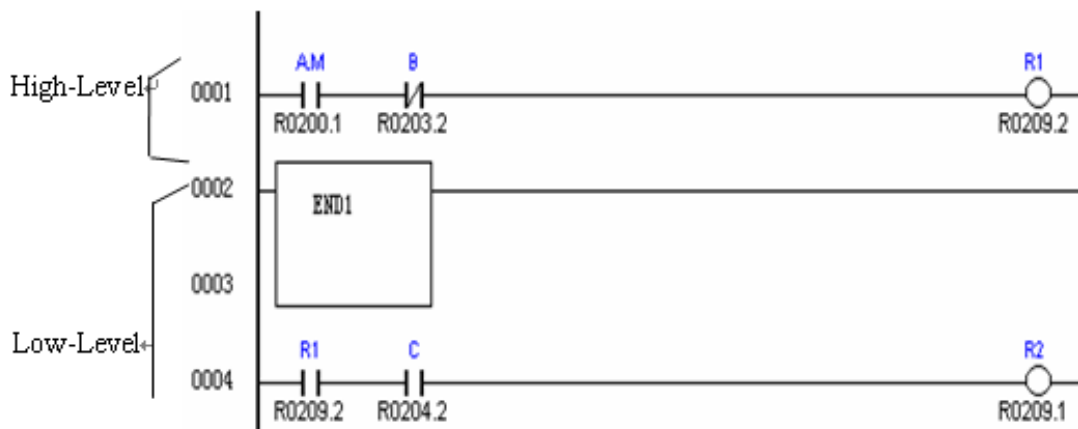


Fig 2

When A.M and C are turn on, B is turn off:

- 1) In fig 1, when R1 is turn on, R2 may not be turn on.
- 2) In fig 2, when R1 is turn on, R2 must be turn on.

1.4 Sequence Program Maximum Steps

In the GSK983M/ T NC's PLC, the maximum steps number is 2000.

2 NC Signal Address PLC

2.1 Signals Class

In the GSK983M/ T NC system, the signals classes:

- 1) Input to PLC from MT signals.
- 2) Input to PLC from CNC signals.
- 3) Output from PLC to CNC signals.
- 4) Output from PLC to MT signals.
- 5) Control relay

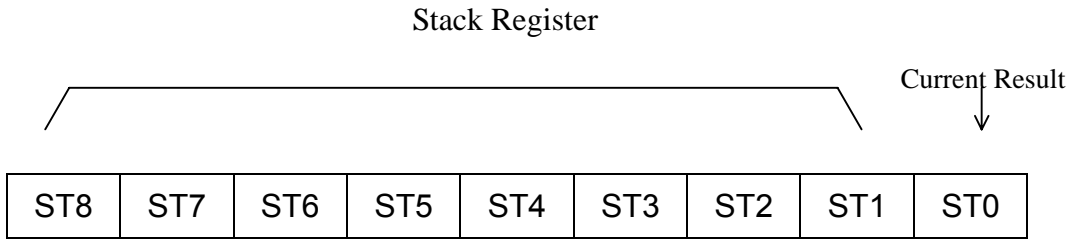
2.2 Address Symbol Meaning in The Ladder

SYMBO	MEANING	ADDRESS RANG
L		
Y	PLC->MT	0—15
X	MT->PLC	32—55
F	CNC->PLC	64—79
G	PLC->CNC	96—123
R	CONTROL RELAY	200—232
K	KEEP RELAY	596—638

3 Instruction System

3.1 Logic Operation Result

In the GSK983M/T CNC system, a register is provided for storing the intermediate results of a logical operation during operation of a sequence program. The register consists of 9 bits. See the figure bellow.



1) ST0

RD/RD.NOT instruction can put the status of a signal into the ST0. The register shift left one bit, when execute the RD.STK/RD.NOT.STK instruction one time. Then put the signal's status into ST0. As to say, the result can be saved before executing the RD.STK/RD.NOT.STK instruction. It can most save eight results.

2) ST1~ ST8

AND STK or OR.STK instruction can complete ST1 and ST0, and put the result into ST0. The register can shift right one bit.

3.2 Instruction Type

In the GSK983M/T CNC System, There are two types of PLC instructions, basic and function.

(i) Basic instruction

Basic instructions are most often used when designing sequence programs. They perform one-bit operations, such as AND or OR. There are 12 types.

(ii) Function instruction

Function instructions ease programming of machine movements. That is difficult to program with basic instructions. There are 22 types.

Basic instructions and contents of processing

No.	Instruction	Contents of processing
1	RD	Reads the status of a specified signal and sets it in ST0
2	RD.NOT	Inverts the logical status of a specified signal, reads and sets it in ST0.
3	WRT	Outputs the results of logical operations(status of

		ST0) to a specified address.
4	WRT.NOT	Inverts the results of logical operations(status of ST0)and outputs it to a specified address.
5	AND	Induces a logical product.
6	AND.NOT	Inverts the status of a specified signal and induces a logical product.
7	OR	Induces a logical sum.
8	OR.NOT	Inverts the status of a specified signal and induces a logical sum.
9	RD.STK	Shift the stack register left one bit and sets the status of a specified signal in ST0.
10	RD.NOT.STK	Shifts the stack register left one bit, inverts the logical status of a specified signal, reads and sets it in ST0.
11	AND.STK	Sets the logical product of ST0 and ST1,and shifts the stack register right one bit.
12	OR.STK	Sets the logical sum of ST0 and ST1,and shifts the stack register right by one bit.

Function instructions and contents of processing

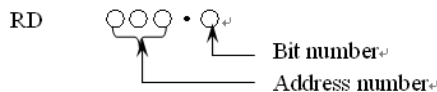
No.	Instruction	Contents of processing
1	END1	High-level program end
2	END2	Low-level program end
3	TMR	Timer process
4	DEC	Decoding
5	CTR	Counter process
6	ROT	Control of rotation
7	COD	Code conversion
8	MOVE	Data transfer after AND
9	COM	Common line control

10	JMP	Jump
11	PARI	Parity check
12	MWRT	Nonvolatile memory write
13	DCNV	Data conversion(binary to BCD and vice versa)
14	COMP	Comparison
15	COIN	Coincidence check
16	DSCH	Data search
17	NMOV	Indexed data transfer
18	ADD	Addition
19	SUB	Subtraction
20	MUL	Multiplication
21	DIV	Division
22	NUME	Definition of constant

3.3 Basic Instructions Illustration

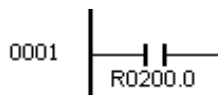
1) RD

Format:



Function: Reads the status (1 or 0) of a signal at a specified address and sets it in ST0.

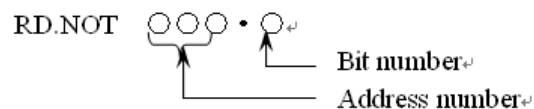
Example:



RD R200.0

2) RD.NOT

Format:



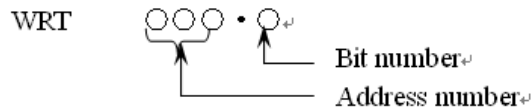
Function: Inverts the status of a signal at a specified address and set it in ST0.

Example:



3) WRT

Format:



Function: Outputs the results of logical operations .that is , the status (1 or 0)of ST0 to a specified address.

Example:

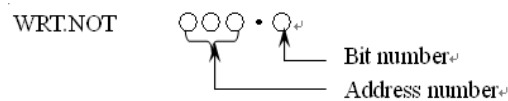


RD X32.3

WRT Y2.0

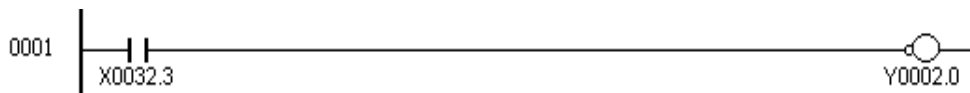
4) WRT.NOT

Format:



Function: Inverts the results of logical operations, that is, the status of ST0 and outputs it to a specified address.

Example:

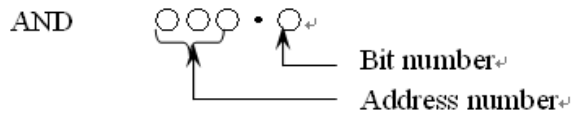


RD X32.3

WRT.NOT Y2.0

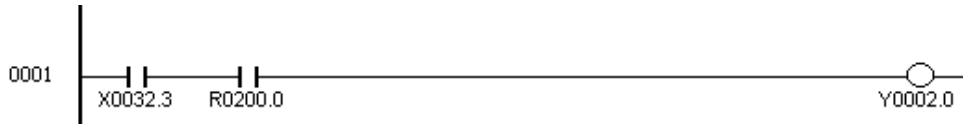
5) AND

Format:



Function: Induces a logical product.

Example:



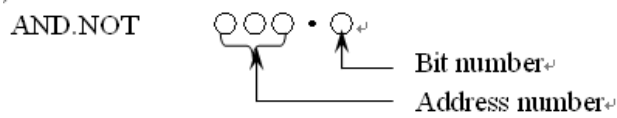
RD X32.3

AND R200.0

WRT Y2.0

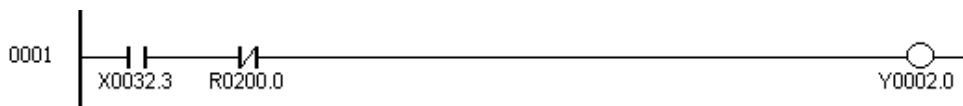
6) AND.NOT

Format:



Function: Inverts the status of a signal at a specified address and induces a logical product.

Example:



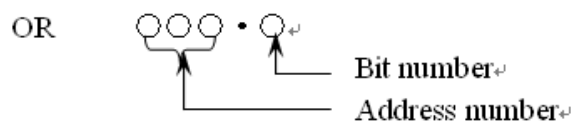
RD X32.3

AND.NOT R200.0

WRT Y2.0

7) OR

Format:



Function: Induces a logical sum.

Example:



RD X32.4

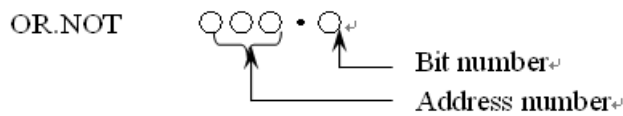
AND.NOT R201.2

OR Y7.0

WRT Y7.0

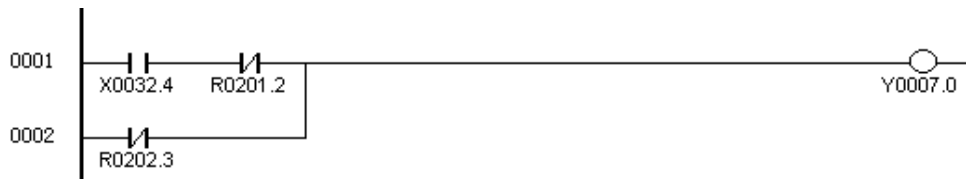
8) OR.NOT

Format:



Function: Inverts the status of a signal at a specified address and induces a logical sum.

Example:



RD X32.4

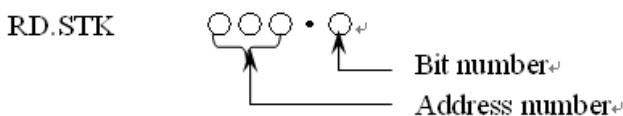
AND.NOT R201.2

OR.NOT R202.3

WRT Y7.0

9) RD.STK

Format:



Function: Stacks the intermediate results of a logical operation. Shifts the stack register left one bit, stacks the contents of ST0 in ST1, and sets a signal at a specified address to ST0.

Example:



RD R200.0

OR R200.3

RD.STK R200.1

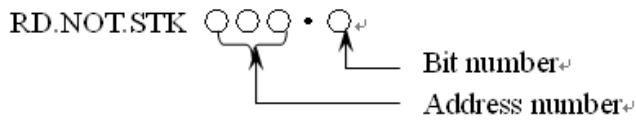
OR R200.4

AND.STK

WRT R201.4

10) RD.NOT.STK

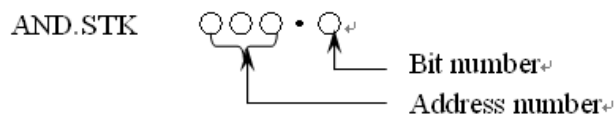
Format:



Function: Stacks the intermediate results of logical operations. Shifts left the stack register left one bit, inverts the status of a signal at a specified address and sets it in STD.

11) AND.STK

Format:

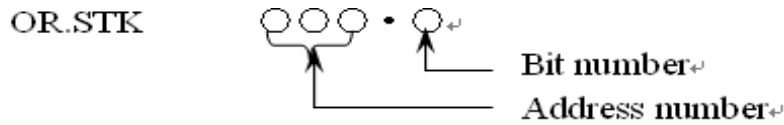


Function: Induces a logical product from the operation results in ST0 and ST1, sets the result in ST1, and shifts the stack register right one bit.

See **RD.STK** for an example of using the AND.STK instruction.

12) OR.STK

Format:



Function: Induces a logical sum from the operation results in ST0 and in ST1, sets the result in ST1, and shifts the stack register right on bit.

Example:



```

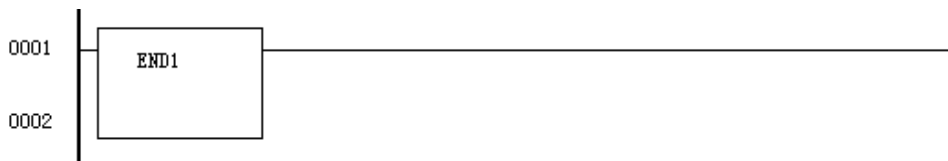
RD    R200.0
AND   R200.1
RD.STK R200.2
AND   R200.3
OR.STK
WRT   R201.1
    
```

3.4 Function Instructions Illustration

1) END 1

(1) Function: Must be specified once in a sequence program, either at the end of the high level sequence, or at the beginning of the low level sequence when there is no high level sequence.

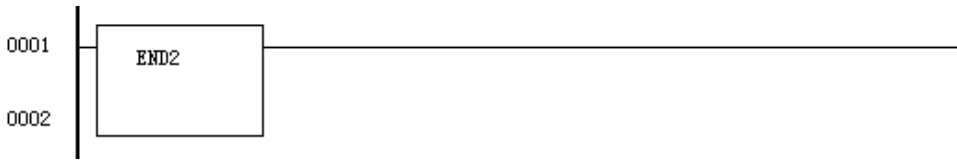
(2) Format: END1



2) END 2

(1) Function: Indicates the end of the sequence program and must be specified at the end of the low level sequence.

(2) Format: END2



3) TMR(Timer)

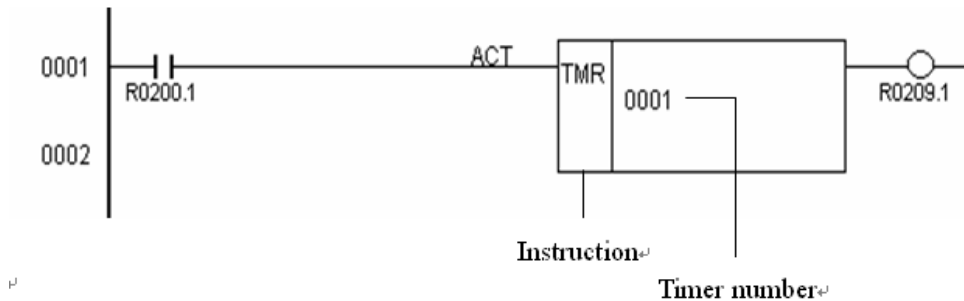
(1) Function: Timing.

(2) Format: TMR OO Timer number

(3) Control condition: ACT=0: turns off the timer relay (TM00).

ACT=1: initiates the timer.

(4) For example:



```
RD R200.1
TMR 1
WRT R209.1
```

Note:

The timer can be set via the MDI&CRT unit.

The setting time is every 50ms for timer number 1 to 8. The range for timers 1 to 2 is from 50 ms to 3276.7 sec, the range for timers 3 to 8 is from 50 ms to 12.7 sec

The timer less than 50ms is discarded.

The maximum error time of the timer is +130ms.

4) DEC(Decode)

(1) Function

The address of the 2-digit decimal data to be decoded is specified by the data address. Decode instruction specifies a code to be decoded. When the data coincides with the decode number, the relay for decoding result is set on.

(2) Format: DEC OOOO Address number of decode signal (two-digit BCD)

OOOO Decode instruction

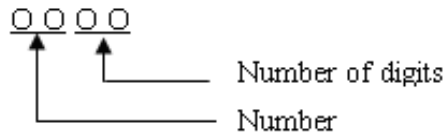
(3) Control condition: ACT=0: turns the decoding result output off (R0203.0).

ACT=1: performs decoding.

(4) Decode specification:

There are two paths, the number and the number of digits.

Decode specification



(i) Number: Must always be decoded in two digits. For example.

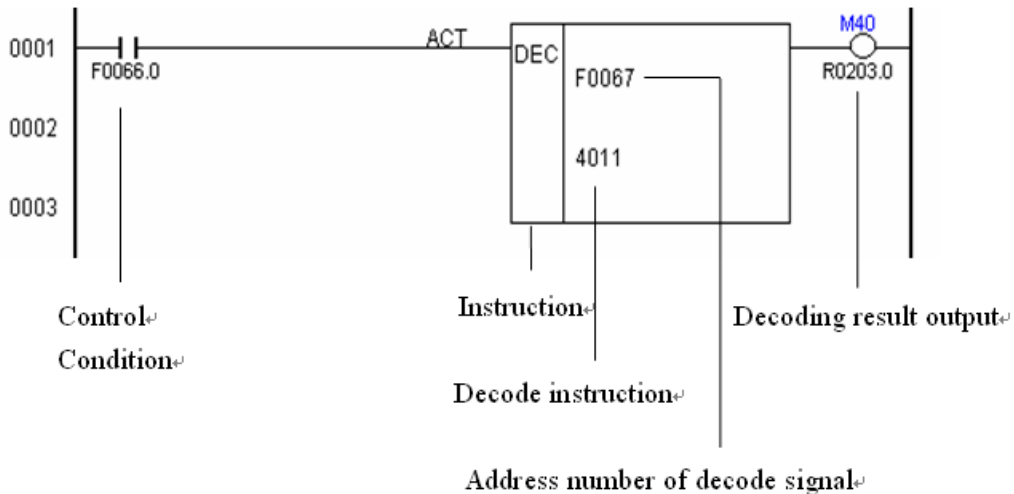
(ii) Number of digits

01: the high-order digit of two decimal digits is set to 0 and only the low-order digit is decoded.

10: the low-order digit is set to 0 and only the high-order digit is decoded.

11: two decimal digits are decoded.

(5) Example:



RD F66.0

DEC F67

4011

WRT R203.0

When the specified number equal to the code signal .R0203.0=1; when not, R0203.0=0.

R0203.0 decoding result output :

R0203.0 is 1 when the status of the code signal at a specified address is equal to a

specified number, 0 when not. The address of R203.0 is determined by the designer.

5) CTR(Counter)

(1) Function

CTR is used as a counter. The following counters are offered for more controlling NC machine tools. The types of control can be selected as required.

(i) Preset counter

Outputs a signal when the preset count is reached. The number can be preset from the MDI&CRT panel, or set in the sequence program.

(ii) Ring counter

Upon reaching the preset. Returns to the initial value by issuing another count signal.

(iii) UP/down counter

The count can be either up or down.

(iv) Selection of initial value

Selects the initial value as either 0 or 1

A combination of the preceding functions results in the ring counter below. Such a counter permits the position of a rotor to be memorized.

(2) Format: CTR

OO Counter number

(3) Control conditions

(a) Specify the initial value.

CNO=0: begins the value of the counter with 0.

CNO=1: begins the value of the counter with 1 (0 is not used).

(b) Specify up or down counter.

UPDOWN=0: up counter. The counter begins with 0 when CNO=0; 1 when 1.

UPDOWN=1: down counter. The counter begins with the preset value.

(c) Reset

RST=0: disabled reset.

RST=1: enables reset.

R becomes 0. The cumulative value is reset to the initial value.

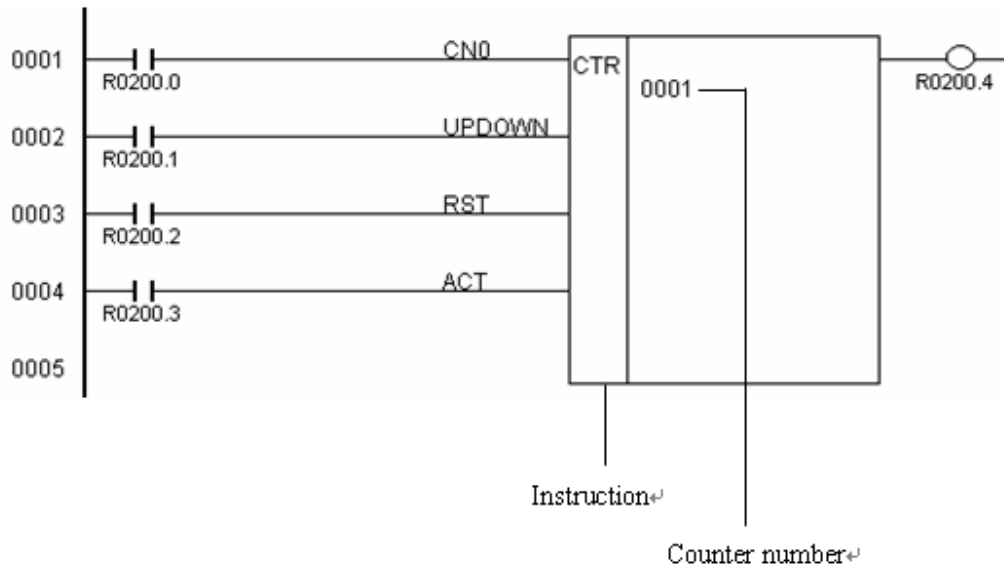
Note: Set RST to 1, only when reset is required. Otherwise the setting for nonvolatile memory may not be disabled.

(d) Count signal (ACT)

ACT=0 Counter does not operate.

ACT=1 Count is made by catching the rise of ACT.

(4) For example:



6) ROT (Control of Rotation)

(1) Function

Controls rotors, such as the tool post, ATC, rotary table, etc. and is used for the following functions.

- (a) Selection of the rotation direction via the shorter path
- (b) Calculation of the number of steps between the current position and the goal position
- (c) Calculation of the position one position before the goal or the number of steps up to one position before the goal.

(2) Format: ROT

- OOOO Rotor indexing address
- OOOO Current position address
- OOOO Goal position address
- OOOO Calculating result output address

(3) Control conditions

(a) Specify the starting number of the rotor.

RNO=0: begins the number of the position of the rotor with 0.

RNO=1: begins the number of the position of the rotor with 1.

(b) Specify the number of digits of the process data (position data).

BYT=0: BCD two digits

BYT=1: BCD four digits

(c) Select the rotation direction via the shorter path or not.

DIR=0: No direction is selected. The direction of rotation is only forward.

DIR=1: Selected. See (8) for details on the rotation direction .

(d) Specify the operating conditions

POS=0: Calculates the difference of the number of steps between the current position and the goal position.

POS=1: calculates the position one position before the goal, or the number of steps up to one position before the goal.

(e) Specify the position or the number of steps.

INC=0: calculates the number of the position. If the position one position before the goal position is to be calculated specify INC=0 and POS=1.

INC=1: calculates the number of the position. If the difference between the current position and the goal position is to be calculated, specify INC=1 and POS=0.

(f) Execution command

ACT=0: The ROT instruction is not executed. R1 does not change.

ACT=1: Executed. Normally, set ACT=0. If the operation results are required, set ACT=1.

(4) For example, following figure illustrates a ladder diagram for a 12-position rotor to be controlled for rotation via the shorter path and for deceleration at the position one position before the goal.

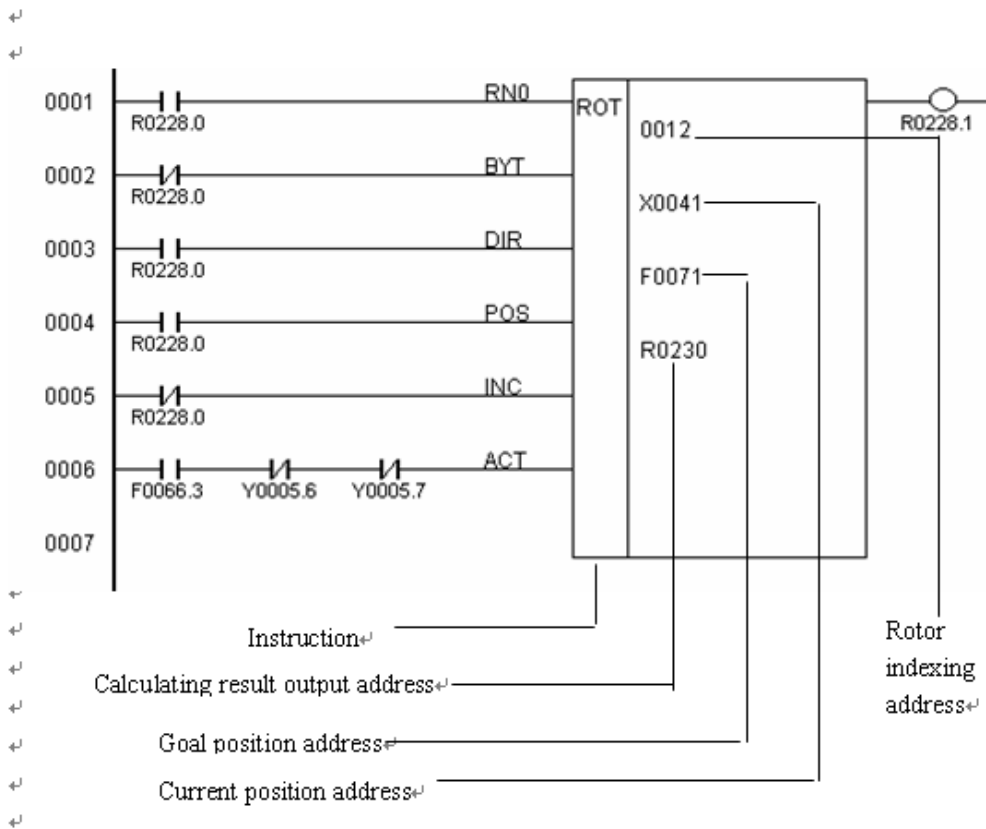
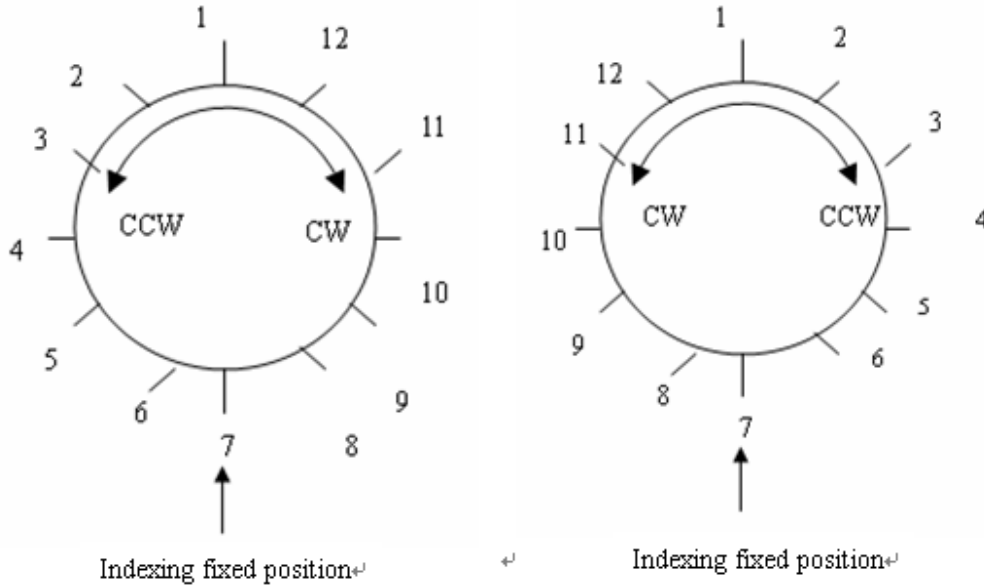
·The goal position is specified with NC T two digits (address 71)

·The current position is entered with the two-digit BCD code signal (address 41) from the machine tool.

·The result to calculating the position one position before the goal is output to address

230 (workarea).

- Operation starts with the output TF (address 66.3) from the NC.
- The coincidence check instruction (COIN) is used to detect the deceleration and stop positions.



7) COD (Code Conversion)

(1) Function

Converts BCD codes into an arbitrary 2-digit or 4-digit BCD number. For code conversion, the conversion table address, conversion table, and convert data output address must be provided.

(2) Format: COD

OOOO	Size of the data table
OOOO	Conversion table address
OOOO	Convert data output address
OOOO	
	Conversion table
OOOO	

(3) Control condition

(a) Specify the data size

BYT=0: Specifies that the conversion table data is to be BCD two digits.

BYT=1: Specifies that the conversion table data is to be BCD four digits.

(b) Error output reset

RST=0: Disable reset

RST=1: Sets error output R1 to 0 (resets).

(c) Execution command

ACT=0: The COD instruction is not executed.

ACT=1: Executed.

(4) Specify the size of table

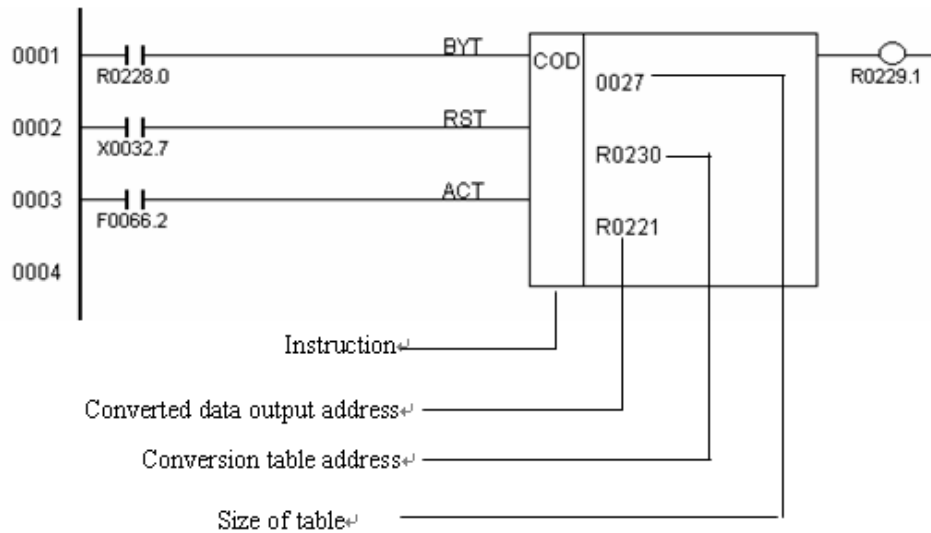
A conversion table address from 0 to 99 can be specified.

(5) Error output (R)

R=0: The conversion is right.

R=1: The conversion is error.

(6) For example:



8) MOVE (Logical Product Transfer)

(1) Function

AND comparison and process data. And transfers the results to specified address. It can also be used to remove unnecessary bits from an eight-bit signal in a specific address, etc.

(Comparison data)^(Process data)→To a specified address

The process data is one byte (eight bits).

(2) Format: MOVE

OOOO High-order 4-bit comparison data

OOOO Low-order 4-bit comparison data

OOOO Process data address

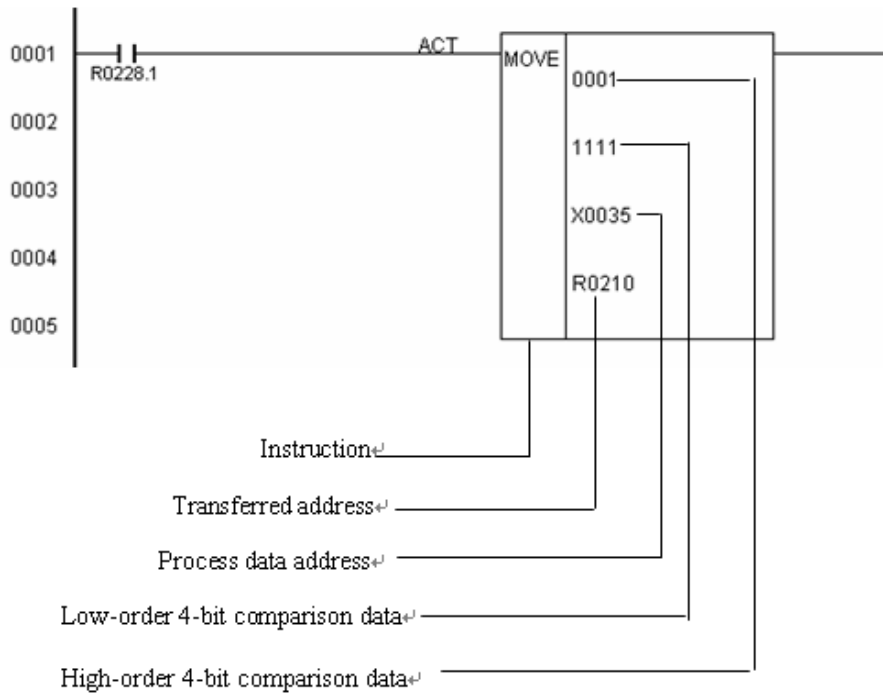
OOOO Transferred address

(3) Execution command

ACT=0: Move instruction not executed.

ACT=1: Executed.

(4) Example of using the MOVE instruction



9) COM(Common Line Control)

(1) Function

Turn off specified relays.

(2) Format: COM

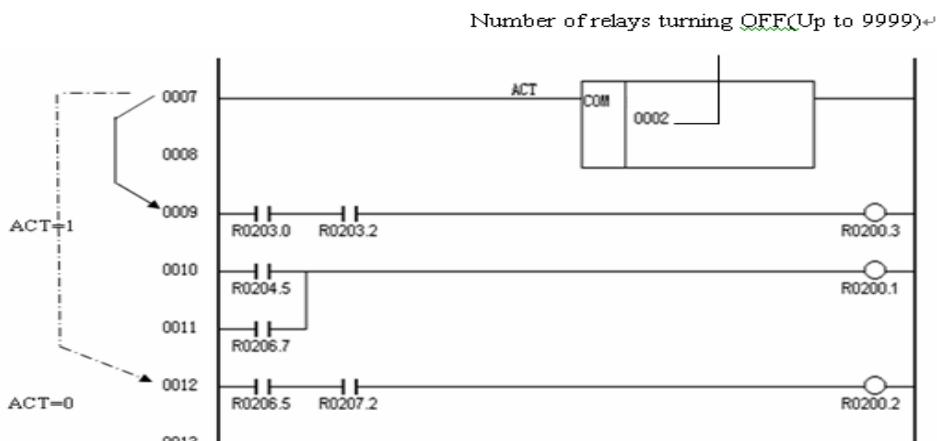
OOOO Number of relays turning off

(3) Control conditions

ACT 0: unconditionally turns off (0) the specified relays and begins processing from the next step to the relay turned off last.

1: begins processing from the next step after COM instruction.

(4) For example:



Note1: A function instruction in a range specified by COM execute processing, regardless of COM ACT. However, if COM ACT=0, however, the relay of the execution result becomes 0 unconditionally.

Note2: Another COM instruction cannot be specified in the range specified by the COM instruction.

Note3: If COM ACT=0, the relay written in by a WRT.NOT instruction in a range specified by COM becomes 1 unconditionally.

10) JMP (Jump)

(1) Function

Skip logical operations according to the number of specified relay.

(2) Format: JMP

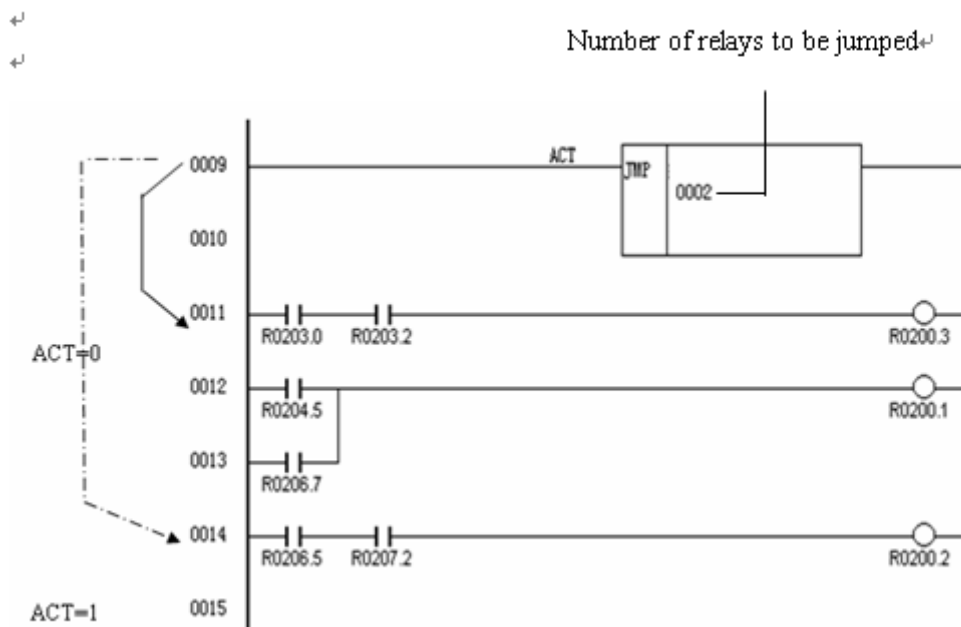
OOOO Number of relays to be skipped

(1) Control conditions

ACT=0: Skip the JMP function, execute the next step directly.

ACT=1: Logical operations are skipped according to the specified number of relays.

(5) For example:



11) PARI (Parity Check)

(1) Function

Checks the parity of code signals, and outputs an error if an abnormality is detected.

Specify either an even or odd-parity check. Only one-byte (eight bits) of data can be checked.

(2) Format: PARI

OOOO Check data address

(3) Control conditions

(i) Specify even or odd.

OE=0: even-parity check

OE=1: odd-parity check

(ii) Reset

RST=0: Disables reset.

RST=1: sets error output R to 0. That is, when a parity error occurs, setting RST to 1 result in resetting.

(iii) Execution command

ACT=0: Parity checks are not performed.

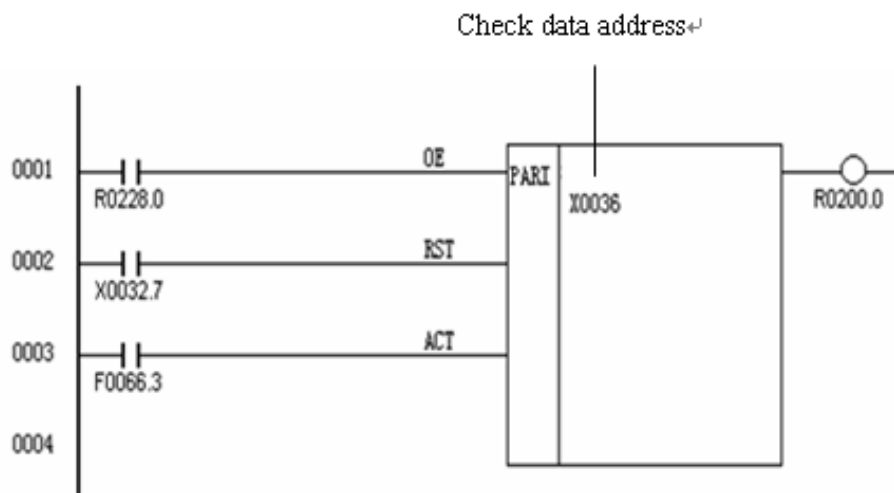
ACT=1: executes the PARI instruction, performing a parity check.

(4) Error output

If the results of executing the PARI instruction is abnormal, R=1 and an error is posted.

The R address can be determined arbitrarily.

(5) Example of using the PARI instruction



12) MWRT (Write in nonvolatile memory)

(1) Function

Writes the contents of nonvolatile memory image area address (nonvolatile memory image) (Note) into the nonvolatile memory. When the sequence program changes the data such as the contents of the keep relay, counters, and data tables, the MWRT instruction can be used to rewrite the contents of the nonvolatile memory.

(2) Format: MWRT

OOOO Rewrite address

(3) Control conditions

(i) Rewrite address specification

DIN=0: direct addressing

Specifies rewrite addresses directly in the rewrite address part of the MWRT Instruction.

DIN=1: indirect addressing

The address specified in the MWRT instruction differs from the actual rewrite address. The sequence program specifies in the address an address in nonvolatile memory, the contents of which is to be rewritten.

In indirect addressing, the address in which a data is rewritten is not fixed when preparing a sequence program, but can be changed during execution of the sequence program.

This specification is effective each time the write-in address changes (because one MWRT instruction will do).

(ii) Execution command

ACT=0: The MWRT instruction is not executed.

ACT=1: Executed.

(4) Write-in end notice

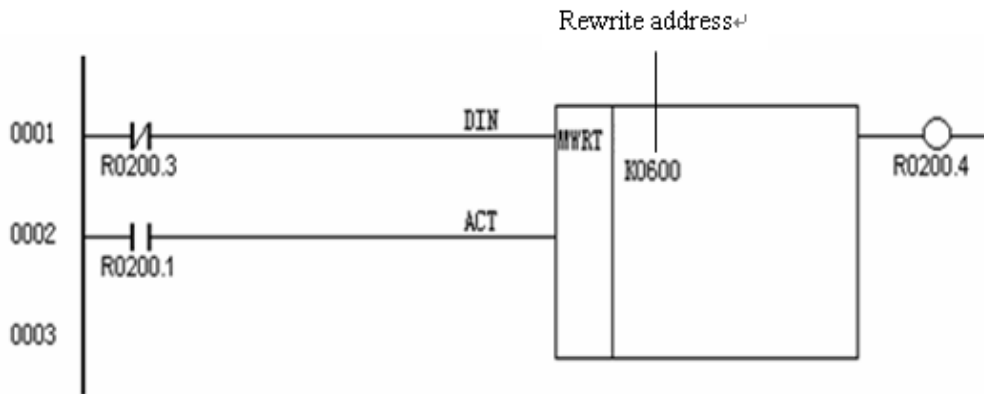
R=0: completes no write-in. Also, when ACT=0, R=0.

R=1: completes write-in. Set ACT to 0, immediately after R becomes 1. Keeping ACT 1 after write-in is completed, repeats R=1 to R=0 to R=1...

(5) Rewrite address

Specify the rewrite nonvolatile memory address. The rewrite address can be specified in two ways: direct addressing and indirect addressing.

(6) Example of using the MWRT instruction



13) DCNV (Data conversion)

(1) Function

Convert binary-code data into BCD-code data and vice versa.

(2) Format: DCNV

- OOOO Converted data address
- OOOO Conversion result storage address

(3) Control conditions

(i) Specify data size.

- BYT=0: Process data in length of one byte (8 bits)
- BYT=1: Process data in length of two bytes (16 bits)

(ii) Specify the type of conversion

- CNV=0: converts binary-code data into BCD-code data.
- CNV=1: converts BCD-code data into binary-code data.

(iii) Reset

- RST=0: Disables reset.
- RST=1: resets error output R. That is, setting RST to 1 when R=1, makes R=0.

(iv) Execution command

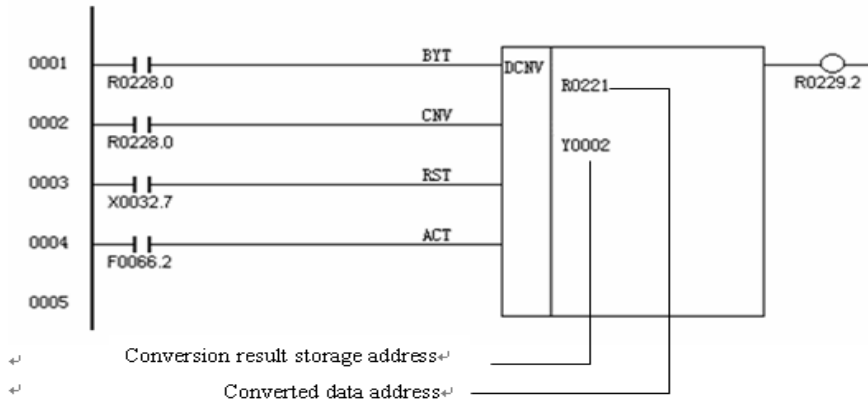
- ACT=0: Data is not converted.
- ACT=1: Data is converted.

(4) Error output (R)

- R=0: normal
- R=1: conversion error

R=1 of the converted data which should be BCD data, is binary data, or if the data size (byte length) specified in advance is exceeded when converting binary data into BCD data.

(5) Example of using DCNV instruction



14) COMP (Comparison)

(1) Function

Compares reference and comparison values.

(2) Format: COMP

OOOO Reference value (constant or address)

OOOO Comparison value address

(3) Control conditions

(i) Specify the data size.

BYT=0: process data, reference value, and comparison value. Each BCD is two digits long.

BYT=1: Each BCD is four digits long.

(ii) Specify the reference value format.

DAT=0: specifies the reference value as a constant.

DAT=1: specifies the reference value as an address (the address storing the reference value)

(iii) Execution command

ACT=0: The COMP instruction is not executed.

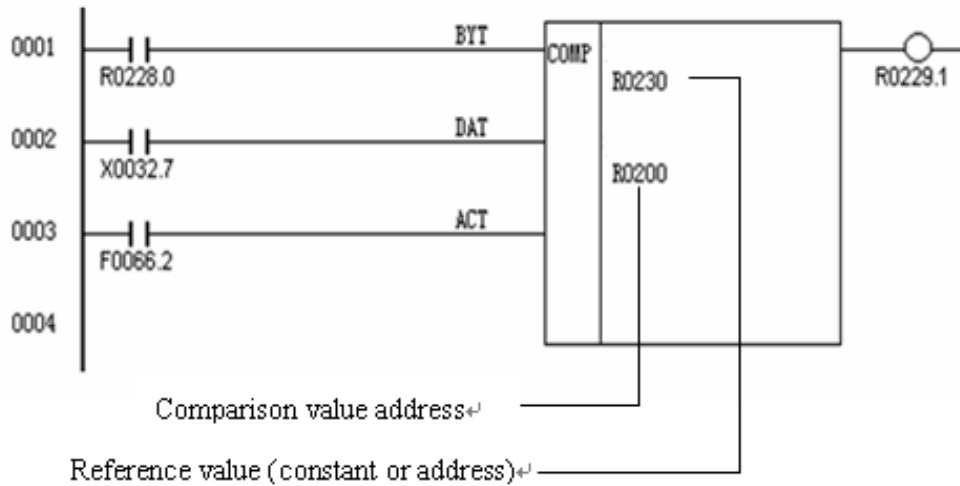
ACT=1: The COMP instruction is executed and the result is output to R.

(4) Comparison result output

R=0: reference value > comparison value

R=1: reference value ≤ comparison value

(5) For example:



15) COIN (Coincidence Check)

(1) Function

Check whether the reference and comparison values coincide.

(2) Format: COIN

OOOO Reference value (address)

OOOO Comparison value address

(3) Control conditions

(i) Specify the data size

BYT=0: process data (reference and comparison values). Each BCD is two digits long.

BYT=1: Each BCD is four digits long.

(ii) Specify the reference value format.

DAT=0: specifies the reference value as a constant.

DAT=1: specifies the reference value as an address (the address storing the reference value)

(iii) Execution command

ACT=0: The COIN instruction is not executed.

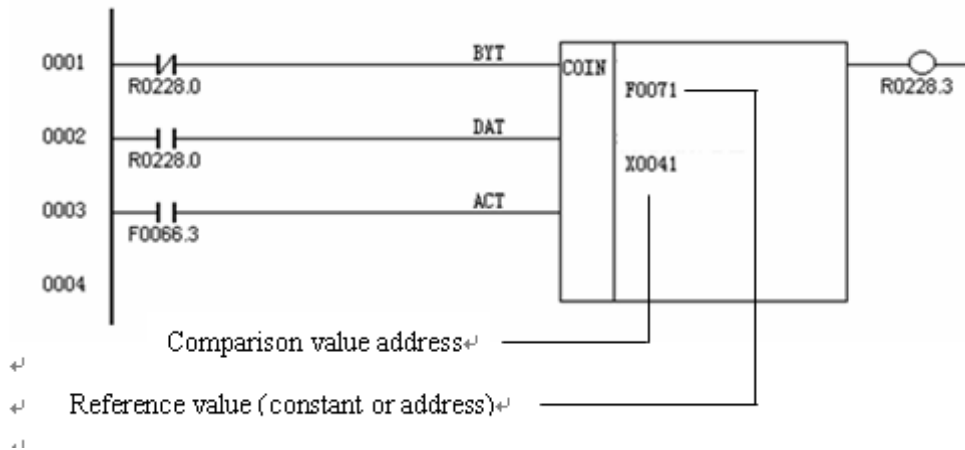
ACT=1: The COIN instruction is executed and the results output to R.

(4) Comparison result output

R=0: reference value≠comparison value

R=1: reference value=comparison value

(5) For example:



16) DSCH (Data Search)

(1) Function

DSCH is only valid for data tables (see item 10.2) which can be used by the PLC. DSCH searches the data table for the specified data, outputs the address storing it counting from the beginning of the data table. If the data cannot be found, an output is made accordingly.

(2) Format: DSCH

- OOOO Number of data of the data table
- OOOO Data table heading address
- OOOO Search data address
- OOOO Search result output address

(3) Control conditions

(i) Specify the data size

BYT=0: data stored in the data table, BCD two digits long

BYT=1: data stored in the data table, BCD four digits long

(ii) Reset

RST=0: disable reset.

RST=1: enables a reset, that is, sets R to 0.

(iii) Execution command

ACT=0: The DSCH instruction is not executed. R does not change.

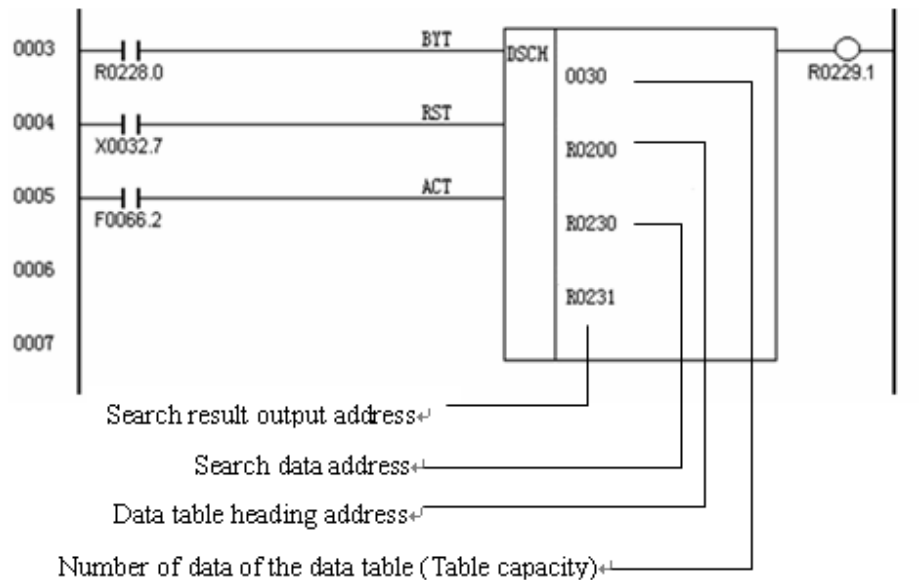
ACT=1: The DSCH is executed, and the table internal number storing the desired data is output. If the data cannot be found, R=1.

(4) Search data presence/absence output

R=0: The data to be searched exists.

R=1: The data to be searched does not exist.

(5) For example:



17) XMOV (Indexed Data Transfer)

(1) Function

Reads or rewrites the contents of the data table. Like the DSCH instruction, XMOV is only valid for data tables which can be used by the PLC.

(2) Format: XMOV

OOOO Number of data of the data table

OOOO Data table head address

OOOO Address storing input/output data

OOOO Address storing the table internal number

(3) Control Condition

(i) Specify the number of digits of data.

BYT=0: data stored in the data table, BCD is two digits long.

BYT=1: data stored in the data table, BCD is four digits long.

(ii) Specify read or write.

RW=0: Data is read from the data table.

RW=1: Data is written in the data table.

(iii) Reset

RST=0: disables reset.

RST=1: enables reset, that is, sets R to 0.

(iv) Execution command

ACT=0: The XMOV instruction is not executed.

ACT=1: The XMOV instruction is executed.

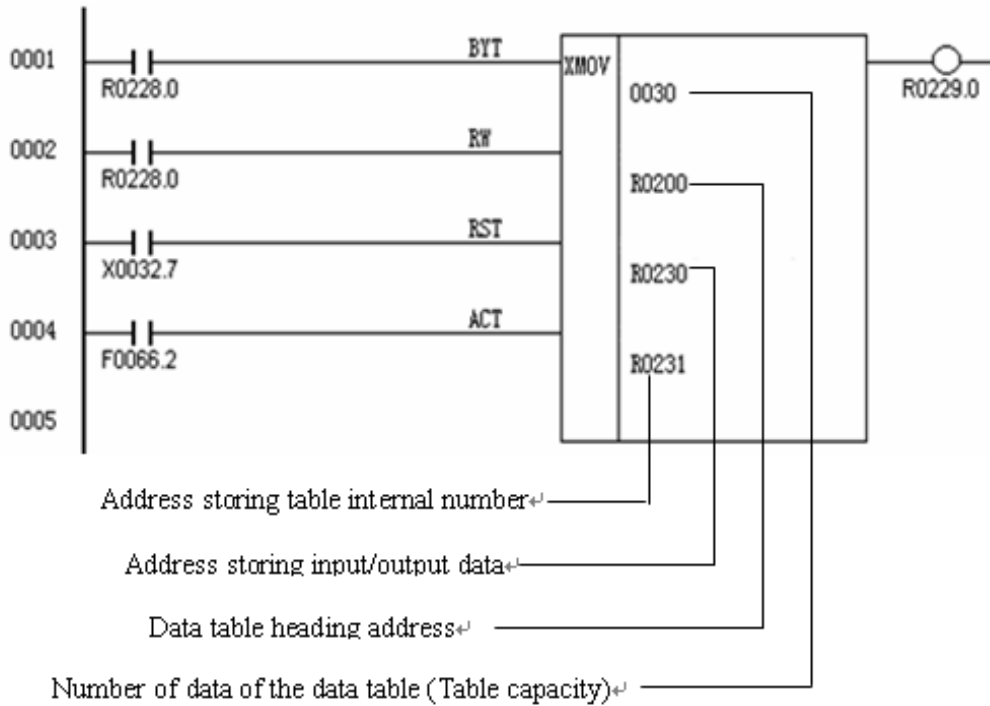
(4) Error output

R=0: There is no error.

R=1: There is an error.

An error occurs if a table internal number exceeding the previously programmed number of the data table is specified.

(5) For example:



18) ADD (Addition)

(1) Function

Adds BCD two-digit or four-digit data.

(2) Format: ADD

OOOO Summand address
 OOOO Addend or addend address
 OOOO Sum output address

(3) Control conditions

(i) Specify the number of digits of data.

BYT=0: Data is BCD two digits long

BYT=1: Data is BCD four digits long

(ii) Address the addend.

DAT=0: sets the address storing the addend (indirect addressing).

DAT=1: directly specifies the addend.

(iii) Reset

R=0: disable reset.

R=1: resets error output R, that is, sets R to 0.

(iv) Execution command

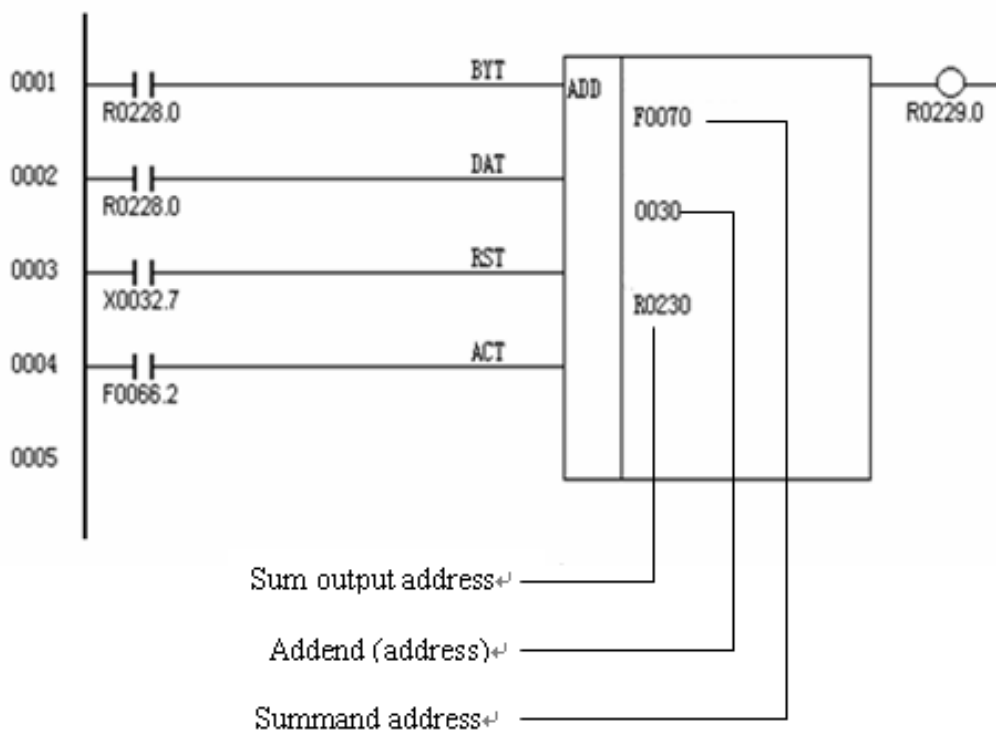
ACT=0: The ADD instruction is not executed.

ACT=1: The ADD instruction is executed.

(4) Error output

If the sum exceeds the data size specified in (3), (i), R=1 is set to indicate an error.

(5) For example:



19) SUB (Subtraction)

(1) Function

Subtracts BCD two-digit or four-digit data.

(2) Format: SUB

OOOO Minuend address

OOOO Subtrahend or subtrahend address

OOOO Difference output address

(3) Control conditions

(i) BYT=0: Data BDC is two digits long

BYT=1: Data BCD is four digits long

(ii) Address the subtrahend.

DAT=0; sets the address storing the subtrahend (indirect addressing)

DAT=1: directly specifies the subtrahend.

(iii) Reset

RST=0: disables reset.

RST=1: resets error output R, that is, sets R to 0.

(iv) Execution command

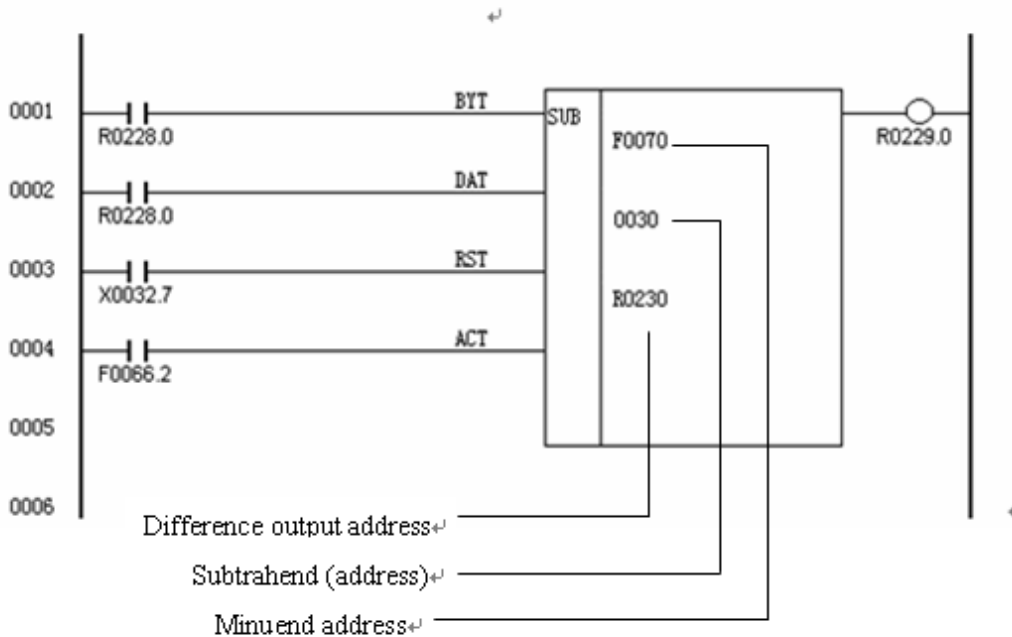
ACT=0: The SUB instructions is not executed. R does not change.

ACT=1: The SUB instruction is executed.

(4) Error output

R is set to indicate an error if the difference is negative.

(5) For example:



20) MUL (Multiplication)

(1) Function

Multiplies BCD two-digit or four-digit data. The product must also be BCD two- or four-digit data.

(2) Format: MUL

- OOOO Multiplicand address
- OOOO Multiplier or multiplier address
- OOOO Product output address

(3) Control conditions

(i) Specify the number of digits of data.

BYT=0: Data is BCD two digits long.

BYT=1: Data is BCD four digits long.

(ii) Address the multiplier.

DAT=0: specifies the address storing the multiplier (indirect addressing).

DAT=1: directly specifies the multiplier.

(iii) Rest

RST=0: disables reset.

RST=1: resets error output R, that is, sets R to 0.

(iv) Execution command

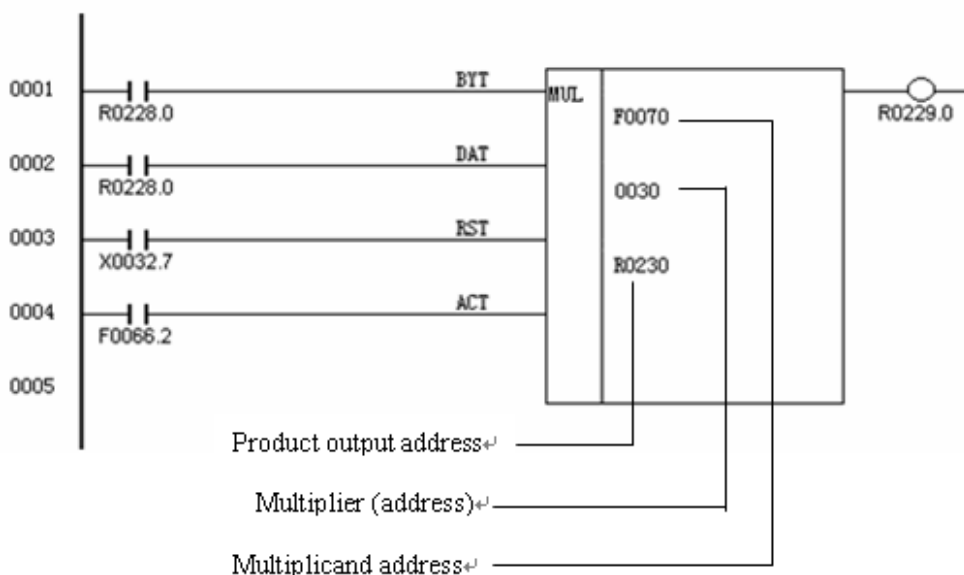
ACT=0: The MUL instruction is not executed.

ACT=1: The MUL instruction is executed.

(4) Error output

R=1 is set to indicate an error if the product exceeds the size specified in (3), (i).

(5) For example:



21) DIV (Division)

(1) Function

Divides BCD two-digit or four-digit data. Remainders are discarded.

(2) Format: OOOO Dividend address

OOOO Divider or divider address

OOOO Quotient output address

(3) Control conditions

(i) Specify the number of digits of data.

BYT=0: Data is BCD two digits long.

BYT=1: Data is BCD four digits long.

(ii) Address the divider.

DAT=0: specifies the address storing the divider (indirect addressing)

DAT=1: directly specifies the divider

(iii) Reset

RST=0: disables reset.

RST=1: resets error output R, that is, sets R to 0.

(iv) Execution command

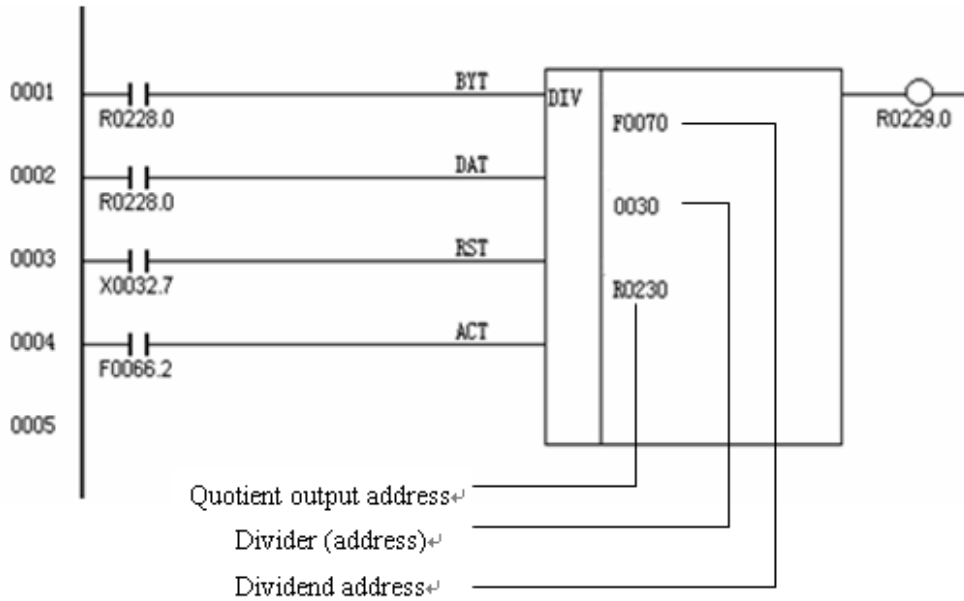
ACT=0: The DIV instruction is not executed. R does not change.

ACT=1: DIV instruction is executed.

(4) Error output

R=1 is set to indicate an error if the divider is 0.

(5) For example:



22) NUME (Definition of a Constants)

(1) Function

Defines constants, when required.

(2) Format: NUME

OOOO	Constant
OOOO	Constant output address

(3) Control conditions

(i) Specify the number of digits of the constant.

BYT=0: Constant is BCD two digits long.

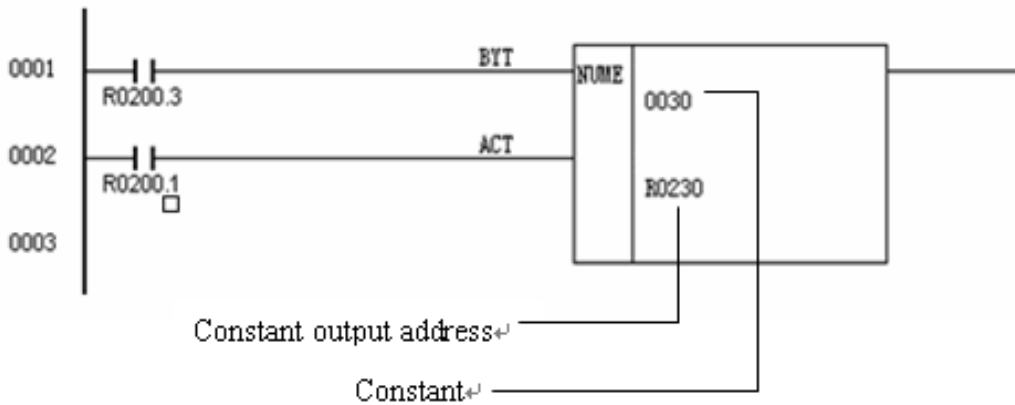
BYT=1: Constant is BCD four digits long.

(ii) Execution command

ACT=0: The NUME instruction is not executed.

ACT=1: The NUME instruction is executed.

(4) For example:



4 Nonvolatile Memory

Contents in nonvolatile memory are not erased when the power is turn off.

It can be used as following:

1 Used for the timer

Time can be set and display at the MDI & CRT unit of the CNC. The set time cannot be read or written by a sequence program instruction.

2 Used for the counter

Values can be set and display at the MDI & CRT unit of the CNC. These values can be read and written by a sequence program instruction. The data format if four digits of BCD, and the lower-order digits are entered at the smaller address.

Example: PLC counter addresses are 596 and 597 the set value is 1234.

	7	6	5	4	3	2	1	0	
Address	0	0	1	1	0	1	0	0	(34)
0596									

0597	0	0	0	1	0	0	1	0	(12)
------	---	---	---	---	---	---	---	---	------

3 Nonvolatile sequence memory

This memory is used as parameters, keep relays, etc. for sequence control. Setting and display are possible from the MDI & DPL unit of the CNC and sequence program instructions can be used for reading and writing. The data format is eight binary.

4 Nonvolatile memory control

This memory is used in the case of the position of a moving part of the machine tool, such as lathe turret, is stored in code and to maintain it while power is off.

5 PLC Data Displaying and Setting

5.1 PLC Data Displaying Page Selection

LCD displays PLD data when push key [PARAM] tow times, and can select the data by push [PAGE] or [UP] or [DOWON]. For example:

```
PC  PARAMETER  01:
NO.0001  DATA  10010010
```

5.2 Signal Status Displaying

It can display the sequence status by following steps.

- (1) Push key **【DGNOS】** , in the diagnose page.
- (2) Push key **【N】** , and then push the address number to be searched, and push **【INPUT】**

5.3 Manual Control PLC Signal status

Can setting the status of signal to 1 or 0.

Setting steps as follow:

- (1) Set bit 7 of No.11 parameter to 1.
- (2) Select MDI model.
- (3) Move cursor to the address of signal need to changed.
- (4) Push key **【P】** ,and then push the signal data, push key **【INPUT】**

5.4 Setting and Displaying Timer Values

The timer number: from No.1 to No.18

Timer number specified by program	Timer number specified by the MDI&LCD	Minimum set time	Maximum set time
1~2	1001~1002	50 ms	3276.7 sec
3~18	1003~1018	50 ms	12.7s

The timer number is used by the MDI&DPL are the timer numbers used by the sequence program plus 1000.

For example, if the timer number used by the sequenced in the ladder is No.1, user must put timer value into the NO.1001 by the MDI&DPL

1) Displaying Timer Values

- a) Push **【PARAM】** and select the PC screen
- b) Push key**【N】**, and then push the timer number to be displayed, and push**【INPUT】**
- c) The timer value of the specified timer number is displayed at the upper left of the screen.

2) Setting Timer Values

To set timer values, set the NC memory write switch to enable and put the NC in MDI mode.

Following steps display timer values.

- a) Push key **【P】** .
- b) Set the timer value.
- c) Push **【INPUT】** .The time is set in the specified timer number.
- d) Restore the NC memory write switch.

5.5 Setting and Displaying Counter Preset Values and Current Values

Counter	Setting Preset Value	Setting Current Value
---------	----------------------	-----------------------

number specified by program	Counter number specified by display	Address specified by program	Counter number specified by program	Address specified by program
1	2001	596, 597	2101	604, 605
2	2002	598, 599	2102	606, 607

Use the following procedure to display the counter values.

- a) Push **【PARAM】** and select the PC screen.
- b) Push key **【N】** and push the number of the counter to be displayed, and push **【INPUT】** .
- c) The current value of the specified counter number is displayed at the screen.

Use the following procedure to set the counter values.

- a) Set the NC parameter write switch to enable and put the NC in MDI mode.
- b) Push key **【P】** and set the value, then push **【INPUT】** .

5.6 Setting and Displaying The Sequence Memory

This memory is used for keep relays and sequence control parameters and is not erased when power is turned off. The sequence memory numbers used by the MDI&DPL unit are 3001 to 3004, corresponding to the addresses written in the address table, that is 3001 is used for setting or displaying in address 600. The data set or displayed is 0 or 1.

Address specified by program	number specified by display
600	3001
601	3002
602	3003
603	3004

5.7 Setting and Displaying Data Table

- 1) Number of data items

#603

--	--	--	--	--	--	--	--

The number of data items is specified with the last internal number of the data table in BCD format.

2) Clearing data tables.

- a) Put the NC in MDI mode.
- b) Set the NC memory write switch to enable.
- c) Push **【PARAM】** and select the PC screen.
- d) Push key **【N】** and the numerical keys to set 4999, then push **【INPUT】** .
- e) Push key **【P】** and the numerical keys to set 9999, then put **【INPUT】**, the data table is cleared to zero.
- f) Restore the NC memory write switch.

3) Setting and Displaying

Setting and display is the same as Item 5.4. But the numbers by displayed are added 4000 by the number specified by data table.

number specified by display at LCD	Number specified by data table	Address
4000	0	608
4001	1	609
.....
4030	30	638

6 Address Table

1. Capacity

Maximal number of input I/O: 96

Maximal number of output I/O: 64

Maximal number of LADDER steps : 2000

2. Address of Signal

- 000~015 output signals to machine tool
- 032~051 input signals from machine tool
- 064~ signals from CNC to PLC
- 096~ signals from PLC to CNC
- 200~ internal relays of PLC

3. Introduce of the address of the following signals being fixed

(1) 983M

#32

		*DECX				*-LX	*+LX
--	--	-------	--	--	--	------	------

*DECX: ZERO POINT RETURN DECELERATION OF X AXIS

*-LX: OVER-TRAVEL OF -X AXIS

*+LX: OVER-TRAVEL OF +X AXIS

#33

		*DECY				*-LY	*+LY
--	--	-------	--	--	--	------	------

*DECY: ZERO POINT RETURN DECELERATION OF Y AXIS

*-LY: OVER-TRAVEL OF -Y AXIS

*+LY: OVER-TRAVEL OF +Y AXIS

#34

		*DECZ				*-LZ	*+LZ
--	--	-------	--	--	--	------	------

*DECZ: ZERO POINT RETURN DECELERATION OF Z AXIS

*-LZ: OVER-TRAVEL OF -Z AXIS

*+LZ: OVER-TRAVEL OF +Z AXIS

#35

		*DEC4				*-L4	*+L4
--	--	-------	--	--	--	------	------

*DEC4: ZERO POINT RETURN DECELERATION OF 4 AXIS

*-L4: OVER-TRAVEL OF -4TH AXIS

*+L4: OVER-TRAVEL OF +4TH AXIS

#38

		*SP	*ESP				
--	--	-----	------	--	--	--	--

*SP: FEED HOLD

*ESP: EMERGENCY STOP

The signals from CNC to PLC is fixed. The meaning following:

#64

OP	SA	STL	SPL	ZP4	ZPZ	ZPY	ZPX
----	----	-----	-----	-----	-----	-----	-----

OP: AUTO RUNNING

SA: SERVO READY

STL: CYCLE START-UP

SPL: FEED HOLD

ZP4, ZPZ,ZPY,ZPX: ZERO POINT RETURN DECELERATION OF AXISES

#65

MA	FMF	SSP/TAP	SRV	DEN	RWD	RST	ALM
----	-----	---------	-----	-----	-----	-----	-----

MA: SYSTEM READY

FMF: READ FIX CYCLE AUXILIARY FUNCTION

SSP/TAP: SPINDLE STOP

SRV: SPINDLE REVERSE

RST: RESET

ALM: ALARM

#66

		DST	BT	TF	SF	EF	MF
--	--	-----	----	----	----	----	----

DST: MANUAL INPUT DATA START

BT: READ SECOND AUXILIARY FUNCTION CODE.

TF: READ T-CODE.

SF: READ S-CODE.

EF: EXTERNAL OPERATION FUNCTION.

MF: READ M-CODE.

#67

M28	M24	M22	M21	M18	M14	M12	M11
-----	-----	-----	-----	-----	-----	-----	-----

BCD CODE OF M FUNCTION OUTPUT

#68

R08	R07	R06	R05	R04	R03	R02	R01
-----	-----	-----	-----	-----	-----	-----	-----

R01~R08: SPINDLE SPEED CODE OF CONSTANT SURFACE SPEED CONTROL

#69

M00	M01	M02	M30	R12	R11	R10	R09
-----	-----	-----	-----	-----	-----	-----	-----

M00: PROGRAM STOP

M01: OPTIONAL BLOCK

M02: PROGRAM END

R09~R12: SPINDLE SPEED CODE OF CONSTANT SURFACE SPEED CONTROL

#70

S28	S24	S22	S21	S18	S14	S12/HIG	S11/LWG
-----	-----	-----	-----	-----	-----	---------	---------

BCD CODE OF S FUNCTION OUTPUT OR GEAR SELECTION

#71

T28	T24	T22	T21	T18	T14	T12	T11
-----	-----	-----	-----	-----	-----	-----	-----

BCD CODE OF T FUNCTION OUTPUT

#74

ERE/REN				SEND			
---------	--	--	--	------	--	--	--

ERE/REN: EXTERNAL DATA INPUT

#75

T48	T44	T42	T41	T38	T34	T32	T31
-----	-----	-----	-----	-----	-----	-----	-----

BCD CODE OF T FUNCTION OUTPUT

#78

B38	B34	B32	B31	B28	B24	B22	B21
-----	-----	-----	-----	-----	-----	-----	-----

THE SECOND AUXILIARY FUNCTION CODE OUTPUT

#79

B18	B14	B12	B11	ZP24	ZP2Z	ZP2Y	ZP2X
-----	-----	-----	-----	------	------	------	------

B11~B18: THE SECOND AUXILIARY FUNCTION CODE OUTPUT

ZP24: 4TH AXIS RETURN THE SECOND REFERENCE POINT

ZP2Z: Z AXIS RETURN THE SECOND REFERENCE POINT

ZP2Y: Y AXIS RETURN THE SECOND REFERENCE POINT

ZP2X: X AXIS RETURN THE SECOND REFERENCE POINT

The signals from PLC to NC are effective when the PLC is used. The meaning of signals is confirmed by NC. As follows,

#96

HX	*SVFX		*ITX	-X	+X		
----	-------	--	------	----	----	--	--

HX: X AXIS HANDLE

*SVFX: THE SEVOR OF X AXIS DISCONNECTION

-X: JOG -X

+X: JOG +X

#97

HY	*SVFY		*ITY	-Y	+Y		
----	-------	--	------	----	----	--	--

HY: Y AXIS HANDLE

*SVFY: THE SEVOR OF Y AXIS DISCONNECTION

-Y: JOG -Y

+Y: JOG +Y

#98

HZ	*SVFZ		*ITZ	-Z	+Z		
----	-------	--	------	----	----	--	--

HZ: Z AXIS HANDLE

*SVFZ: THE SEVOR OF Z AXIS DISCONNECTION

-Z: JOG -Z

+Z: JOG +Z

#99

H4	*SVF4		*IT4	-4TH	+4TH		
----	-------	--	------	------	------	--	--

H4: 4TH AXIS HANDLE

*SVFX: THE SEVOR OF 4TH AXIS DISCONNECTION

- 4TH: JOG -4TH AXIS

+ 4TH: JOG + 4TH AXIS

#100

MLK	DLK	ZNG	OVC	SBK	BDT1	DRN	AFL
-----	-----	-----	-----	-----	------	-----	-----

MLK: MACHINE LOCK

DLK: DISPLAY LOCK

ZNG: INSTRATION OF Z AXIS CANCEL

OVC: RATIO CANCEL

SBK: SINGLE BLOCK

BDT1: OPTIONAL SKIP BLOCK

DRN: DRY RUN

AFL: M.S.T FUNCTION LOCKED

#101

ZRN	SRN	ABS	SAR	FIN	ST	MIY	MIX
-----	-----	-----	-----	-----	----	-----	-----

ZRN: RETURN ZERO POINT

SRN: PROGRAM RESTART

ABS: HANDLE ABSOLUTE

SAR: SPINDLE SPEED ARRIVAL

FIN: M.S.T FUNCTION FINISH

ST: CYCLE START

MIY: MIRROR IMAGE OF Y AXIS

MIX: MIRROR IMAGE OF X AXIS

#102

ERS	RRW	*SP	*ESP	GST	SPC	SPB	SPA
-----	-----	-----	------	-----	-----	-----	-----

ERS: EXTERNAL RESTART

*SP: FEED KEEP

*ESP: EMERGENCY STOP

SPA, SPB, SPC: SPINDLE OVERRIDE

#103

RT	ROV2	ROV1	*JV16	*JV8	*JV4	*JV2	*JV1
----	------	------	-------	------	------	------	------

ROV2, ROV1: RAPID OVERRIDE

*JV1, *JV2, *JV4, *JV8, *JV16: JOG MODE FEEDRATE

#104

MP4	MP2	MP1	*FV16	*FV8	*FV4	*FV2	*FV1
-----	-----	-----	-------	------	------	------	------

MP1, MP2, MP4: HANDLE OVERRIDE

*FV1, *FV2, *FV4, *FV8, *FV16: FEED OVERRIDE

#105

KEY	EDT	MEM	T	D	J	H	S
-----	-----	-----	---	---	---	---	---

KEY: PROGRAM PROTECT

EDT: EDIT

MEM: MEMORY

T: DNC

D: HANDLE INPUT DATA

J: JOG MODE

H: HANDLE

S: STEP

#106

*SSTP	SOR						
-------	-----	--	--	--	--	--	--

*SSTP: SPINDLE STOP

SOR: SPINDLE ORIENTATION

#114

		BDT9	BDT8			GRB	GRA
--	--	------	------	--	--	-----	-----

BDT8, BDT9: OPTIONAL SKIP PROGRAM BLOCK

GRA, GRB: CONSTAN SURFACE SPEED CONTROL GEAR.

#116

ED7	ED6	ED5	ED4	ED3	ED2	ED1	ED0
-----	-----	-----	-----	-----	-----	-----	-----

ED0~ED15: EXTERNAL DATA INPUT

#117

ED15	ED14	ED13	ED12	ED11	ED10	ED9	ED8
------	------	------	------	------	------	-----	-----

#118

ESTB	EA6	EA5	EA4	EA3	EA2	EA1	EA0
------	-----	-----	-----	-----	-----	-----	-----

ESTB: EXTERNAL DATA INPUT

EA0~EA6: EXTERNAL ADDRES INPUT

#121

BDT7		BDT6		BDT5	BDT4	BDT3	BDT2
------	--	------	--	------	------	------	------

#122

UI7	UI6	UI5	UI4	UI3	UI2	UI1	UI0
-----	-----	-----	-----	-----	-----	-----	-----

UI0~UI15: USER'S MACRO PROGRAM

#123

UI15	UI14	UI13	UI12	UI11	UI10	UI9	UI8
------	------	------	------	------	------	-----	-----

Internal relays of PLC

The addresses of the relays can save the PLC middle value and the result of function instruction. User can define the address that not be used by CNC.

(1) Preset value at counter No.1

#596

--	--	--	--	--	--	--	--

BCD LOW

#597

--	--	--	--	--	--	--	--

BCD HIGH

(2) Preset value at counter No.2

#598

--	--	--	--	--	--	--	--

#599

--	--	--	--	--	--	--	--

(3) Nonvolatile sequence memory

#600

--	--	--	--	--	--	--	--

#601

--	--	--	--	--	--	--	--

#602

--	--	--	--	--	--	--	--

#603

--	--	--	--	--	--	--	--

(4) Current value at counter No.1

#604

--	--	--	--	--	--	--	--

#605

--	--	--	--	--	--	--	--

(5)、Current value at counter No.2

#606

--	--	--	--	--	--	--	--

#607

--	--	--	--	--	--	--	--

(2) 983T

#04

R08	R07	R06	R05	R04	R03	R02	R01
-----	-----	-----	-----	-----	-----	-----	-----

12-BIT CODE OF SPINDLE OUTPUT

#05

				R12	R11	R10	R9
--	--	--	--	-----	-----	-----	----

12-BIT CODE OF SPINDLE OUTPUT

#32

		*DECX				*-LX	*+LX
--	--	-------	--	--	--	------	------

*DECX: ZERO POINT RETURN DECELERATION OF X AXIS

*-LX: OVER-TRAVEL OF -X AXIS

*+LX: OVER-TRAVEL OF +X AXIS

#33

		*DECZ				*-LZ	*+LZ
--	--	-------	--	--	--	------	------

*DECZ: ZERO POINT RETURN DECELERATION OF Z AXIS

*-LZ: OVER-TRAVEL OF -Z AXIS

*+LZ: OVER-TRAVEL OF +Z AXIS

#43

	SKIP						
--	------	--	--	--	--	--	--

SKIP: SKIP

The meanings of the signals from CNC to PLC are confirmed by CNC. As follows,

#64

OP	SA	STL	SPL			ZPZ	ZPX
----	----	-----	-----	--	--	-----	-----

OP: AUTO RUNNING

SA: SERVO READY

STL: CYCLE START-UP

SPL: FEED HOLD

ZPZ, ZPX: ZERO POINT RETURN DECELERATION OF AXISES

#65

MA	M30	M02	M00	DEN		RST	ALM
----	-----	-----	-----	-----	--	-----	-----

MA: SYSTEM READY

M00, M02, M30: M CODE SIGNALS

RST: RESET

ALM: ALARM

#66

		DST		TF1	SF	M01	MF
--	--	-----	--	-----	----	-----	----

#67

M28	M24	M22	M21	M18	M14	M12	M11
-----	-----	-----	-----	-----	-----	-----	-----

BCD CODE OF M FUNCTION OUTPUT

#68

R08	R07	R06	R05	R04	R03	R02	R01
-----	-----	-----	-----	-----	-----	-----	-----

12 BIT S-CODE OUTPUT

#69

		ZP2Z	ZP2X	R12	R11	R10	R09
--	--	------	------	-----	-----	-----	-----

#70

S28	S24	S22	S21	S18	S14	S12	S11
-----	-----	-----	-----	-----	-----	-----	-----

BCD S-CODE OUTPUT

#71

T28	T24	T22	T21	T18	T14	T12	T11
-----	-----	-----	-----	-----	-----	-----	-----

BCD T-CODE OUTPUT

The meanings of the signals from PLC to CNC are confirmed by CNC. User can not define. As follows,

#96

HX	*SVFX		*ITX	-X	+X		
----	-------	--	------	----	----	--	--

HX: X AXIS HANDLE

*SVFX: THE SEVOR OF X AXIS DISCONNECTION

*ITX:

-X: JOG -X

+X: JOG +X

#97

HZ	*SVFZ		*ITZ	-Z	+Z		
----	-------	--	------	----	----	--	--

HZ: Z AXIS HANDLE

*SVFZ: THE SEVOR OF Z AXIS DISCONNECTION

*ITZ:

-Z: JOG -Z

+Z: JOG +Z

#98

			STLK				
--	--	--	------	--	--	--	--

#100

MLK	DLK	PRC	OVC	SBK	BDT	DRN	ALF
-----	-----	-----	-----	-----	-----	-----	-----

MLK: MACHINE LOCK

DLK: DISPLAY LOCK

ZNG: INSTRATION OF Z AXIS CANCEL

OVC: RATIO CANCEL

SBK: SINGLE BLOCK

BDT: OPTIONAL SKIP BLOCK

DRN: DRY RUN

AFL: M.S.T FUNCTION LOCKED

#101

ZRN	SRN		SAR	FIN	ST		MIX
-----	-----	--	-----	-----	----	--	-----

ZRN: RETURN ZERO POINT

SRN: PROGRAM RESTART

ABS: HANDLE ABSOLUTE

SAR: SPINDLE SPEED ARRIVAL

FIN: M.S.T FUNCTION FINISH

ST: CYCLE START

MIY: MIRROR IMAGE OF Y AXIS

MIX: MIRROR IMAGE OF X AXIS

#102

ERS		*SP	*ESP	GST	SPC	SPB	SPA
-----	--	-----	------	-----	-----	-----	-----

ERS: EXTERNAL RESTART

*SP: FEED KEEP

*ESP: EMERGENCY STOP

SPA, SPB, SPC: SPINDLE OVERRIDE

#103

RT	ROV2	ROV1	*OV16	*OV8	*OV4	*OV2	*OV1
----	------	------	-------	------	------	------	------

ROV2, ROV1: RAPID OVERRIDE

*OV1~*OV16: JOG MODE FEEDRATE

#104

MP4	MP2	MP1			SMZ	ABS	CDZ
-----	-----	-----	--	--	-----	-----	-----

MP1, MP2, MP4: HANDLE OVERRIDE

#105

KEY	EDT	MEM	T	MDI	JOG	HS	
-----	-----	-----	---	-----	-----	----	--

KEY: PROGRAM PROTECT

EDT: EDIT

MEM: MEMORY

T: DNC

MDI: HANDLE INPUT DATA

JOG: JOG MODE

HS: HANDLE

#106

*SSTP	SOR			GR4	GR3	GR2	GR1
-------	-----	--	--	-----	-----	-----	-----

*SSTP: SPINDLE STOP

SOR: SPINDLE ORIENTATIO

GR1~GR4: GEAR SELECTION

#107

			WM6	WN8	WN4	WN2	WN1
--	--	--	-----	-----	-----	-----	-----

Internal relays of PLC

The addresses of the relays can save the PLC middle value and the result of function instruction. User can define the address that not be used by CNC.

(1) Preset value at counter No.1

#596

--	--	--	--	--	--	--	--

BCD LOW

#597

--	--	--	--	--	--	--	--

BCD HIGH

(2) Preset value at counter No.2

#598

--	--	--	--	--	--	--	--

#599

--	--	--	--	--	--	--	--

(3) Nonvolatile sequence memory

#600

--	--	--	--	--	--	--	--

#601

--	--	--	--	--	--	--	--

#602

--	--	--	--	--	--	--	--

(4) Current value at counter No.1

#603

--	--	--	--	--	--	--	--

#604

--	--	--	--	--	--	--	--

(5) Current value at counter No.2

#605

--	--	--	--	--	--	--	--

#607

--	--	--	--	--	--	--	--

Section II GSK983M/T CNC System

PLC Programmer Software Introduction

1 Summary

The PLC ladder programmer can be edited with the PLC programmer software that is the configuration software of the upper machine in **GSK983 CNC System**, The software's the interface is the simple and direct, amity, the edit intelligence, and be easy to used.

The PLC ladder program software of the GSK983 M/T CNC System can run under the WINDOWS 2000/XP operating system. Edited the PLC ladder program can be stored as the file, and it can be printed.

This software has the features as follows:

(1)The ladder programmer can edit the component annotation, segment annotation and is convenient to reading procedure for the users.

(2)The user environment menu and shortcut key make the software easy to be operation.

(3)It can print the ladder and address defined table.

Relevant convention on the user file extension in this software:

I/O and Relay address table file	.DEF
Ladder program file	.LDD
Sequence programmer EPROM solidify file	.ROM
Instruction file get from the ladder conversion	.TXT

2 System Requirements

- OS : Windows2000/XP

- CPU: Pentium is 133 MHz or higher
- Hard disk: 200M or higher
- Memory: 32 MB RAM or higher
- Display: Standard VGA, 1024*768, 24-bit true color
- Other: Keyboard, Mouse

3 The interface description

3.1 The total interface

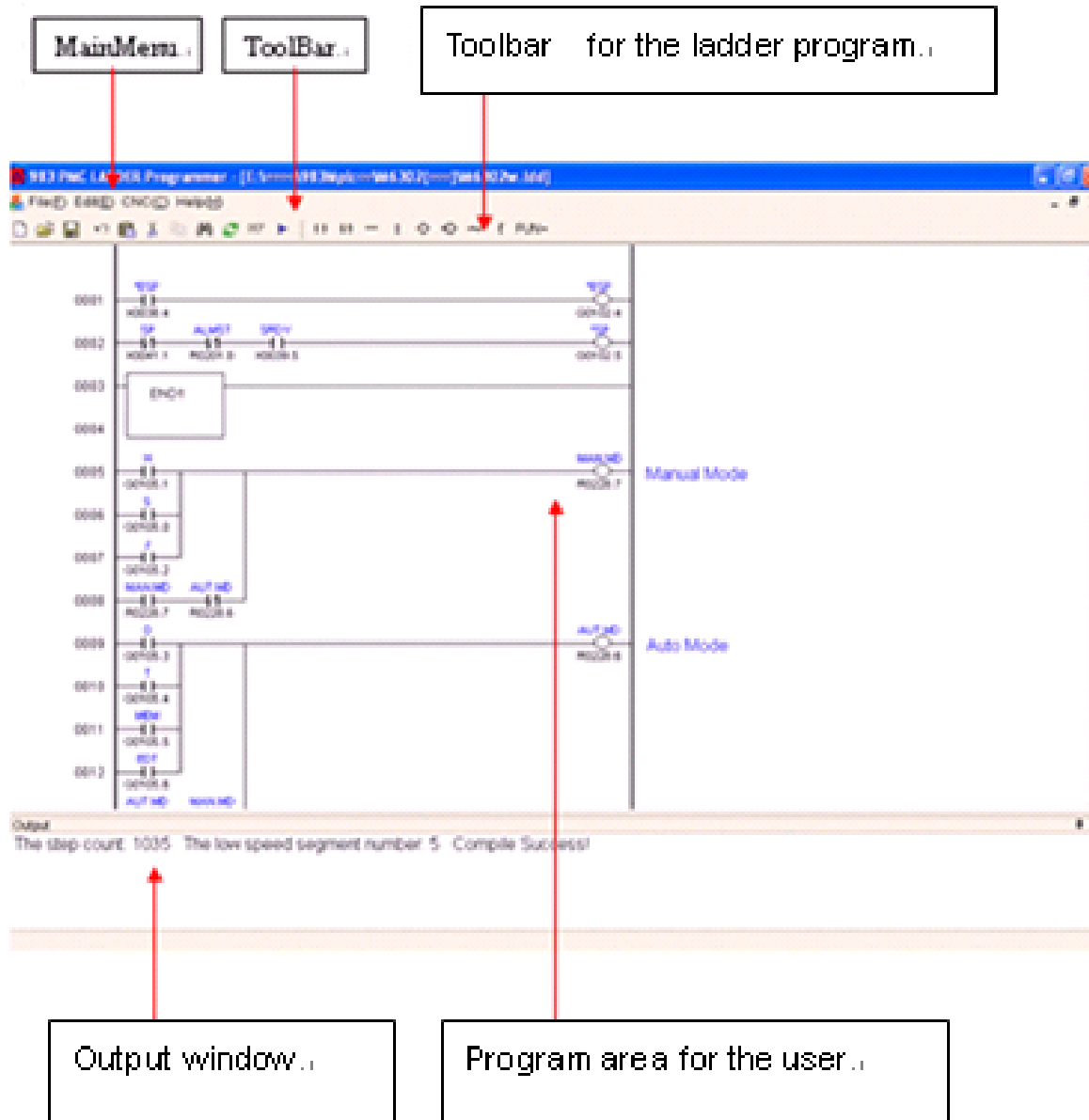


Fig. 3.1

- Main menu All software operations command
- Toolbar The common operations command
- Toolbar for the ladder program The operations command for the ladder edit
- Output window Output the ladder , compiler and edit an information
- Program area for the user At this area, the user can edit the ladder program

3.2 Menu command

3.2.1 [File] Menu

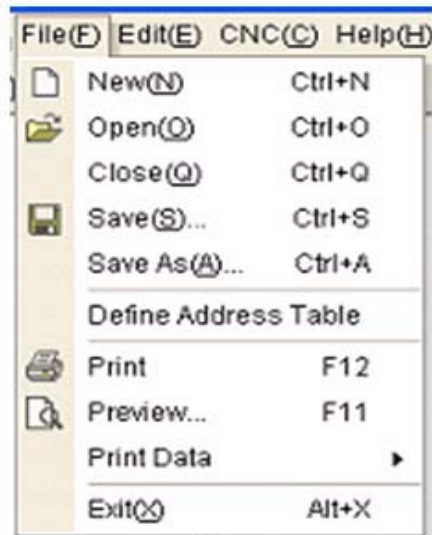



Fig. 3.2

- **[New]Command** Creates a new program (*.LDD).

Use the shortcut key: **[Ctrl+N]** or click shortcut toolbar : 

After executing that command, create a new file in the installation path that is default ,and its default filename is New*.ldd.

- **[Open]Command** Opens a program (*.LDD).

Use the shortcut key: **[Ctrl+O]** or click shortcut toolbar : 

After executing that command, the dialog box will appear, select the *.ldd file and

click **[Open]** button , then you can open the ladder file.

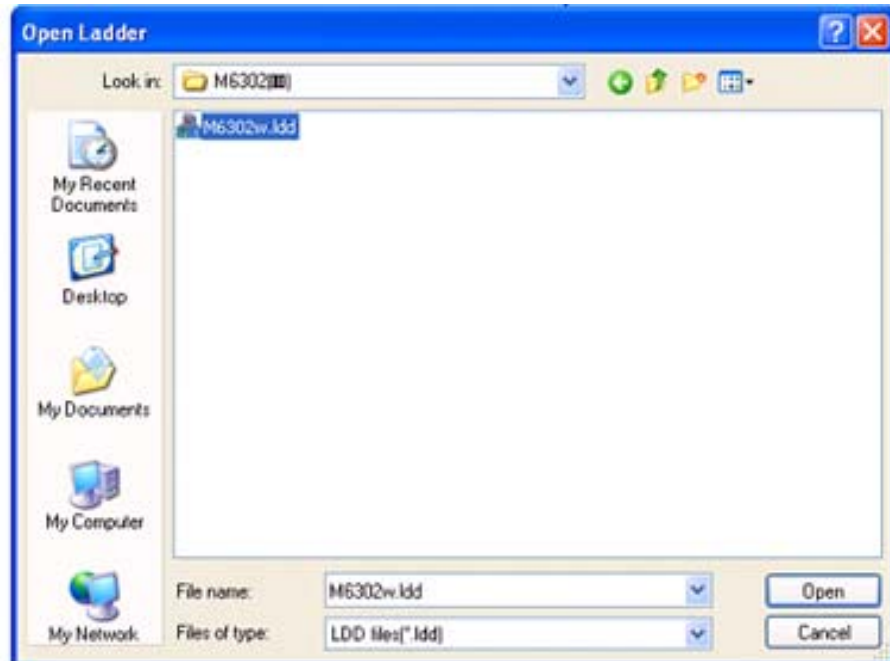


Fig. 3. 3

- **[Close]**Command Closes a program

[close] Close the current ladder file. If the ladder file have been edited, then the dialog box will popup and give the choice to save or not for the modified file. Click **[Yes]**, the current ladder file will be saved. Click **[No]**,the ladder file will keep the original content. Use the shortcut key:

[Ctrl+Q]

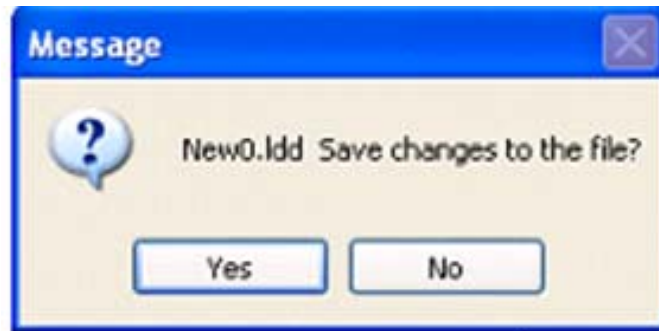



Fig. 3.4.

- **[Save]Command** Overwrites and Saves a program
[save]Save the current ladder file to the original path, use the shortcut key **[Ctrl+Q]**or click  in the toolbar.

- **[Save as]Command**
 The current ladder file saved as another backup, the dialog box will popup as the creating command after the execution of the command, then select a path and input a new filename, and click **[Save]**or**[Ok]**to be saved. use the shortcut key **[Ctrl+A]**.

- **[Address defined table]Command** Creates a new I/O and Relay address table (*.DEF) for a new program.

Every sequence program can be created a *. DEF file to store the address definition message. Once the ladder file was created by the user, there is a . DEF file that is same to the ladder filename and the path. If the ladder filename was modified, the name of the address defined table would be modified automatic correspondingly. If the user want to call up the original defined table, it only modify the defined table name to bring it into line with the ladder file name, and paste the name under the directory.

- **[Print]**Command After executing that command ,prints a program or address defined table,
use the shortcut key : **F12**
- **[Preview]**Command Displays Print Preview
use the shortcut key : **F11**
- **[Print Data]**Command It can be chosen to print the ladder or address defined table.

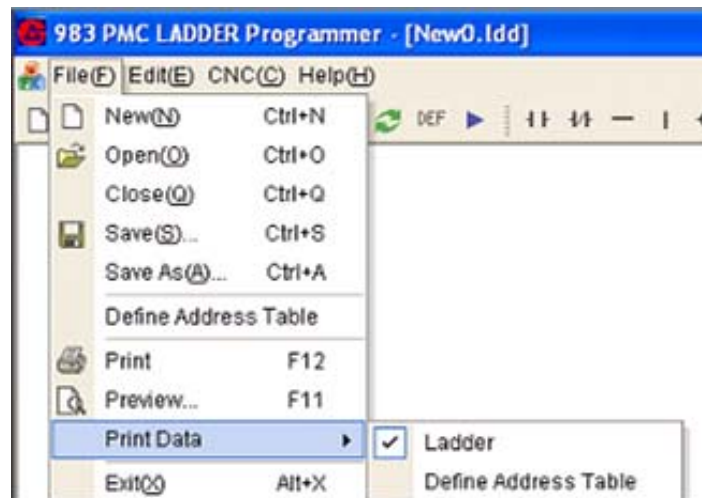


Fig. 3. 5

- **[Exit]**Command Closes this software,
Close the current application software. If the current file was not saved, the software would tell the user to save the current file or not.

3.2.2 [Edit]Menu

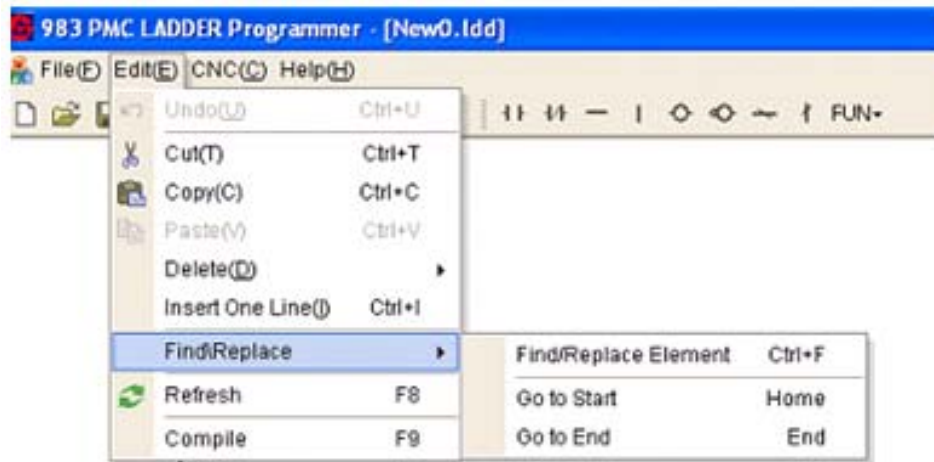


Fig. 3.6


- **[Undo]**Command Cancels the last action.

use the shortcut key **[Ctrl+U]** or click  in the toolbar.


- **[Cut]**Cut the selected range and copy to the clipboard.

Use the shortcut key **[Ctrl+T]** or click  in the toolbar.

- **[Copy]** Copy the selected range to the clipboard.

Use the shortcut key **[Ctrl+C]** or click  in the toolbar.

- **[Paste]** Paste the selected range to the appointed position, it support multiwindow paste.

Use the shortcut key **[Ctrl+V]** or click  in the toolbar.

- **[Delete]** Delete the selected range. Select the delete element ,then the note and line can be chosen to delete.

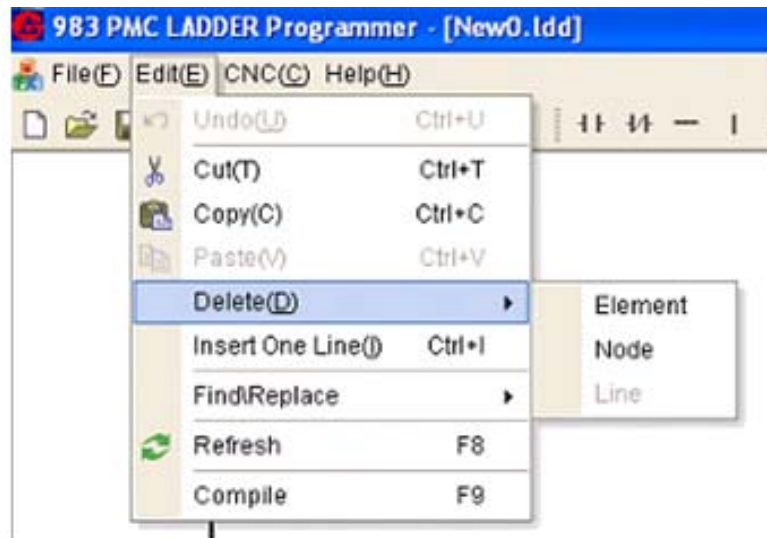


Fig. 3. 7

Select the **[Element]**, the current range of the ladder can be deleted. Use the shortcut

key delete or click  in the toolbar.

Select the **[Node]**, the current node can be deleted.

Select the whole **[Line]**, the current whole line or different lines can be selected.

- **[Insert one line]** Insert one line to the appointed location.
Use the shortcut key **[Ctrl+I]**.
- **[Find\replace]**The user can find the instruction or address that is according with the input condition to the ladder software,and the content have been found can replaced the appointed content.

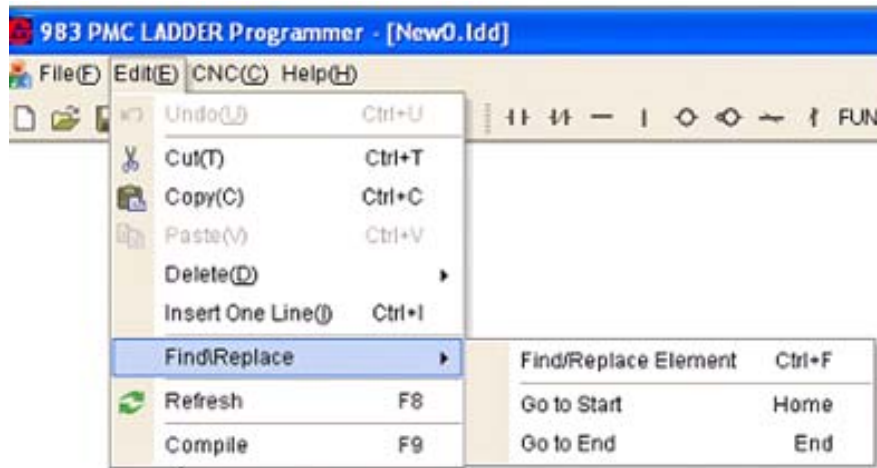


Fig. 3.8

- Execute the **[find\replace]** command, the dialogbox of the find\replace would popup.

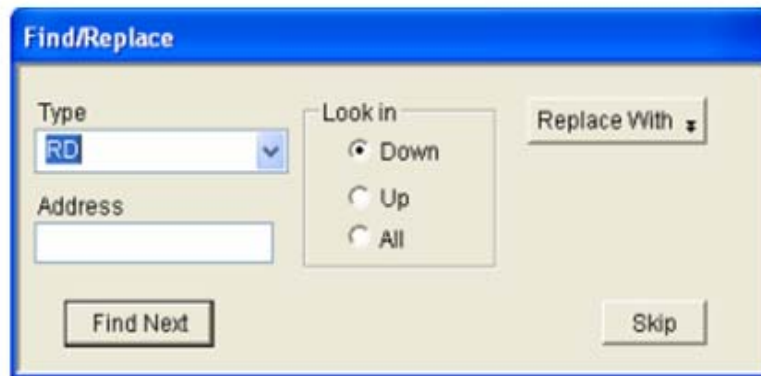


Fig. 3.9

- The find **[type]** command include **RD, RD.NOT, WRT, WRT.NOT** and function instruction selection. Input the address that needed to the element address, then click **[Find Next]** and the find work will go on. Also the direction can be selected. If the **[All]** selected, the same element type and the number of the address would appear in the popup dialogbox. When the function instruction command selected, input the function instruction name to find. Click **[Replace With]**, the dialogbox would be changed automatically.



Fig. 3.10

If the replace element type selected, input the replace address and click replace command, the element needed to be changed would be found and replace. That support all replace, but the function instruction except.

Use the shortcut key **[Ctrl+F]** or click  in the toolbar.

- **Click [Go to start]** command, the ladder software would go to start speediness.


Use the shortcut key: **Home**.

- **Click [Go to end]** command, the ladder software would go to end speediness.

Use the shortcut key **End**.

- **[Refresh]** Refresh the display in ladder interface. Click  in the toolbar.

- **[Compiler]** Translate and edit the current ladder program, if the ladder program was not right, the error information would be indicated in the interface output window, if it was right, the current file would be saved to *. ROM file.

Use the shortcut key **[F9]** or click  in the toolbar.

3.3.3 [CNC]Menu

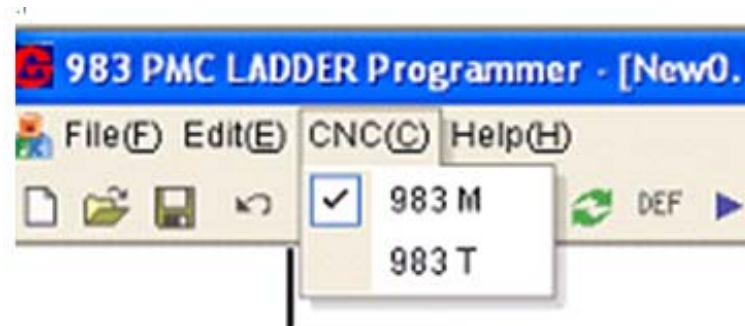


Fig. 3.11

Select relevant system configure according to the system be used, compile and create relevant *. ROM file.

3.2.4 [Help] Menu

Use the shortcut key **F1** to get the help.

3.3 Toolbar buttons

3.3.1 Toolbar

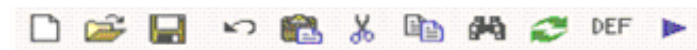













Fig. 3.12

-  Creates a new program (*.LDD).
-  Opens a program (*.LDD).
-  Overwrites and Saves a program
-  Cancels the last action.
-  Copies selected range
-  Cuts selected range
-  Pastes clipboard data

-  Find and Replace the data.
-  Refresh program area
-  Edit address defined table
-  Translate ladder program

3.3.2 Toolbar for the ladder program



Fig. 3.13

-  Add **RD** to the cursor,click it and the dialog box will popup.

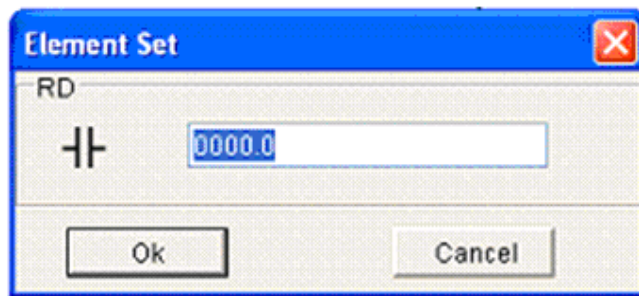









Fig. 3.14

Input the element's address. Use the shortcut key **F2**

-  Add **RD.NOT** to the cursor ,Use the shortcut key **F3**
-  Add horizontal link to the cursor ,Use the shortcut key **F4**
-  Add vertical link to the cursor ,Use the shortcut key **F5**
-  Add **WRT** to the last list of the cursor. Use the shortcut key **F6**
-  Add **WRT.NOT** to the last list of the cursor. Use the shortcut key **F7**
-  Delete the selected element, Use the shortcut key **Delete**.
-  Delete the selected node

FUN Click this button,the drop- down list will appear to select function command.

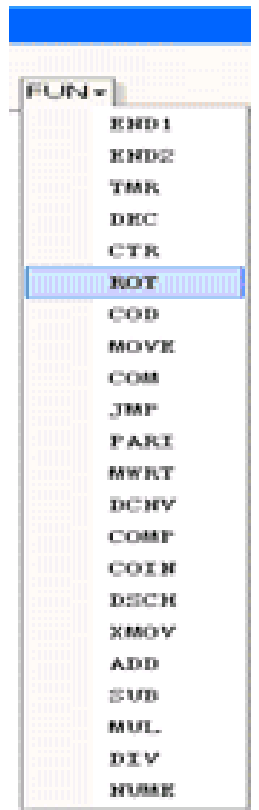


Fig. 3.15

When click that command,the parameter input dialog will popup except **END1**and **END2** function command.For example,click **ROT**,the dialog box will popup as follow:

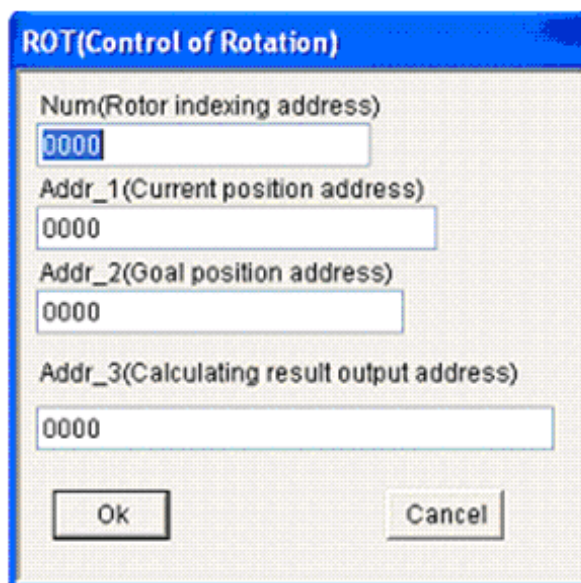


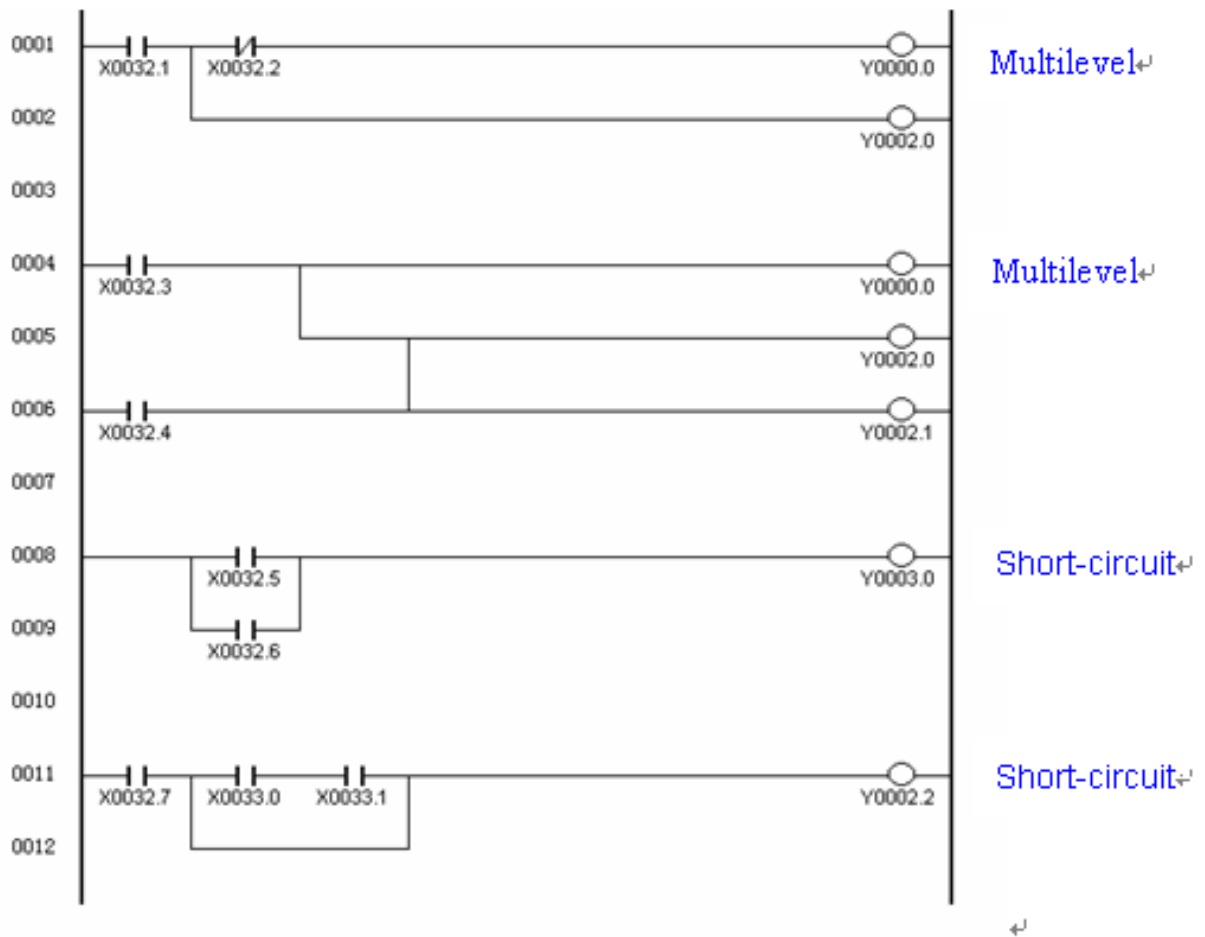
Fig. 3.16

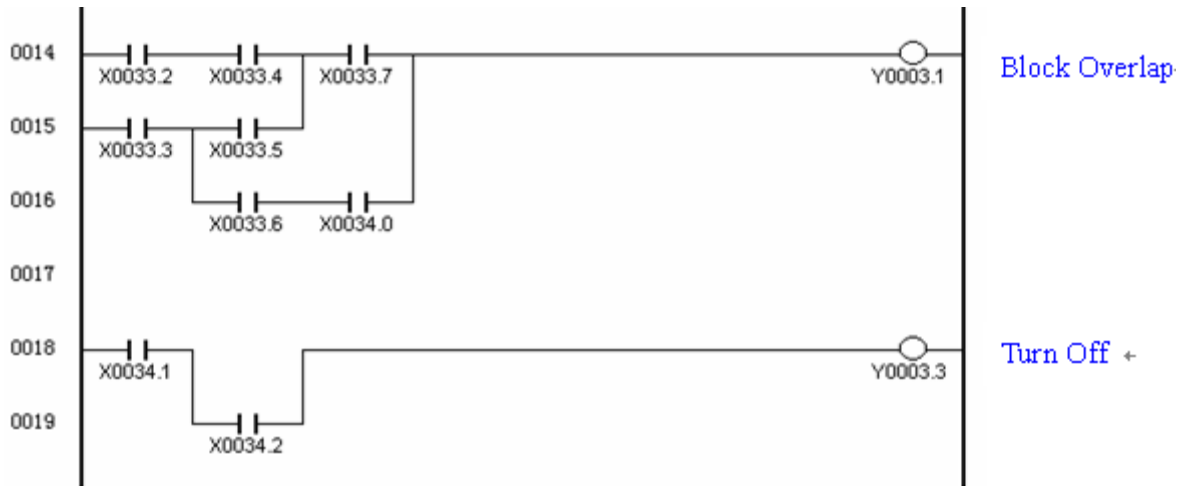
Input the right parameter, then click **[Ok]**, and add the function command to the fourth list of the cursor.

4 The limit of the Ladder program edit

1. The ladder program must be specified the **END1** and **END2** that is indicate the end of the high level sequence and the low level sequence, and the **END1** must be following the **END2**.

2. It is not support the multilevel output, only the juxtaposition output. and there are partial limits to the edit of the juxtaposition output. As shown in the figure from the line 4 to 6 is treated as the output error.





5 GSK983M/T P L C Programmer Procedure

First, design the ladder in the GSK983M/T ladder programmer software according to the control flow. Then select the compile button of the menu. It will display the success information in the output window and the next the productive ROM file should be written into the EEPROM with the programmer. In this way, the PLC program that has been edited is able to run in the GSK983M/T CNC. If there are something errors with the edit or logic, the alarm information will come up, and you should modify constantly according to the alarm instructions until it is correct.