Smarter. Greener. Together.

Industrial Automation Headquarters
Delta Electronics, Inc.
Taoyuan Technology Center
No.18, Xinglong Rd., Taoyuan City,
Taoyuan County 33068, Taiwan
TEL: 886-3-362-6301 / FAX: 886-3-371-6301

Asia
Delta Electronics (Jiangsu) Ltd.
Wujiang Plant 3
1688 Jiangxing East Road,
Wujiang Economic Development Zone
Wujiang City, Jiang Su Province,
People's Republic of China (Post code: 215200)
TEL: 86-512-6340-3008 / FAX: 86-769-6340-7290

Delta Greentech (China) Co., Ltd.
238 Min-Xia Road, Pudong District,
ShangHai, P.R.C.
Post code : 201209
TEL: 86-21-58635678 / FAX: 86-21-58630003

Delta Electronics (Japan), Inc.
Tokyo Office
2-1-14 Minato-ku Shibadaimon,
Tokyo 105-0012, Japan
TEL: 81-3-5733-1111 / FAX: 81-3-5733-1211

Delta Electronics (Korea), Inc.
1511, Byucksan Digital Valley 6-cha, Gasan-dong,
Geumcheon-gu, Seoul, Korea, 153-704
TEL: 82-2-515-5303 / FAX: 82-2-515-5302

Delta Electronics Int'l (S) Pte Ltd
4 Kaki Bukit Ave 1, #05-05, Singapore 417939
TEL: 65-6747-5155 / FAX: 65-6744-9228

Delta Electronics (India) Pvt. Ltd.
Plot No 43 Sector 35, HSIIDC
Gurgaon, PIN 122001, Haryana, India
TEL : 91-124-4874900 / FAX : 91-124-4874945

Americas
Delta Products Corporation (USA)
Raleigh Office
P.O. Box 12173,5101 Davis Drive,
Research Triangle Park, NC 27709, U.S.A.
TEL: 1-919-767-3800 / FAX: 1-919-767-8080

Delta Greentech (Brasil) S.A
Sao Paulo Office
Rua Itapeva, 26 - 3° andar Edificio Itapeva One-Bela Vista
01332-000-São Paulo-SP-Brazil
TEL: +55 11 3568-3855 / FAX: +55 11 3568-3865

Europe
Deltronics (The Netherlands) B.V.
Eindhoven Office
De Witbogt 15, 5652 AG Eindhoven, The Netherlands
TEL: 31-40-2592850 / FAX: 31-40-2592851

Delta CNC Solution

NC Series MLC Application Manual

www.delta.com.tw/ia

Smarter. Greener. Together.

Preface

Thank you for choosing this product. Before using the product, please read through this manual carefully in order to ensure the correct use of the product. In addition, please place the manual safely for quick reference whenever is needed.

This manual includes:

- List of MLC devices
- MLC basic instructions
- Introduction to MLC application instructions
- List of MLC application instructions
- MLC Special M, D commands and functions
- MLC application examples

Features of NC series controllers

- Built-in 32-bit highspeed dual CPU for multi-task execution and performance improvement
- Friendly HMI Interface
- Auto tuning interface are provided for optimizing the machine's performance efficiency
- CNC Soft software tools to facilitate the development of customized images
- Front USB interface to facilitate data access, data backup, and parameters copy
- Spindle forms for users to choose between communication type and analog voltage type
- Serial I/O modules for flexible I/O contacts configuration

How to use this manual:

This manual can be used as reference while learning NC controllers. It lists instructions, special M and D commands, as well as instructs how to edit MLC with application examples. Before using and setting this product, please read through this manual carefully.

DELTA technical services

Please consult the distributors or DELTA customer service center if any problem occurs.

Safety Precautions

- Please follow the instruction of pin definition when wiring. Ground is a must.
- When the power is being supplied, do not disconnect the controller, change the wiring, or touch the power supply.

Please pay close attention to the following safety precautions during inspecting, installation, operating, maintenance and troubleshooting.

The symbols of "**DANGER**", "**WARNING**" and "**STOP**" represent:

DANGER	**It indicates the potential hazards. It is possible to cause severe injury or fatal harm if not follow the instructions.**
WARNING	**It indicates the potential hazards. It is possible to cause minor injury or lead to serious damage of the product or even malfunction if not follow the instructions.**
STOP	**It indicates the absolute prohibited activity. It is possible to damage the product or cannot be used due to malfunction if not follow the instructions.**

Installation

DANGER	■ Please follow the installation instructions in this manual; otherwise it may cause damage to the equipment. ■ It is prohibited to expose the product to the environment containing water, corrosive gas, inflammable gas etc. Otherwise, electric shock or fire may occur.

Wiring

DANGER	■ Please connect the ground terminal to class-3 ground system (under 100 Ω). Poor grounding may result in electric shock or fire.

Operation

WARNING	■ Correctly plan out the I/O actions with MLC Editor Software, or abnormal results may occur. ■ Before operation, please properly adjust the parameter settings of the machine, otherwise it may cause abnormal operation. ■ Please ensure the emergency stop can be activated at any time, and avoid operating the machine in unprotected condition.

STOP	■ Do not modify wiring while power is being supplied. Otherwise, it may cause personal injury due to electric shock. ■ Never use a sharp-pointed object to touch the panel, as doing this might dent the screen and lead to malfunction of the controller.

Maintenance and Inspection

- While power is being supplied, do not disassemble the controller panel or touch the internal parts, otherwise electric shock may occur.
- Do not touch the ground terminal within 10 minutes after turning off the power, as the residual voltage may cause electric shock.
- Turn OFF the power first before replacing backup battery, and recheck the system settings afterwards.
- Do not block the vent holes during operation, as malfunction may easily occur due to poor ventilation.

Wiring Method

- Power supply: In order to avoid danger, use a 24 V_{DC} power supply for the controller and comply with the wire specification when wiring.
- Wiring materials: Use multi-stranded twisted-pair wires or multi-core shielded-pair wires to isolate all cables.
- The maximum cable length for remote I/O signals and DMCNET communication is 20 m and the maximum cable length for other signal cable is 10 m.
- To control the input and output signals, a 24 V_{DC} power is required for the controller I/O and remote I/O.

Wiring of Communication Circuit

- DMCNET wiring: The wiring materials should be in compliance with the standard specification.
- Please make sure the wiring between the controller and servo drive is tight and secure, as loose cables may cause abnormal operation.

For the differences among the various versions, please refer to DELTA's website for the latest information (http://www.delta.com.tw/industrialautomation/).

(This page is intentionally left blank.)

Table of Contents

1 MLC Devices

2 Basic Instructions

4 Description of Application Instructions

5 MLC Special M, D Commands and Functions

6 MLC Application Examples

(This page is intentionally left blank.)

MLC Devices

1

This chapter describes the functions of the MLC devices, and lists the devices' number and definition.

1

1.1 List of MLC devices

NC series MLC contains many different devices, which are listed as below:

1.1.1 All devices in MLC

<table>
<tr><th>Type</th><th>Device</th><th colspan="2">Item</th><th colspan="2">Range and number</th><th>Contents</th></tr>
<tr><td rowspan="8">Relay (bit)</td><td>X</td><td colspan="2">External input relay</td><td>0 ~ 33
256 ~ 511</td><td>Total 289 points</td><td>I/O</td></tr>
<tr><td>Y</td><td colspan="2">External output relay</td><td>0 ~ 27
256 ~ 511</td><td>Total 284 points</td><td>I/O</td></tr>
<tr><td>M</td><td colspan="2">Auxiliary relay</td><td>0 ~ 3071</td><td>Total 3072 points</td><td>I/O</td></tr>
<tr><td>A</td><td colspan="2">Alarm</td><td>0 ~ 511</td><td>Total 512 points</td><td>I/O</td></tr>
<tr><td>T</td><td colspan="2">Timer</td><td>0 ~ 255</td><td>Total 256 points</td><td>I/O</td></tr>
<tr><td rowspan="3">C</td><td rowspan="3">Counter</td><td>16-bit</td><td>0 ~ 63</td><td rowspan="3">Total 80 points</td><td rowspan="3">I/O</td></tr>
<tr><td>32-bit</td><td>64 ~ 77</td></tr>
<tr><td>32-bit high-speed</td><td>78 ~ 79</td></tr>
<tr><td rowspan="7">Register (word)</td><td>T</td><td>Timer</td><td>16-bit</td><td>0 ~ 255</td><td>Total 256 points</td><td>0 ~ 65535</td></tr>
<tr><td rowspan="3">C</td><td rowspan="3">Counter</td><td>16-bit</td><td>0 ~ 63</td><td rowspan="3">Total 80 points</td><td>0 ~ 65535</td></tr>
<tr><td>32-bit</td><td>64 ~ 77</td><td>-2147,483,648 ~ 2147,483,647</td></tr>
<tr><td>32-bit high-speed</td><td>78 ~ 79</td><td>-2147,483,648 ~ 2147,483,647</td></tr>
<tr><td>D</td><td>Data register</td><td>16-bit</td><td>0 ~ 1535</td><td>Total 1536 points</td><td>-32,768 ~ 32,767</td></tr>
<tr><td>V</td><td>Indirect reference register</td><td>16-bit</td><td>0 ~ 7</td><td>Total 8 points</td><td>-32,768 ~ 32,767</td></tr>
<tr><td>Z</td><td>Indirect reference register</td><td>16-bit</td><td>0 ~ 7</td><td>Total 8 points</td><td>-32,768 ~ 32,767</td></tr>
<tr><td rowspan="3">Indicator</td><td>N</td><td colspan="2">Loop indicator</td><td>0 ~ 7</td><td>Total 8 points</td><td></td></tr>
<tr><td>P</td><td colspan="2">Jump indicator</td><td>0 ~ 255</td><td>Total 256 points</td><td>None</td></tr>
<tr><td>I</td><td colspan="2">Interrupt indicator
(IX00 ~ IX07)
(IC00 ~ IC01)
(IR00 ~ IR23)</td><td>0 ~ 33</td><td>Total 34 points</td><td>None</td></tr>
<tr><td>Constant</td><td>K</td><td colspan="2">Decimal constant</td><td>N/A</td><td>N/A</td><td>N/A</td></tr>
<tr><td>Floating point</td><td>F</td><td colspan="2">Floating point</td><td>N/A</td><td>N/A</td><td>N/A</td></tr>
</table>

1.1.2 Settings of MLC devices

Device name		Non-outrage retaining					Outrage retaining	Point
X mechanic input signal (Bit)		On Board	MPG	Undefined	2nd panel	Remote		296
		X0~X27	X28~X33	X34~X63	X64~X255	X256~X511	None	
Y mechanic output signal (Bit)		Y0 ~ Y27		Y28~Y63	Y64~Y255	Y256~Y511	None	296
M auxiliary relay (Bit)		General use		Special M for system		Special M for MLC	M512 ~ M1023	3072
		M0 ~ M511		M1024 ~ M2335		M2816 ~ M3071		
A Alarm(Bit)		A0 ~ A511					None	512
T	Timer (Bit)	T0 ~ T199 (in unit of 100ms)		T200 ~ T255 (in unit of 10ms)			None	256
	Timer (Word)	T0~T255 (16-bit, range 0 ~ 65535)						
C Counter	(Bit)	C0 ~ C79					None	80
	Word or D Word	16-bit (count up)		32-bit (count up and down)				
		Range	0 ~ 32767	-2,147,483,648 ~ +2,147,483,647				
		C0 ~ C63		C64 ~ C797				
		None		M2944~M2957 Starting countdown afterwards				
D data register (word)		General use		Special D for system		Special D for MLC	D512~ D1023	1536
		D0 ~ D511 (-32768 ~ +32767)		MLC->NC	NC->MLC	For MLC		
				D1024 ~ D1125	D1336 ~ D1396	D1456 ~ D1535		
V register		V0~V7 (-32768 ~ +32768)					None	8
Z register		Z0~Z7 (-32768 ~ +32768)					None	8
Indicators		Function			Range			
N (Loop indicator)		For primary control loop			N0 ~ N7		None	8
P (Jump indicator)		For CJ · CALL			P0 ~ P255		None	256
I (Interrupt indicator)		For interruption	On Board hardware		IX00 ~ IX07		None	34
			Hardware counting		IC00 ~ IC01			
			Remote hardware		IR00 ~ IR23			
K Constant		Decimal constant		K-32,768~K+32,767(16-bit operation)			None	
				K-2,147,483,648~ K+2,147,483,647 (32-bit operation)			None	
F Floating point		Three decimal places		-99999.999 ~ 99999.999			None	

1.2 Values and constants

For different control purposes, NC series MLC applies following types of numeric values to perform the operation. See explanation below for the tasks and functions of each type of numeric value.

1.2.1 Binary number (BIN)

All the operations and storage of values in MLC are conducted in binary format. The binary number and terms used in this manual are described as below.

1. Bit: Bit is the basic unit of a binary value, either 1 or 0.
2. Nibble: Composed of four consecutive bits (e.g. bit0 ~ bit3). Presented as the values 0 ~ 15 in decimal number or 0 ~ F in hexadecimal number.
3. Byte: Composed of two consecutive nibbles (i.e. 8 bits, bit0 ~ bit7). Presented as 00 ~ FF in hexadecimal number.
4. Word: Composed of two consecutive bytes (i.e. 16-bit, bit0 ~ bit15). Presented as 0000 ~ FFFF in 4-digit hexadecimal number.
5. Double Word: Composed of two consecutive words (i.e. 32-bit, bit0 ~ bit31). Presented as 00000000 ~ FFFFFFFF in 8-digit hexadecimal number.

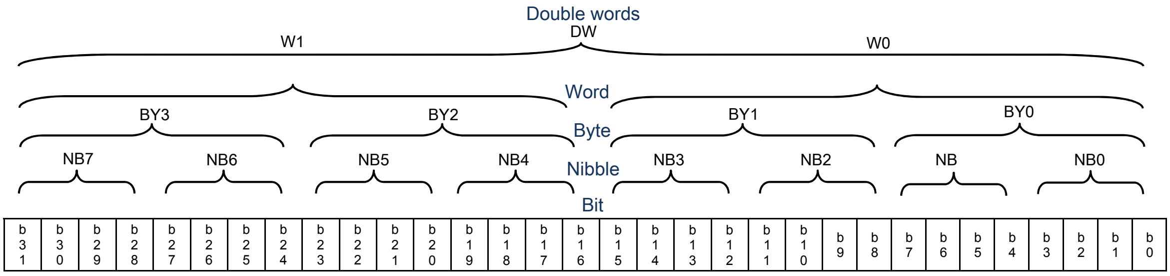


Figure 1.2.1.1 Bit, nibble, byte, word, double words in a binary system

1.2.2 Decimal number (DEC)

The numerical operations or storage in MLC are conducted in binary format. But MLC also uses decimal numbers in the occasions listed below:

1. (1) The No. of external input and output terminals are numbered in decimal number:
   External input device No.: X0 ~ X39, X64 ~ X511…
   External output device No.: Y0 ~ Y39, Y64 ~ Y511…
   (2) No. of devices M, A, T, C, D, V, Z, K, P, I and N, e.g. M10 and T30.
   (3) Set values for Timer T and Counter C, e.g. TMR T0 K50 (constant K).
   (4) As operands in application instructions, e.g. MOV K123 D0 (constant K).
2. K constant:
   Decimal number in MLC system are prefixed with the letter K in most cases. E.g. K100 is a decimal constant of value 100.
   Note: Constant K can be combined with bit devices X, Y, M and A to express data in the nibble, byte, word or double words format. E.g. K2Y10 and K4M100, here K1 refers to a 4-bit data and K2 ~ K4 refer to 8-bit, 12-bit and 16-bit data respectively.
3. Constant F:
   Floating point in MLC system are prefixed with the letter F in most cases, and used as operands in application instructions. E.g. FADD F12.3 F0 D0, F indicates a floating point constant.

1.3 Numbering and functions of the external input/output contacts (X / Y)

1.3.1 No. of the input/output contacts

In MLC system, the No. of input/output contacts begins at X0 and Y0, including On Board I/O, 2nd panel and remote I/O devices.

Device	Main board I/O	2nd panel I/O	Expansion I/O (Remote I/O)			
Input X	X0 ~ X27	X64 ~ X255	Station 1	Station 2	Station 3	Station 4
			X256 ~ X287	X288 ~ X319	X320 ~ X351	X352 ~ X383
			Station 5	Station 6	Station 7	Station 8
			X384 ~ X415	X416 ~ X447	X448 ~ X479	X480 ~ X511
Output Y	Y0 ~ Y27	Y64 ~ Y255	Station 1	Station 2	Station 3	Station 4
			Y256 ~ Y287	Y288 ~ Y319	Y320 ~ Y351	Y352 ~ Y383
			Station 5	Station 6	Station 7	Station 8
			Y384 ~ Y415	Y416 ~ Y447	Y448 ~ Y479	Y480 ~ Y511

Note: Initial No. of expansion I/O corresponds to its connection station. There are 8 stations in total with up to 256 points.

1

1.3.2 Functions of the input/output relays

MLC actions are enabled/disabled by input/output relays, of which the functions and state are described hereunder:

Function of input contact X:

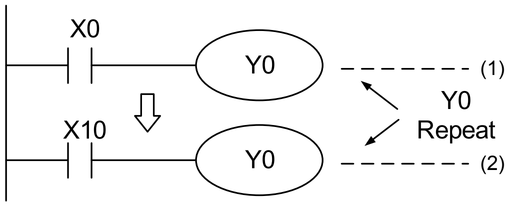
Input contact X connects to the input device and sets the input signal to MLC. The A or B contact of each input contact X has no use limit in the program. The ON/OFF state of input contact X varies only with the input device.

Function of output contact Y:

Output contact Y is used to send an ON/OFF signal to drive the load connected to output contact Y. There are two kinds of output contacts, relay and transistor. The A or B contact of output contact Y has no use limit in the program.

While using output contacts, please be aware of the following notes:

The No. of the output coil can only be used once in the program. Otherwise, according to the scan principle of MLC program, its output state will be determined by the last Y output circuit in the program (see the figure below).



Output of Y0 is determined by circuit (2), which means the ON/OFF state of X10 will determine the output state of Y0.

1.4 Numbering and functions of the auxiliary relay (M)

The system's auxiliary relay makes it easier for the user to edit MLC. The auxiliary relay starts to calculate at M0, and is used for general purpose, outage retaining, system special purpose, and MLC special purpose. Details are described below:

Auxiliary relay M		
General purpose	M0 ~ M511, 512 points, specific for non-outage retaining zone	Total 3,072 points
Outage retaining	M512 ~ M1023, 512 points, specific for outage retaining zone	
System special purpose	M1024 ~ M2335, 480 points, all non-outage for outage retaining	
MLC special purpose	M2816 ~ M3071, 256 points, all non-outage for outage retaining	

1.4.1 Functions of the auxiliary relay

Same as output relay Y, auxiliary relay M features output coil and A/B contacts, which have no use limit in the program. Auxiliary relay M can be used for combining control loop but it cannot directly drive the external load. There are three types of auxiliary relay:

1. General purpose auxiliary relay:
   If a power failure occurs during MLC operation, state of the general purpose auxiliary relay will be reset to OFF and remain OFF when power is resumed.
2. Outage retaining auxiliary relay:
   If a power failure occurs during MLC operation, state of the outage retaining auxiliary relay will be retained and its state will remain the same when power is resumed.
3. Special purpose auxiliary relay:
   Special purpose auxiliary relays are used for NC and MLC state or signal transmission. They are for an individual device's special function, e.g. M2832 is for C64 to count down. Each relay of this kind has its own specific function, and the undefined ones should not be used. Special purpose auxiliary relays are available to individual models.

While using auxiliary relays, please be aware of following notes:
When one auxiliary relay (M) is used for the output state, it can only be used once in the program. If it has been used more than once, the output state will be determined by the state of the last M output circuit in the program.

1.5 Numbering and functions of the custom alarm relay (A)

MLC provides the function of custom alarm relay, allowing users trigger it through the specified I/O actions while editing MLC. This function helps the user to detect self-defined irregularities. The custom alarm relay starts to calculate at A0.

Custom alarm relay A		
General purpose	A0 ~ A511, 512 points, specific for non-outage retaining zone	Total 512 points

1.5.1 Functions of the custom alarm relay

Same as output relay Y, custom alarm relay A features output coil and A/B contacts, which have no use limit in the program. Custom alarm relay A can be used for combining control loop but not for driving external loads. When a power failure occurs during MLC operation, state of the general purpose custom alarm relay will be reset to OFF. It remains in OFF state when power is resumed.

1.6 Numbering and functions of timers (T)

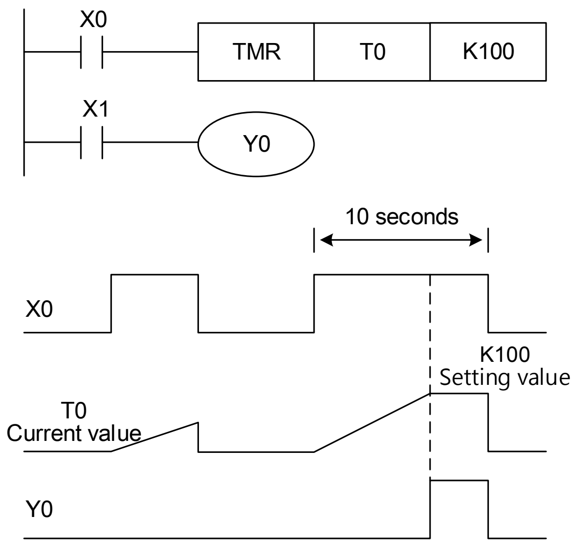
In MLC system, timer allows the user to start timing through the specified I/O actions while editing MLC. When the component reaches the set time, the planned actions will be executed. The timer starts to time at T0.

Timer T		
100ms, general purpose	T0 ~ T199, 200 points	Total 256 points
10ms, general purpose	T200 ~ T255, 55 points	

1.6.1 Setting the timer

The units of the timer are 10 ms and 100 ms and the counting method is counting up. The output coil will be enabled when the timer’s current value reaches the set value. The set value is a decimal constant (K), which can also be data register D.

Timer: The general purpose timer times once after the execution of each TMR instruction. When the current value of the timer reaches the set value, the output coil will be turned ON. See figure below.



When X0 = ON, the current value of the timer T0 counts up in unit of 100 ms. When the current value of T0 equals the set value K100, the output coil will be turned ON. When X0 = OFF or power failure occurs, the current value of timer T0 will be reset to 0 and the output coil T0 will be set to OFF.

Below are the methods to set the time value:

The actual set time in the timer = unit of timing × set values

By constant K: Directly assign constant K as the set time.

Indirectly set by constant D: Indirectly assign data register D as the set time.

1

1.7 Numbering and functions of counters (C)

Counters in MLC allow users to count via the specified I/O actions during MLC editing. When the component is triggered for a specific number of times, the planned actions will be executed. The counter starts to count at C0.

Counter C		
16-bit count up, general purpose	C0 ~ C63, 64 points, specific for non-outage retaining zone	Total 80 points
32-bit count up/down, general purpose	C64 ~ C77, 14 points, can be changed to count down with M2832 ~ M2845 settings	
32-bit count up/down, high speed	C78 ~ C79	

Item	16-bit counter	32-bit counter
Type	General purpose	General purpose
Direction	Up	Up and down
Set value	0 ~ 65,536	-2,147,483,648 ~ +2,147,483,647
Type of set value	Constant K or data register D	Constant K or data register D (assign both)
Change of the current value	Stop counting when the set value is reached.	Stop counting when the set value is reached.
Output contact	When the set value is reached, contact will be set to ON and remain its state.	When the set value is reached during counting up/down, contact will be set to ON and remain its state.
Reset	The RST instruction reset current value to 0 and contact to OFF.	
Contact action	Acts when the scanning is completed.	

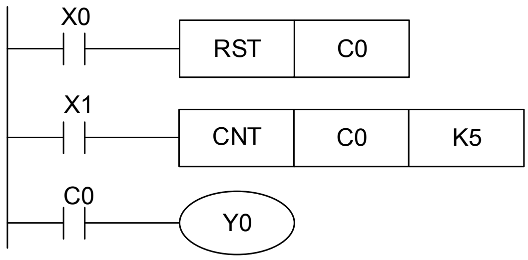
1.7.1 Functions of the counter

If the input signal of the counter's counting pulse changes from OFF to ON, the counter starts to count up by adding 1 to its current value. When the current value equals the set value, the output coil is turned on. The set value is a decimal constant (K) or a data register D. The functions of the 16-bit counter and 32-bit counter are described below:

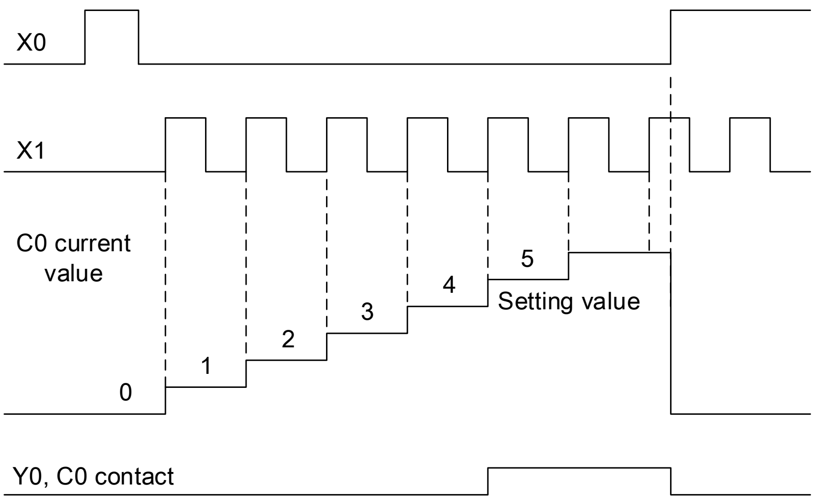
- 16-bit counter C0 ~ C63:
  The set value of the 16-bit counter is in the range of K0 ~ K65,536. (K0 and K1 have the same set value, and the output contact is set to ON at the first counting.) The counter’s set value can be either a direct constant K or an indirect data register D (excluding special data register D1024~D1536).

Example:

1. When X0 = ON, RST instruction is executed to set C0 to 0 and output contact to OFF.
2. When X1 changes from OFF to ON, C0 starts to count up by adding 1 to its current value.
3. When the current value of C0 reaches the set value K5, C0 contact will be turned ON and remain its state. C0 will accept the X1 trigger signal again only if X0 = ON and the current value of C0 is reset to 0.



Counter C0: Action sequence diagram



- 32-bit counter C64 ~ C77:

  The set value of the 32-bit general purpose counter is in the range of K-2,147,483,648 ~ K2, 147,483,647. The counter's counting direction (up or down) can be switched by special auxiliary relay M2834 ~ M2845. E.g. M2834 = OFF indicates C64 is for addition (counting up) and M2834 = ON for subtraction (counting down). The set value can be constant K or data register D, and the value can be positive or negative. Two consecutive data registers are required for one set value. Counter's current value changes from 2,147,483,647 to -2,147,483,648 when counting upward and from -2,147,483,648 to 2,147,483,647 when counting downward.

1

1.8 Numbering and functions of registers (D), (V), (Z)

1.8.1 Data register (D)

The data register is used for keeping 16-bit numeric data in the range of -32,768 ~ +32,767. The highest (most left) bit is a sign bit (+ or -). Two 16-bit registers can be combined into one 32-bit register. If one D is assigned for 32-bit register, the system automatically assigns D+1 for the same 32-bit register. The smaller D represents the lower 16 bits with the highest bit serving as the sign bit (+ or -). It can store numeric data in the range of -2,147,483,648 ~ +2,147,483,647. See example below.

Example: If D0 is assigned for 32-bit register, the system automatically assigns D1 for the same 32-bit register. D0 represents the lower 16 bits, while D1 represents the higher 16 bits.

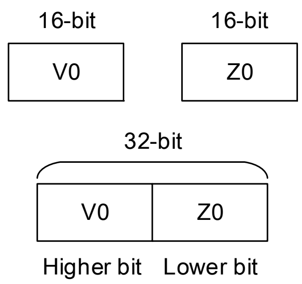
Data register D		
General purpose	D0 ~ D511, 512 points	Total 1,536 points
Outage retaining*	D512 ~ D1023, 512 points, specific for outage retaining zone	
Special purpose (MLC->NC)	D1024 ~ D1118, 95 points.	
Special purpose (NC->MLC)	D1336 ~ D1384, 49 points.	
MLC special purpose	D1456 ~ D1535, 80 points.	

The data register types fall into following four categories:

1. General purpose register:
   When MLC state turns from RUN to STOP, data will not be cleared. However, data will be reset to 0 if a power failure occurs.
2. Register for outage retaining D512 ~ D1023:
   When a power failure occurs, data from registers in this zone will not unchanged and remain the same if power is resumed. RST or ZRST instructions can be used to clear data contained in outage retaining registers.
3. Special purpose register:
   Each special purpose register has its own meaning and usage, mainly for system state storage, error message, and monitoring state.
4. Indirect reference register (V), (Z):
   Indirect reference registers are 16-bit registers. Total 16 points (V0 ~ V7, Z0 ~ Z7) are offered in MLC system. Assign register V for 32-bit registers, which stops register Z from being used any more. See Section 1.8.2 for more details.

1.8.2 Indirect index register (V), (Z)

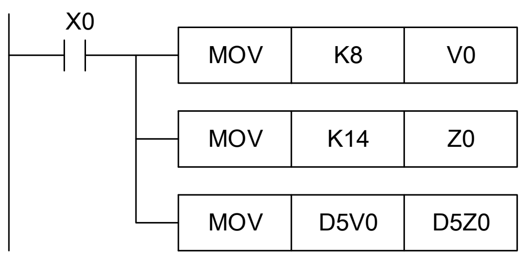
Same as general purposes registers, Register V, Z are 16-bit registers which can be edited and read. The user has to assign register V for a 32-bit registers. In such cases, register Z will automatically be included and cannot be used anymore. Otherwise, the 32-bit date in register V would be incorrect. See figure and table below.



Register V, Z in 32-bit register			
V0	Z0	V4	Z4
V1	Z1	V5	Z5
V2	Z2	V6	Z6
V3	Z3	V7	Z7

Same as general operands, indirect reference register can be used for data movement and comparison. But some instructions do not support this. Register V, Z can be used to modify operands.

Example:



When X0 = ON, firstly V0 = 8, Z0 = 14, secondly D5V0 = D (5 + 8) = D13, D5Z0 = D (5+14) = D19, and then the contents in D13 will be moved to D19.

1

1.9 Indicator (N), (P) and interrupt indicator (I)

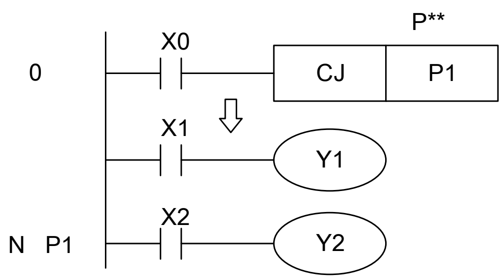
MLC system has indicator N, P, I. These indicators allow the system only to run user-designed programs while editing MLC. This can reduce errors caused by MLC scanning.

<table>
<tr><th colspan="5">Indicator</th></tr>
<tr><td>N</td><td colspan="2">For main control loop</td><td>N0 ~ N7, 8 points</td><td>Control point of the main control loop</td></tr>
<tr><td>P</td><td colspan="2">For CJ, CALL instruction</td><td>P0 ~ P255, 256 points</td><td>Position indicator of CJ, CALL</td></tr>
<tr><td rowspan="3">I</td><td rowspan="3">For interruption</td><td>On Board hardware interrupt</td><td>IX00 ~ IX07, 8 points</td><td rowspan="3">Position indicator of interrupt subroutine</td></tr>
<tr><td>Hardware counting interrupt</td><td>IC00 ~ IC01, 2 points</td></tr>
<tr><td>Remote I/O hardware interrupt</td><td>IR00 ~ IR23, 24 points</td></tr>
</table>

1. Indicator N, P
   - Indicator N: Work with MC instruction (the main control initial instruction) and MCR instruction. While running MC, instructions between MC and MCR run in a normal manner.
   - Indicator P: Work with application instructions API 00 CJ, API 01 CALL, and API 02 SRET. See Chapter 4 for descriptions on CJ, CALL, and SRET instructions.

Example 1:

When X0 = ON, the program jumps from address 0 to N (the assigned indicator P1), and continues its execution. The addresses in between will not be executed. When X0 = Off, the program executes from address 0 downward and CJ instruction will not be executed.

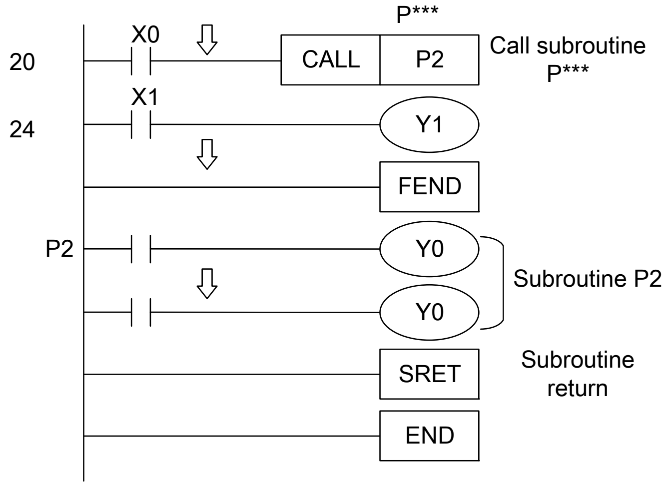


CJ conditional Jump

Example 2:

When X0 = ON, the program executes CALL instruction and jumps to the subroutine specified by indicator P2. When running SRET, the program returns to address 24 and keeps on execution.

CALL for subroutine calling and SRET for subroutine ending



2. Interrupt indicator I

Interrupt indicator I works with application instruction API 04 EI, API 05 DI, and API 03 IRET. See Chapter 4 for details. The interrupt actions are executed by EI instruction (enabling interrupt), DI instruction (disabling interrupt), and IRET instruction (interruption return).

- External interrupt:
  When the input contact X0 ~ X7 is rising-edge or falling-edge triggered by the input signal, the currently running program will complete its execution and then conduct interruption. The program will jump to the specified interruption subroutine indicator IX00(X0), IX01(X1), IX02(X2), IX03(X3), IX04(X4), IX05(X5), IX06(X6), IX07(X7) and execute IR00 ~ 23 (Remote X256 ~ 258, X288 ~ 290, X320 ~ 322…). IR00 ~ 23 correspond to the first three INPUT of the 8 Remote I/O respectively. When IRET instruction is executed, the program returns to the original position before the interruption and continues its execution.
- Interrupt when counting reaches set value:
  When the comparison instruction API 53 DHSCS of the high speed counter reaches its set value, the program will stop the currently running instruction and jump to the specified interrupt subroutine to execute interrupt indicator IC00 and IC01.

(This page is intentionally left blank.)

Basic Instructions 2

This chapter provides detailed descriptions about MLC basic instructions along with the application methods.

2.1 Summary of basic instructions

NC series MLC applies many different basic instructions. This section lists all the basic instructions along with their functions, operands, execution speed and STEP.

■ General purpose instructions

Instruction code	Function	Operand	Execution speed (us)	STEP
LD	Load A contact	X, Y, M, A, T, C	-	1 ~ 2
LDI	Load B contact	X, Y, M, A, T, C	-	1 ~ 2
AND	Serial connect A contact	X, Y, M, A, T, C	-	1 ~ 2
ANI	Serial connect B contact	X, Y, M, A, T, C	-	1 ~ 2
OR	Parallel connect A contact	X, Y, M, A, T, C	-	1 ~ 2
ORI	Parallel connect B contact	X, Y, M, A, T, C	-	1 ~ 2
ANB	Serial connect loop block	-	-	1
ORB	Parallel connect loop block	-	-	1
MPS	Saves it in stack	-	-	1
MRD	Stack retrieval (indicator remain intact)	-	-	1
MPP	Read stack	-	-	1

■ Output instructions

Instruction code	Function	Operand	Execution speed (us)	STEP
OUT	Driving coil	Y, A, M	-	1 ~ 2
SET	Action remains (ON)	Y, A, M	-	1 ~ 2
RST	Clear contact or register	Y, M, A, T, C, D, V, Z	-	1 ~ 2

■ Timer and counter

Instruction code	Function	Operand	Execution speed (us)	STEP
TMR	16-bit timer	T-K or T-D	9.6	3
CNT	16-bit counter	C-K or C-D (16-bit)	12.8	3
DCNT	32-bit counter	C-K or C-D (32-bit)	14.3	3 ~ 4

■ Primary control instructions

Instruction code	Function	Operand	Execution speed (us)	STEP
MC	Connection of serial contacts	N0 ~ N7	5.6	1
MCR	Disconnection of serial contacts	N0 ~ N7	5.7	1

■ Contact's rising/falling-edge detection instructions

Instruction code	Function	Operand	Execution speed (us)	STEP
LDP	Start of rising-edge detection	X, Y, M, A, T, C	-	2
LDF	Start of falling-edge detection	X, Y, M, A, T, C	-	2
ANDP	Serial connection of rising-edge detection	X, Y, M, A, T, C	-	2
ANDF	Serial connection of falling-edge detection	X, Y, M, A, T, C	-	2

Instruction code	Function	Operand	Execution speed (us)	STEP
ORP	Parallel connection of rising-edge detection	X, Y, M, A, T, C	-	2
ORF	Parallel connection of falling-edge detection	X, Y, M, A, T, C	-	2

- Upper and lower differential output instructions

Instruction code	Function	Operand	Execution speed (us)	STEP
PLS	Upper differential output	Y, M, A	-	1 ~ 2
PLF	Lower differential output	Y, M, A	-	1 ~ 2

- Program ends

Instruction code	Function	Operand	Execution speed (us)	STEP
END	Program ends	None	-	1

- Other instructions

Instruction code	Function	Operand	Execution speed (us)	STEP
NOP	Null action	None	-	1
INV	Invert operation result	None	-	1
P	Indicator	P0~P255	-	1
I	Interrupt indicator	IX□□, IC□□, IR□□ (Please refer to chapter 1 for the value of □□)	-	1

2.2 Description of basic instructions

This section gives detailed information about the function, operand, instruction description, application method and example of each basic instruction.

■ LD: Load A contact

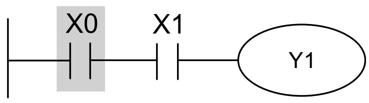
Instruction	Function	Model
LD	Load A contact	NC Series

Operand							
X0 ~ X39 X64 ~ X511	Y0 ~ Y39 Y64 ~ Y51	M0 ~ M3,071	A0 ~ A511	T0 ~ T255	C0 ~ C77	D0 ~ D1,535	V, Z
▪	▪	▪	▪	▪	▪	-	-

Instruction description:

The LD instruction applies to the starting A contact of a left bus or a starting A contact in loop block. It saves the current value and stores the acquired contact state into a cumulative register.

Example:



LD(X0) Ladder diagram

■ LDI: Load B contact

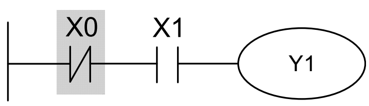
Instruction	Function	Model
LDI	Load B contact	NC Series

Operand							
X0 ~ X39 X64 ~ X511	Y0 ~ Y39 Y64 ~ Y51	M0 ~ M3,071	A0 ~ A511	T0 ~ T255	C0 ~ C77	D0 ~ D1,535	V, Z
▪	▪	▪	▪	▪	▪	-	-

Instruction description:

The LDI instruction applies to the starting B contact of a left bus or a starting B contact in the loop block. It saves the current value and stores the acquired contact state into a cumulative register.

Example:



LDI(X0) Ladder diagram

■ AND: Serial connect A contact

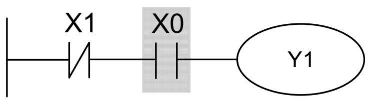
Instruction	Function	Model
AND	Serial connect A contact	NC Series

Operand							
X0 ~ X39 X64 ~ X511	Y0 ~ Y39 Y64 ~ Y51	M0 ~ M3,071	A0 ~ A511	T0 ~ T255	C0 ~ C77	D0 ~ D1,535	V, Z
▪	▪	▪	▪	▪	▪	-	-

Instruction description:

The AND instruction serial connects A contacts. It reads the current state of the given serial contacts and executes the AND operation on the acquired data together with the results from previous logic operations. The final result will be stored in a cumulative register.

Example:



AND(X0) Ladder diagram

■ ANI: Serial connect B contact

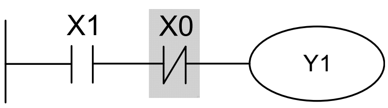
Instruction	Function	Model
ANI	Serial connect B contact	NC Series

Operand							
X0 ~ X39 X64 ~ X511	Y0 ~ Y39 Y64 ~ Y51	M0 ~ M3,071	A0 ~ A511	T0 ~ T255	C0 ~ C77	D0 ~ D1,535	V, Z
▪	▪	▪	▪	▪	▪	-	-

Instruction description:

The ANI instruction serial connects B contacts. It reads the current state of the given serial contacts and executes the AND operation on the acquired data together with the results from previous logic operations. The final result will be stored in a cumulative register.

Example:



ANI(X0) Ladder diagram

■ OR: Parallel connect A contact

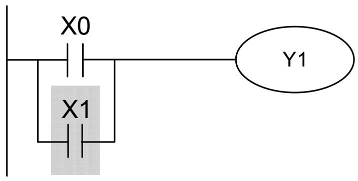
Instruction	Function	Model
OR	Parallel connect A contact	NC Series

Operand							
X0 ~ X39 X64 ~ X511	Y0 ~ Y39 Y64 ~ Y51	M0 ~ M3,071	A0 ~ A511	T0 ~ T255	C0 ~ C77	D0 ~ D1,535	V, Z
▪	▪	▪	▪	▪	▪	-	-

Instruction description:

The OR instruction parallel connects A contacts. It reads the current state of the given serial contacts and executes the OR operation on the acquired data together with the results from previous logic operations. The final result will be stored in a cumulative register.

Example:



OR(X1) Ladder diagram

■ ORI: Parallel connect B contact

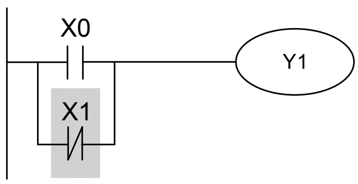
Instruction	Function	Model
ORI	Parallel connect B contact	NC Series

Operand							
X0 ~ X39 X64 ~ X511	Y0 ~ Y39 Y64 ~ Y51	M0 ~ M3,071	A0 ~ A511	T0 ~ T255	C0 ~ C77	D0 ~ D1,535	V, Z
▪	▪	▪	▪	▪	▪	-	-

Instruction description:

The ORI instruction parallel connects B contacts. It reads the current state of the given serial contacts and executes OR operation on the acquired data together with the results from previous logic operations. The final result will be stored in a cumulative register.

Example:



ORI(X1) Ladder diagram

■ ANB: Serial connect loop block

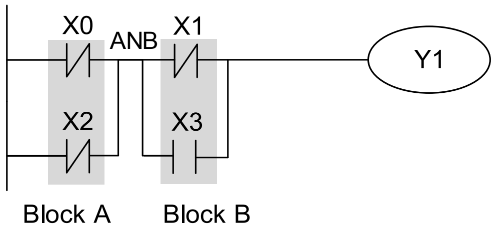
Instruction	Function	Model
ANB	Serial connect loop block	NC Series

Operand							
X0 ~ X39 X64 ~ X511	Y0 ~ Y39 Y64 ~ Y51	M0 ~ M3,071	A0 ~ A511	T0 ~ T255	C0 ~ C77	D0 ~ D1,535	V, Z
-	-	-	-	-	-	-	-

Instruction description:

The ANB instruction executes the AND operation on previously saved logic operation result with the current value in a cumulative register.

Example:



ANB(X0+X2), (X1+X3) Ladder diagram

■ ORB: Parallel connect loop block

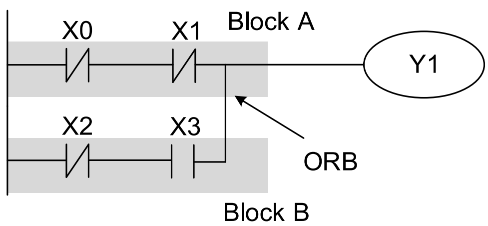
Instruction	Function	Model
ORB	Parallel connect loop block	NC Series

Operand							
X0 ~ X39 X64 ~ X511	Y0 ~ Y39 Y64 ~ Y51	M0 ~ M3,071	A0 ~ A511	T0 ~ T255	C0 ~ C77	D0 ~ D1,535	V, Z
-	-	-	-	-	-	-	-

Instruction description:

The ORB instruction executes the OR operation on previously saved logic operation result with the current value in a cumulative register.

Example:



ORB(X0+X1), (X2+X3) Ladder diagram

■ MPS: Save in stack

Instruction	Function	Model
MPS	Save in stack	NC Series

Operand							
X0 ~ X39 X64 ~ X511	Y0 ~ Y39 Y64 ~ Y51	M0 ~ M3,071	A0 ~ A511	T0 ~ T255	C0 ~ C77	D0 ~ D1,535	V, Z
-	-	-	-	-	-	-	-

Instruction description:

The MPS instruction saves the current value contained in the cumulative register in a stack. (Stack index increases by 1)

■ MRD: Read stack (Stack index remain intact)

Instruction	Function	Model
MRD	Read stack (Stack index remain intact)	NC Series

Operand							
X0 ~ X39 X64 ~ X511	Y0 ~ Y39 Y64 ~ Y51	M0 ~ M3,071	A0 ~ A511	T0 ~ T255	C0 ~ C77	D0 ~ D1,535	V, Z
-	-	-	-	-	-	-	-

Instruction description:

The MRD instruction reads the current value contained in the stack and saves it in a cumulative register. (Stack index remains intact)

■ MPP: Read stack (Stack index decreases by 1)

Instruction	Function	Model
MPP	Read stack	NC Series

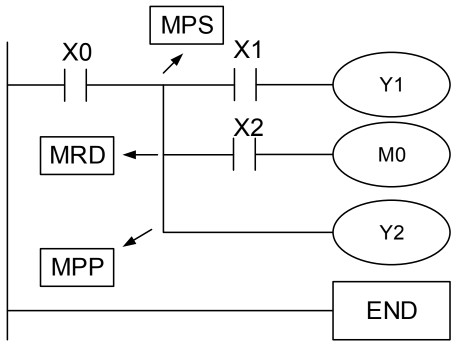
Operand							
X0 ~ X39 X64 ~ X511	Y0 ~ Y39 Y64 ~ Y51	M0 ~ M3,071	A0 ~ A511	T0 ~ T255	C0 ~ C77	D0 ~ D1,535	V, Z
-	-	-	-	-	-	-	-

Instruction description:

The MPP instruction retrieves the last saved logic operation result and saves it in a cumulative register. (Stack index decreases by 1)

2

Example:



MPS, MRD, MPP Ladder diagram

■ OUT: Drives coil

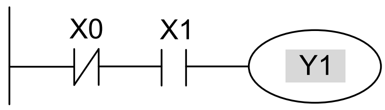
Instruction	Function	Model
OUT	Drives coil	NC Series

Operand							
X0 ~ X39 X64 ~ X511	Y0 ~ Y39 Y64 ~ Y51	M0 ~ M3,071	A0 ~ A511	T0 ~ T255	C0 ~ C77	D0 ~ D1,535	V, Z
-	▪	▪	▪	-	-	-	-

Instruction description:

The OUT instruction outputs the logic operation result before the OUT instruction to a specified device.

Example:



OUT(Y1) Ladder diagram

■ SET: Fix actions (ON)

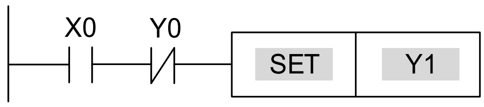
Instruction	Function	Model
SET	Fix actions (ON)	NC Series

Operand							
X0 ~ X39 X64 ~ X511	Y0 ~ Y39 Y64 ~ Y51	M0 ~ M3,071	A0 ~ A511	T0 ~ T255	C0 ~ C77	D0 ~ D1,535	V, Z
-	▪	▪	▪	-	-	-	-

Instruction description:

When SET instruction is executed, the specified bit will be on. The RST instruction can be used to set this bit to off. If the SET instruction is not executed, state of the specified bit remains the same.

Example:



SET(Y1) Ladder diagram

■ RST: Clear contacts or registers

Instruction	Function	Model
RST	Clear contacts or registers	NC Series

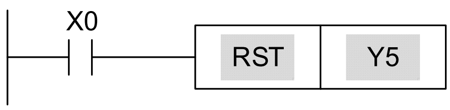
Operand							
X0 ~ X39 X64 ~ X511	Y0 ~ Y39 Y64 ~ Y51	M0 ~ M3,071	A0 ~ A511	T0 ~ T255	C0 ~ C77	D0 ~ D1,535	V, Z
-	▪	▪	▪	▪	▪	▪	▪

Instruction description:

See the table below for actions of devices driven by RST instruction.

Device	State
S, Y, M	Both coils and contacts are set to OFF.
T, C	Current timing and counting data are reset to 0 while coils and contacts are set to OFF.
D, V, Z	Content values are reset to 0.

Example:



RST(Y5) Ladder diagram

■ TMR: 16-bit timer

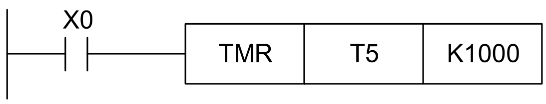
Instruction	Function	Model
TMR	16-bit timer	NC Series

Operand	
RST	T0 ~ T255; K0 ~ K32,767
RST	T0 ~ T255; D0 ~ D1,536

Instruction description:

After a TMR instruction is executed, the timer assigned by it turns ON and starts timing. If the set time is reached, the assigned timer is set to ON. When TMR instruction stops executing, the assigned timer will be reset to 0.

Example:



TMR(T5) Ladder diagram

■ CNT: 16-bit counter

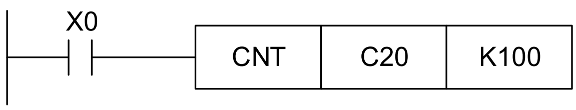
Instruction	Function	Model
CNT	16-bit timer	NC Series

Operand	
C-K	C0~C63; K0~K65,536
C-D	C0~C63; D0~D1,536

Instruction description:

When CNT instruction changes from OFF to ON, the coil assigned by the counter will be switched from OFF to ON, leading to its counting value increasing by 1. After the set value is reached, the contacts and counting values of the counter remain unchanged even when more counting pulse inputs are received. An RST instruction is required to restart counting or clear the value.

Example:



CNT(C20) Ladder diagram

■ DCNT: 32-bit counter

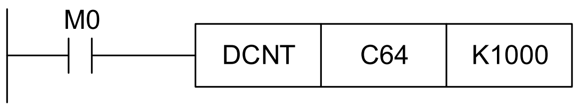
Instruction	Function	Model
DCNT	32-bit timer	NC Series

Operand	
C	C64 ~ C77
C-D&C-K	C64 ~ C77; D0 ~ D1,536; K-2,147,483,648 ~ K2,147,483,647

Instruction description:

The DCNT is the instruction for enabling the 32-bit counters C64 ~ C77. When DCNT instruction changes from OFF to ON, the counter's current value will increase or decreases by 1, and the up and down values of the counter are determined by the state of special M (M2944~M2957).

Example:



DCNT(C64) Ladder diagram

■ MC/MCR: Connection/disconnection of common serial contacts

Instruction	Function	Model
MC/MCR	Connection/disconnection of common serial contacts	NC Series

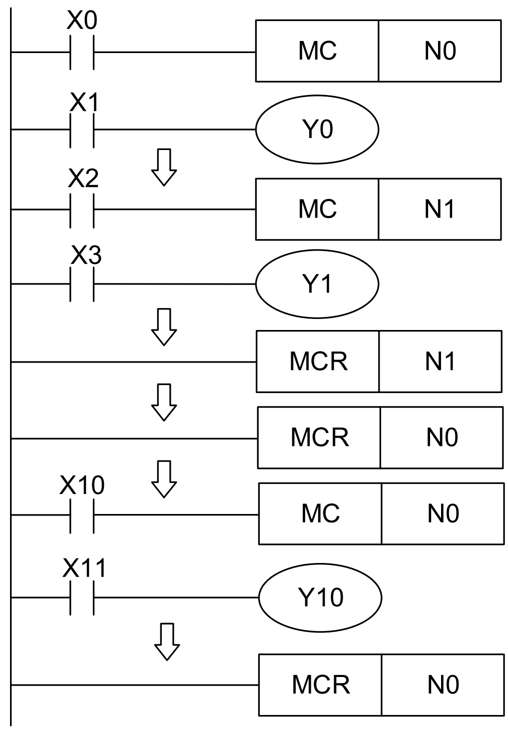
Operand
N0 ~ N7

Instruction description:

The MC instruction serves as the beginning of primary control. After it is executed, instructions placed between MC and MCR run as usual. When the MC instruction is set to OFF, the execution of instructions placed between MC and MCR is described in table below:

Types of instructions	Description
Common timer	Reset timing value, coil OFF, contacts remain inactive
Counter	Coil OFF, counting values and contacts remain at the current state.
Coils driven by OUT instruction	All turned OFF
Components driven by SET and RST instructions	Remain the current state
Application instructions	Action remains intact. The FOR-NEXT nest loop will execute for N times. Instructions in the FOR-NEXT loop will be executed in the same manner as the instructions between MC and MCR.

Example:



MC/MCR Ladder diagram

■ LDP: Start of rising-edge detection

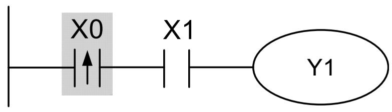
Instruction	Function	Model
LDP	Start of rising-edge detection	NC Series

Operand							
X0 ~ X39 X64 ~ X511	Y0 ~ Y39 Y64 ~ Y51	M0 ~ M3,071	A0 ~ A511	T0 ~ T255	C0 ~ C77	D0 ~ D1,535	V, Z
▪	▪	▪	▪	▪	▪	-	-

Instruction description:

The LDP instruction is used the same as the LD instruction but with a different function. It saves the current contents and the detected rising-edge state of the acquired contact in a cumulative register.

Example:



LDP(X0) Ladder diagram

■ LDF: Start of falling-edge detection

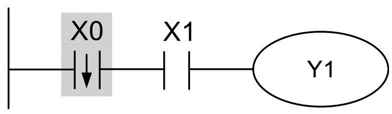
Instruction	Function	Model
LDF	Start of falling-edge detection	NC Series

Operand							
X0 ~ X39 X64 ~ X511	Y0 ~ Y39 Y64 ~ Y51	M0 ~ M3,071	A0 ~ A511	T0 ~ T255	C0 ~ C77	D0 ~ D1,535	V, Z
▪	▪	▪	▪	▪	▪	-	-

Instruction description:

The LDF instruction is used the same as the LD instruction but with a different function. It saves the current contents and the detected falling-edge state of the acquired contact in a cumulative register.

Example:



LDP(X0) Ladder diagram

- ANDP: Rising-edge detection serial connection

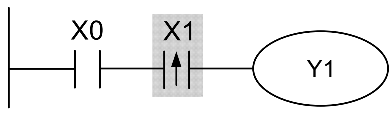
Instruction	Function	Model
ANDP	Rising-edge detection serial connection	NC Series

Operand							
X0 ~ X39 X64 ~ X511	Y0 ~ Y39 Y64 ~ Y51	M0 ~ M3,071	A0 ~ A511	T0 ~ T255	C0 ~ C77	D0 ~ D1,535	V, Z
▪	▪	▪	▪	▪	▪	-	-

Instruction description:

The ANDP instruction serial connects the contact's rising-edge detection.

Example:



ANDP(X1) Ladder diagram

- ANDF: Falling-edge detection serial connection

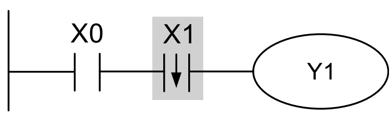
Instruction	Function	Model
ANDF	Falling-edge detection serial connection	NC Series

Operand							
X0 ~ X39 X64 ~ X511	Y0 ~ Y39 Y64 ~ Y51	M0 ~ M3,071	A0 ~ A511	T0 ~ T255	C0 ~ C77	D0 ~ D1,535	V, Z
▪	▪	▪	▪	▪	▪	-	-

Instruction description:

The ANDF instruction serial connects the contact's falling-edge detection.

Example:



ANDF(X1) Ladder diagram

■ ORP: Rising-edge detection parallel connection

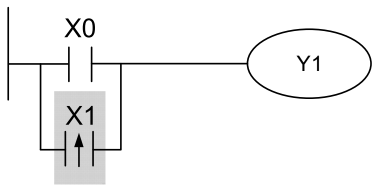
Instruction	Function	Model
ORP	Rising-edge detection parallel connection	NC Series

Operand							
X0 ~ X39 X64 ~ X511	Y0 ~ Y39 Y64 ~ Y51	M0 ~ M3,071	A0 ~ A511	T0 ~ T255	C0 ~ C77	D0 ~ D1,535	V, Z
▪	▪	▪	▪	▪	▪	-	-

Instruction description:

The ORP instruction parallel connects the contact's rising-edge detection.

Example:



ORP(X0, X1) Ladder diagram

■ ORF: Falling-edge detection parallel connection

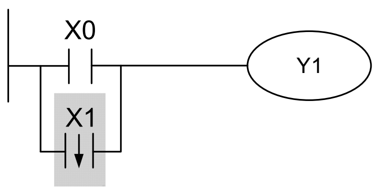
Instruction	Function	Model
ORF	Falling-edge detection parallel connection	NC Series

Operand							
X0 ~ X39 X64 ~ X511	Y0 ~ Y39 Y64 ~ Y51	M0 ~ M3,071	A0 ~ A511	T0 ~ T255	C0 ~ C77	D0 ~ D1,535	V, Z
▪	▪	▪	▪	▪	▪	-	-

Instruction description:

The ORF instruction parallel connects the contact's falling-edge detection.

Example:



ORP(X0, X1) Ladder diagram

- PLS: Upper differential output

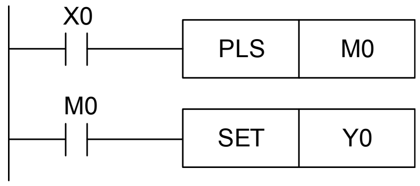
Instruction	Function	Model
PLS	Upper differential output	NC Series

Operand							
X0 ~ X39 X64 ~ X511	Y0 ~ Y39 Y64 ~ Y51	M0 ~ M3,071	A0 ~ A511	T0 ~ T255	C0 ~ C77	D0 ~ D1,535	V, Z
-	▪	▪	-	-	-	-	-

Instruction description:

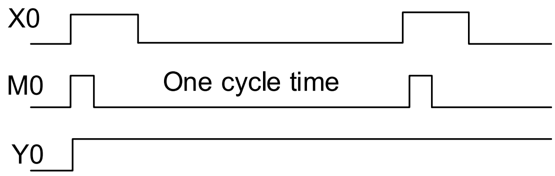
Upper differential output instruction. If X0 goes from OFF to ON (rising-edge triggered), the PLS instruction is executed, and M0 sends one pulse with a length of one cycle time.

Example:



PLS(M0) Ladder diagram

Timing diagram:



PLS(M0) Timing diagram

■ PLF: Lower differential output

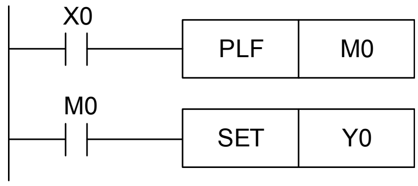
Instruction	Function	Model
PLF	Lower differential output	NC Series

Operand							
X0 ~ X39 X64 ~ X511	Y0 ~ Y39 Y64 ~ Y51	M0 ~ M3,071	A0 ~ A511	T0 ~ T255	C0 ~ C77	D0 ~ D1,535	V, Z
-	▪	▪	-	-	-	-	-

Instruction description:

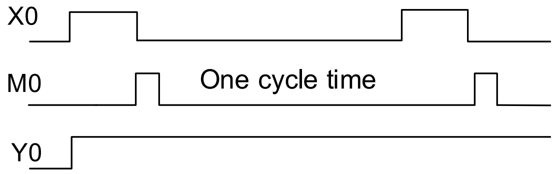
Lower differential output instruction. If X0 goes from ON to OFF (falling-edge triggered), the PLF instruction is executed, and M0 sends one pulse with a length of one cycle time.

Example:



PLF(M0) Ladder diagram

Timing diagram:



PLF(M0) Timing diagram

2

■ END: Program ends

Instruction	Function	Model
END	Program ends	NC Series

Operand							
X0 ~ X39 X64 ~ X511	Y0 ~ Y39 Y64 ~ Y51	M0 ~ M3,071	A0 ~ A511	T0 ~ T255	C0 ~ C77	D0 ~ D1,535	V, Z
-	-	-	-	-	-	-	-

Instruction description:

A ladder or instruction program must end with an END instruction. The MLC scans and runs from address 0 to END instruction and then return to address 0 to repeat.

■ NOP: No action

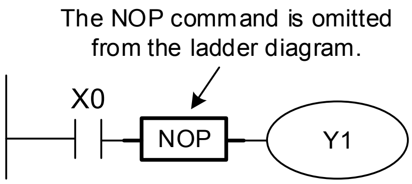
Instruction	Function	Model
NOP	No action	NC Series

Operand							
X0 ~ X39 X64 ~ X511	Y0 ~ Y39 Y64 ~ Y51	M0 ~ M3,071	A0 ~ A511	T0 ~ T255	C0 ~ C77	D0 ~ D1,535	V, Z
-	-	-	-	-	-	-	-

Instruction description:

The NOP instruction does not conduct any operation in the program. After its execution, the existing logical operation result will be kept. If users desire to delete a certain instruction in a program without altering the length of the program, then NOP instruction can be used to replace the instruction.

Example:



NOP Ladder diagram

■ INV: Invert the operation result

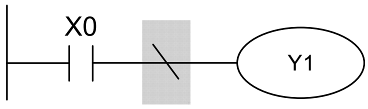
Instruction	Function	Model
INV	Invert the operation result	NC Series

Operand							
X0 ~ X39 X64 ~ X511	Y0 ~ Y39 Y64 ~ Y51	M0 ~ M3,071	A0 ~ A511	T0 ~ T255	C0 ~ C77	D0 ~ D1,535	V, Z
-	-	-	-	-	-	-	-

Instruction description:

The INV instruction inverts the logic operation result before the INV instruction and saves it into a cumulative register.

Example:



INV Ladder diagram

■ P Indicator

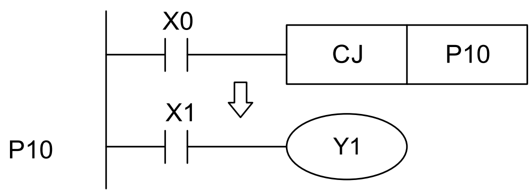
Instruction	Function	Model
P	Indicator	NC Series

Operand
P0 ~ P255

Instruction description:

Indicator P is used for jump instruction CJ and subroutine calling instruction CALL. Indicator P can be used in random sequence. However, its No. cannot be repeated or an unexpected error may result.

Example:



P Ladder diagram

2

■ I Interrupt indicator

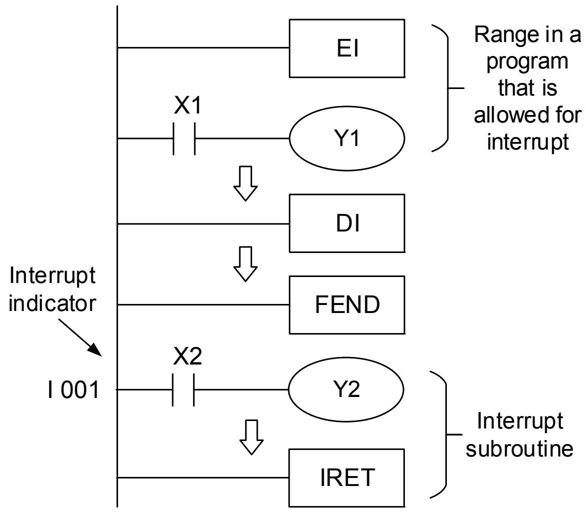
Instruction	Function	Model
I	Interrupt indicator	NC Series

Operand
IX00 ~ IX07, IC00 ~ IC01, IR00 ~ IR23

Instruction description:

To interrupt a service program, use an interrupt indicator (I□□□) to tag its starting point. When the interruption ends, use an IRET instruction to return to the main program. It must be applied together with application instruction IRET, EI and DI.

Example:



I Ladder diagram

Introduction to Application Instructions

3

This chapter introduces the types and basic uses of MLC application instructions.

3

3.1 List of application instructions

NC series MLC employs many different application instructions which are listed in the table below. Please refer to chapter 4 for more detailed descriptions about these instructions.

Type	API	Instruction Code		Number of operand	Function	STEPS	
		16-bit	32-bit			16-bit	32-bit
Loop control	00	CJ	-	1	Conditional jump	2	-
	01	CALL	-	1	Call subroutines	2	-
	02	SRET	-	-	End subroutines	1	-
	03	IRET	-	-	Return from interruption	1	-
	04	EI	-	-	Enable interruption	1	-
	05	DI	-	-	Disable interruption	1	-
	06	FEND	-	-	End main program	1	-
	07	FOR	-	1	Nest loops start	3	-
	08	NEXT	-	-	Nest loops end	1	-
Transmission and comparison	09	MOV	DMOV	2	Move data	4	6
	10	CML	DCML	2	Inverting transmission	4	5
	11	BCD	DBCD	2	BIN to BCD conversion	4	4
	12	BIN	DBIN	2	BCD to BIN conversion	4	4
Arithmetic and logic operation	13	ADD	DADD	3	BIN addition	6	8
	14	SUB	DSUB	3	BIN subtraction	6	8
	15	MUL	DMUL	3	BIN multiplication	6	8
	16	DIV	DDIV	3	BIN division	6	8
	17	INC	DINC	1	Plus one (BIN)	3	3
	18	DEC	DDEC	1	Minus one (BIN)	3	3
	19	WAND	DWAND	3	AND operation	6	8
	20	WOR	DWOR	3	OR operation	6	8
	21	WXOR	DWXOR	3	XOR / WXOR operation	6	8
	22	NEG	DNEG	1	Acquire negative value (2's complement)	3	3
Rotation	23	ROR	DROR	2	Rotate right	4	4
	24	ROL	DROL	2	Rotate left	4	4
Data processing	25	ZRST	-	2	Zone reset	4	-
	26	DECO	-	3	Decoder	6	-
	27	ENCO	-	3	Encoder	6	-
	28	BON	DBON	3	Monitor specified bit state	6	7
	29	ANS	-	3	Alarm trigger	5	-
	30	ANR	-	-	Alarm clear	1	-
High-speed processing	31	REF	-	2	I/O refresh	3	-
	32	-	DHSCS	3	Compare setup (high-speed counter)	-	5
	33	-	DHSCR	3	Compare reset (high-speed counter)	-	5
Convenience	34	ALT	-	1	ON/OFF alternate	3	-
Basic instructions	35	PLS	-	1	Upper differential output	3	-
	36	TMR	-	2	Timer	1	-
	37	CNT	DCNT	2	Counter	3	4
	38	PLF	-	1	Lower differential output	1	-

Type	API	Instruction Code		Number of operand	Function	STEPS	
		16-bit	32-bit			16-bit	32-bit
Contact type comparing instruction	39	LD=	DLD=	2	$S_1 = S_2$	4	6
	40	LD>	DLD>	2	$S_1 > S_2$	4	6
	41	LD<	DLD<	2	$S_1 < S_2$	4	6
	42	LD<>	DLD<>	2	$S_1 \neq S_2$	4	6
	43	LD<=	DLD<=	2	$S_1 \leqq S_2$	4	6
	44	LD>=	DLD>=	2	$S_1 \geqq S_2$	4	6
	45	AND=	DAND=	2	$S_1 = S_2$	4	6
	46	AND>	DAND>	2	$S_1 > S_2$	4	6
	47	AND<	DAND<	2	$S_1 < S_2$	4	6
	48	AND<>	DAND<>	2	$S_1 \neq S_2$	4	6
	49	AND<=	DAND<=	2	$S_1 \leqq S_2$	4	6
	50	AND>=	DAND>=	2	$S_1 \geqq S_2$	4	6
	51	OR=	DOR=	2	$S_1 = S_2$	4	6
	52	OR>	DOR>	2	$S_1 > S_2$	4	6
	53	OR<	DOR<	2	$S_1 < S_2$	4	6
	54	OR<>	DOR<>	2	$S_1 \neq S_2$	4	6
	55	OR<=	DOR<=	2	$S_1 \leqq S_2$	4	6
	56	OR>=	DOR>=	2	$S_1 \geqq S_2$	4	6
	57	VRT	DVRT	3	Variable table	70	134
Floating point operation	58	-	FADD	3	Binary floating point addition	-	7
	59	-	FSUB	3	Binary floating point subtraction	-	7
	60	-	FMUL	3	Binary floating point multiplication	-	7
	61	-	FDIV	3	Binary floating point division	-	7
	62	-	FCMP	3	Binary floating point comparison	-	7
	63	-	FINT	2	Binary floating point convert to BIN integer	-	5
	64	-	FDOT	2	BIN integer convert to binary floating point	-	5
	65	-	FRAD	2	Convert value in degree to radian	-	5
	66	-	FDEG	2	Convert value in radian to degree	-	5

Note 1: All application instructions listed above are valid for NC series.

3

3.2 Composition of application instructions

Application methods are vital for MLC control requirements. This section explains the format and terminologies related to MLC application instructions.

3.2.1 Format of application instructions

(1) (2) (3) (4) (5) (6)

API		MOV	S, D	Move data	NC Series
09	D				

(7)

	Bit device						Word device								
	X	Y	M	A	K		KnX	KnY	KnM	KnA	T	C	D	V	Z
S					*			*	*	*	*	*	*	*	*
D								*	*	*	*	*	*	*	*

16-bit instruction (4 STEP) MOV (Continuous running type)
32-bit instruction (6 STEP) D MOV (Continuous running type)
Flag: None

(8) (9)

Notes on the use of operands:
For S and D operands running on Z devices only 16-bit instructions are valid.

Notes:

(1) Application instruction API code.

(2) The upper box indicates it's a 16-bit instruction. A box with a hyphen indicates that there is no 16-bit version that exists for this instruction.

The lower box indicates it's a 32-bit instruction. A box with a hyphen indicates there is no 32-bit version that exists for this instruction. A 32-bit instruction is identified by a box filled with the letter D. (E.g. API 09 DMOV.)

(3) Application name.

(4) Application's operand format.

(5) Description of application function.

(6) The NC series MLC models that can use this instruction.

(7) Devices marked with the symbol * are the ones that can be used by the operand.

(8) A box shadowed in gray and filled with the symbol * indicates that the device can use register V and Z indirectly.

(9) Notes on the devices.

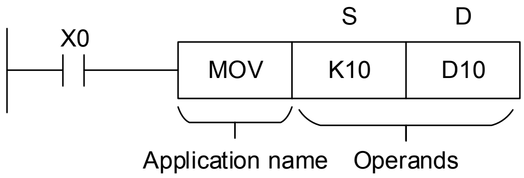
An application instruction is composed of two parts: an instruction name and its operands. The instruction name indicates its function, and operands indicate the devices that the instruction is applied to.

The instruction name takes one STEP in most cases while each one of its operands takes 2 (16-bit instruction) or 4 (32-bit instruction) STEP respectively.

3.2.2 Input of application instructions

Most application instructions contain more than one operand. Still there are a few that have no operand, e.g. EI and DI.

Each application instruction is represented by one code (API 00 ~ API 66) along with a unique name, e.g. the name of API 09 is MOV (Move data). To apply instruction API 09, just type its name "MOV" in the MLC Editor. Each application instruction has its own operand(s). Take MOV as the example.



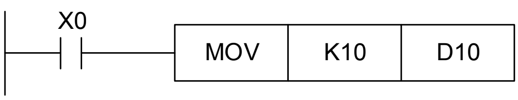
The above MOV instruction moves values in the operand assigned by S to the target operand assigned by D. In this instruction:
S, the source operand: Use S1, S2… to represent multiple operands respectively.
D, the target operand: Use D1, D2… to represent multiple operands respectively.
If the operand can only specify constant K/F or a register, it will be represented by m, m1, m2, n, n1 and n2.

3.2.3 Operand length (16-bit or 32-bit instruction)

Depending on the contents in the operand, the length of an operand can be 16-bit or 32-bit. A 16-bit instruction is for processing 16-bit operands, and 32-bit instruction is for processing 32-bit operands. The 32-bit instruction is indicated by adding a letter D in front of the 16-bit instruction.

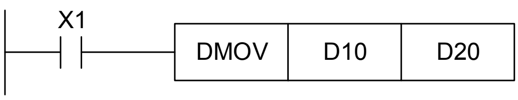
Example:

- 16-bit MOV instruction



When X0 = ON, operand K10 is sent to operand D10.

- 32-bit DMOV instruction

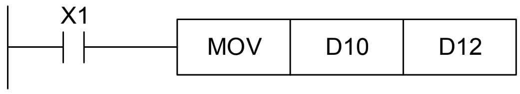


When X0 = ON, operand K10 is sent to operand D10.

3.2.4 Execution types of instructions

MLC instructions are executed continuously, which is called continuous running type.

Example:



When X1 = ON, the MOV instruction is executed once in every scan cycle. Thus, it is called continuous running type.

3.2.5 Designation of operands

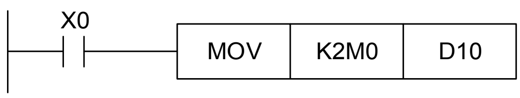
The features of the targets designated by operands are listed hereunder:

1. Bit devices X, Y, M and A can be combined into word devices. They store data for operations in the form of KnX, KnY, KnM and KnA in an application instruction.
2. Data register D, timer T, counter C and indirect index register V and Z are assigned by general operands.
3. A data register is usually in 16-bit, that is of the length of one D register. Users can assign two consecutive D registers to form a 32-bit register.
4. When an operand of 32-bit instruction assigns D0, the 32-bit data register composed of (D1, D0) will be occupied. D0 is the lower 16-bit and D1 is the upper 16-bit. Same rule applies to timer T, 16-bit counter and C0 ~ C63.
5. When 32-bit counter C64 ~ C77 is used as data register, they can only be assigned by the operand of 32-bit instruction.

3.2.6 Devices of the operand

The definitions of the devices assigned by operands are listed hereunder:

1. Devices X, Y, M and A can only set a single point to OFF/ON and are defined as bit device.
2. 16-bit or 32-bit devices T, C, D and registers V, Z are defined as word device.
3. To perform word device operations, users can define X, Y, M and A as word devices by prefixing them with Kn (n = 1 refers to 4 bits; for 16-bit instruction n = K1 ~ K4, for 32-bit instruction n = K1 ~ K8). For example, K2M0 refers to 8 bits, M0 ~ M7.



When X0 = ON, values in M0 ~ M7 will be moved to bit 0 ~ 7 of D10 and the remaining bits (8 ~ 15) will be set to 0.

3.2.7 Data processing of the word devices combined by bit devices

The data values of 16-bit instructions and 32-bit instructions are listed hereunder:

16-bit instructions	
Values assigned by 16-bit instruction are in the range of K-32,768 ~ K32,767	
Values contained in bit groups K1 ~ K4 are:	
K1 (4 bits)	0 ~ 15
K2 (8 bits)	0 ~ 255
K3 (12 bits)	0 ~ 4,095
K4 (16 bits)	-32,768 ~ +32,767

32-bit instructions	
Values assigned by 32-bit instruction are in range of K-2,147,483,648 ~ K2,147,483,647	
Values contained in bit groups K1 ~ K8 are:	
K1 (4 bits)	0 ~ 15
K2 (8 bits)	0 ~ 255
K3 (12 bits)	0 ~ 4,095
K4 (16 bits)	0 ~ 65,535
K5 (20 bits)	0 ~ 1,048,575
K6 (24 bits)	0 ~ 167,772,165
K7 (28 bits)	0 ~ 268,435,455
K1 (4 bits)	-2,147,483,648 ~ +2,147,483,647

3.3 Processing numeric values

This section describes how the devices with numeric values are processed by MLC application instructions.

Devices X, Y, M and A are called bit devices as they only has ON/OFF values. Devices T, C, D, V, and Z are called word devices as they can contain numeric values. Through special declaration, bit device can be used by an operand of application instruction in the form of numeric value. The declaration is to prefix the bit device with a place value in front of Kn. A 16-bit number can expressed by device K1 ~ K4 while a 32-bit number represented by K1 ~ K8.

Example: K2M0 is a 8-bit number presented by M0 ~ M7. Send K1M0, K2M0 and K3M0 to a 16-bit register and fill the upper bits with 0. Send K1M0, K2M0, K3M0, K4M0, K5M0, K6M0, and K7M0 to a 32-bit register and the upper bits will be filled with 0. In 16-bit or 32-bit operation, when the contents of the instruction are assigned with K1 ~ K3 (or K4 ~ K7) bit devices, all the vacant upper bits will be filled with 0. Thus, the operation is usually regarded as positive number operation.

- Assigning continuous numbers: Take data register D as the example. Its consecutive numbers are D0, D1, D2, D3, D4... For bit devices with assigned number, the continuous numbers are shown in table below:

Assigning continuous numbers			
K1X0	K1X4	K1X8	K1X12……
K2Y0	K2Y8	K2Y16	K2Y24……
K3M0	K3M12	K3M24	K3M36..….
K4A0	K4A16	K4A32	K4A48…….

As shown in the table above, the continuous numbers of X devices for K1 are multiples of 4. X devices for K2 are multiples of 8. Do not skip or jump numbers to avoid confusion (e.g. K1X0 and K1X5 are not assigned with a multiple of 4).

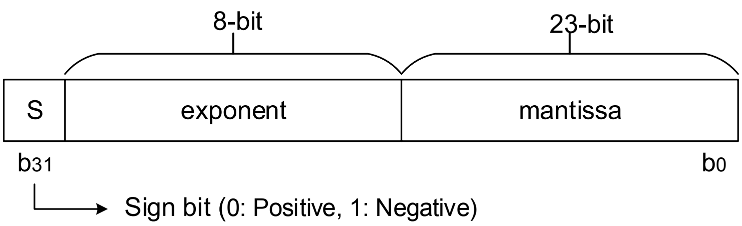
Note: When using K4Y0 in 32-bit operation, the upper 16 bits are regarded as 0. For 32 bits data, please use K8Y0 instead.

- The MLC internal numeric operation is conducted in BIN integers. The operation result can be decimal point (floating point) if decimal operation instructions are used.

Application methods about decimal point (floating point)		
API 58(FADD)	API 61(FDIV)	API 64(FDOT)
API 59(FSUB)	API 62(FCMP)	API 65(FRAD)
API 60(FMUL)	API 63(FINT)	API 66(FDEG)

- Expression of binary floating point:

  The NC series MLC expresses floating point in 32-bit according to the IEEE754 standards. See below for its format:



Valid range of values: $(-1)^S \times 2^{E-B} \times 1.M$ where B = 127

Range of values that can be expressed by 32-bit floating point is ±2-126 ~ ±2+128 or ±1.1755×10-38 ~ ±3.4028×10+38.

Example 1: Express 23 as a 32-bit floating point.

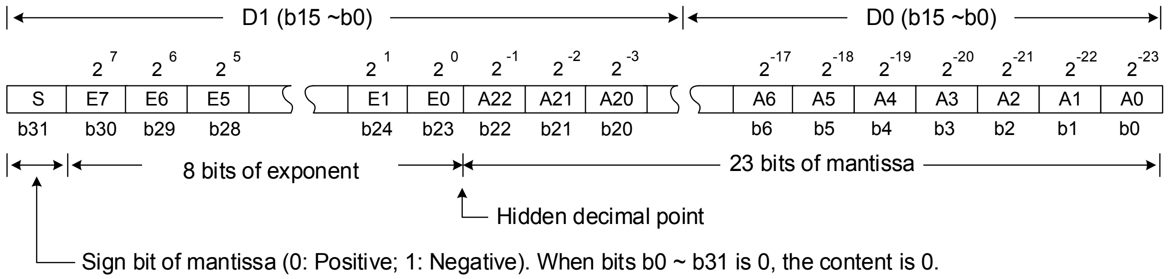
Steps:

1. Convert 23 to binary: 23.0=10111
2. Normalize the binary: 10111=1.0111 × 2^4, 0111 is the mantissa and 4 is the exponent.
3. Get the storage value of the exponent ∵E-B=4 →E-127=4 ∴E=131=100000112
4. Combine sign bit, exponent and mantissa into a floating point.

0 100**0001**1 011**1000**0000**0000**0000**0000**$_2$ = $41B80000_{16}$

Example 2: Express -23.0 as a 32-bit floating point

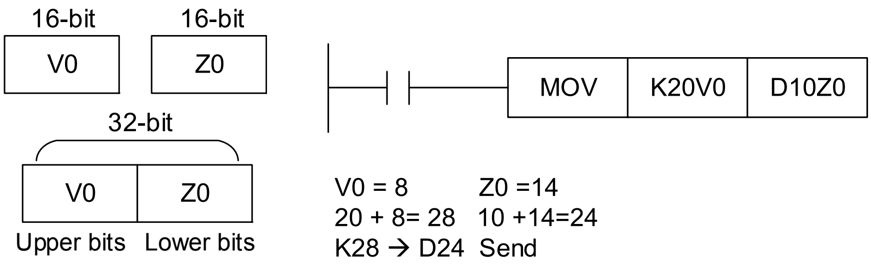
The floating point format of -23.0 can be converted exactly the same as +23.0 except that the sign bit is 1. MLC applies two consecutive registers to form a 32-bit floating point. Here register (D1, D0) is used to keep a binary floating point as described below:



3

3.4 Modifying operands via register V, Z

Indirect index register V and Z can be used to modify operands, which are all 16-bit registers. (Indirect index register is 16-bit registers and can be used to modify operands.) There are total 16 points of Z, V in NC series. Detailed description is given below:



As shown in figure, contents in operands vary with the contents in V and Z. That is, the operands are modified by V and Z and thus indirectly specified. For example, V0 = 8 and K20V0 represents constant K28 (20 + 8). With valid condition, constant K28 will be sent to register D24.

The V and Z registers are all 16-bit registers and can be read and written. When 32-bit length is required, users must assign register V for it. In such case, register V will overwrite Z, and Z cannot be used anymore. Otherwise, errors will occur in register V. The (V, Z) combination used for 32-bit indirect specified register is (V0, Z0), (V1, Z1), (V2, Z2) …and (V7, Z7).

Devices in NC series that can be modified are device P, KnX, KnY, KnM, KnA, T, C and D.
User can assign V or Z to modify 16-bit register, and only V can be assigned to modify 32-bit register.

3.5 Index of the application instructions

Type	API	Instruction code		Function
		16-bit	32-bit	
A	13	ADD	DADD	BIN addition
	29	ANS	-	Alarm trigger
	30	ANR	-	Alarm clear
	34	ALT	-	ON/OFF alternate
	45	AND=	DAND=	$S_1 = S_2$
	46	AND>	DAND>	$S_1 > S_2$
	47	AND<	DAND<	$S_1 < S_2$
	48	AND<>	DAND<>	$S_1 \neq S_2$
	49	AND<=	DAND<=	$S_1 \leqq S_2$
	50	AND>=	DAND>=	$S_1 \geqq S_2$
B	11	BCD	DBCD	BIN to BCD conversion
	12	BIN	DBIN	BCD to BIN conversion
	28	BON	DBON	Monitor specified bit state
C	00	CJ	-	Conditional jump
	01	CALL	-	Call subroutines
	10	CML	DCML	Inverting transmission
D	05	DI	-	Disable interruption
	16	DIV	DDIV	BIN division
	18	DEC	DDEC	Minus one (BIN)
	26	DECO	-	Decoder
E	04	EI	-	Enable interruption
	27	ENCO	-	Encoder
F	06	FEND	-	End main program
	07	FOR	-	Nest loops start
	58	-	FADD	Binary floating point addition
	59	-	FSUB	Binary floating point subtraction
	60	-	FMUL	Binary floating point multiplication
	61	-	FDIV	Binary floating point division
	62	-	FCMP	Binary floating point comparison
	63	-	FINT	Binary floating point convert to integer (truncated)
	64	-	FDOT	Integer convert to binary decimal
	65	-	FRAD	Convert value in degree to radian
	66	-	FDEG	Convert value in radian to degree
H	32	-	DHSCS	Compare setup (high-speed counter)
	33	-	DHSCR	Compare reset (high-speed counter)

Type	API	Instruction code		Function
		16-bit	32-bit	
I	03	IRET	-	Return from interruption
	17	INC	DINC	Plus one (BIN)
L	39	LD=	DLD=	$S_1 = S_2$
	40	LD>	DLD>	$S_1 > S_2$
	41	LD<	DLD<	$S_1 < S_2$
	42	LD<>	DLD<>	$S_1 \neq S_2$
	43	LD<=	DLD<=	$S_1 \leqq S_2$
	44	LD>=	DLD>=	$S_1 \geqq S_2$
M	09	MOV	DMOV	Move data
	15	MUL	DMUL	BIN multiplication
N	08	NEXT	-	Nest loops end
	22	NEG	DNEG	Get negative value (Two's complement)
O	51	OR=	DOR=	$S_1 = S_2$
	52	OR>	DOR>	$S_1 > S_2$
	53	OR<	DOR<	$S_1 < S_2$
	54	OR<>	DOR<>	$S_1 \neq S_2$
	55	OR<=	DOR<=	$S_1 \leqq S_2$
	56	OR>=	DOR>=	$S_1 \geqq S_2$
P	35	PLS	-	Upper differential output
	38	PLF	-	Lower differential output
R	23	ROR	DROR	Rotate right
	24	ROL	DROL	Rotate left
	31	REF	-	I/O refresh
S	02	SRET	-	End subroutines
	14	SUB	DSUB	BIN subtraction
T	36	TMR	-	Timer
V	57	VRT	DVRT	Variable table
W	19	WAND	DAND	AND operation
	20	WOR	DOR	OR operation
	21	WXOR	DXOR	XOR operation
Z	25	ZRST	-	Zone reset

Description of Application Instructions

4

This chapter provides detailed description about MLC application instructions.

4.1 Loop control instructions

■ API-00 CJ: Conditional jump

API		CJ			S			Conditional jump		NC Series					
00	-														
	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z

16-bit instruction: CJ continuous running type (2 STEP).
32-bit instruction: None.
Flag: None

Notes on the use of operands:
Operand S can assign indicator P.
The No. of P can be modified by register V and Z.
Operand S of the NC series model can assign P0 ~ P255.

Instruction description:

S: The destination indicator of a conditional jump instruction.

CJ instruction can be used to skip a section of MLC program to shorten scan time and execute dual outputs. CJ instruction can repeatedly points to the same indicator P. However, do not point CJ and CALL instructions to the same indicator P or program errors will occur.

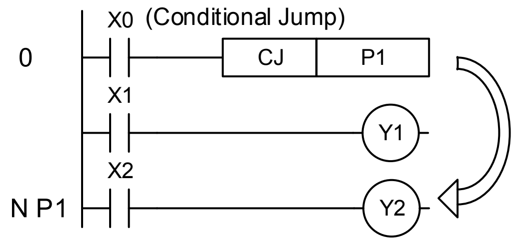
Device actions when executing jump instruction:

1. All instructions given by the user will still be executed when running jump instruction.
2. State of device Y, M, and A remains their previous state before the execution of jump instructions. (The jump instruction will not change the state of device Y, M, and A. Thus, their state remain the same after jump instruction is executed)
3. The currently running 10 ms and 100 ms timers continue timing.
4. The currently running high speed counters C78 and C79 keep on counting and the output contact functions as usual.
5. General application instructions are not executed.
6. Application instructions API 53 DHSCS and API 54 DHSCR that are currently running continue being executed.

Example 1:

When X0 = ON, the program jumps from address 0 to N (the assigned indicator P1) and keeps its execution. The addresses between 0 and N will not be executed.

When X0 = OFF, the program executes from address 0 downward in sequence as an ordinary program. CJ instruction will not be executed at this time.



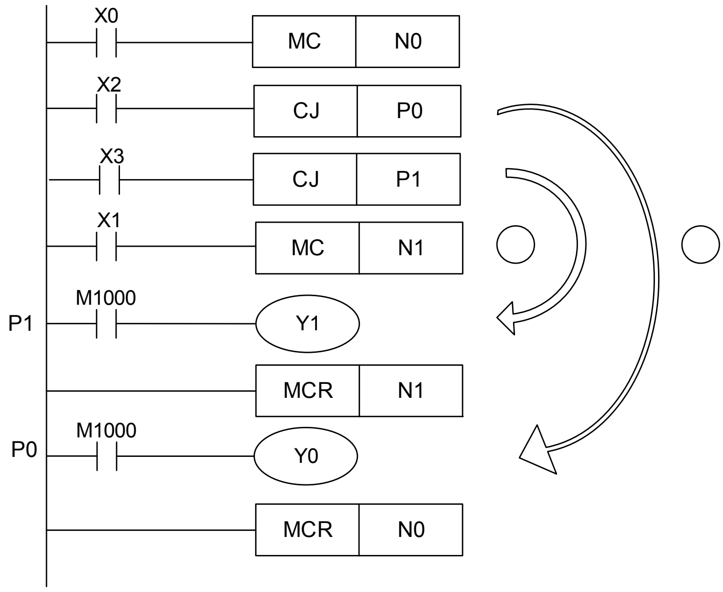
Example 2:

CJ instruction can be used between MC and MCR instructions for following 5 conditions:

1. Out of MC ~ MCR.
2. From outside of MC to within MC. Valid in loop P1 shown in the figure below.
3. Inside MC ~ MCR of the same level N.
4. From within MC to outside of MCR.
5. Jumping from one MC ~ MCR to another MC ~ MCR.

Actions in NC series MLC and higher versions:

When used between MC and MCR instructions, CJ instruction can only be applied to the loops outside of MC ~ MR or within MC ~ MCR in the same N layer. Jumping from one MC ~ MCR to another MC ~ MCR will lead to program errors. That is, only Item 1 and 3 described above can ensure correct actions in MLC, whereas others will cause errors.



Example 3:

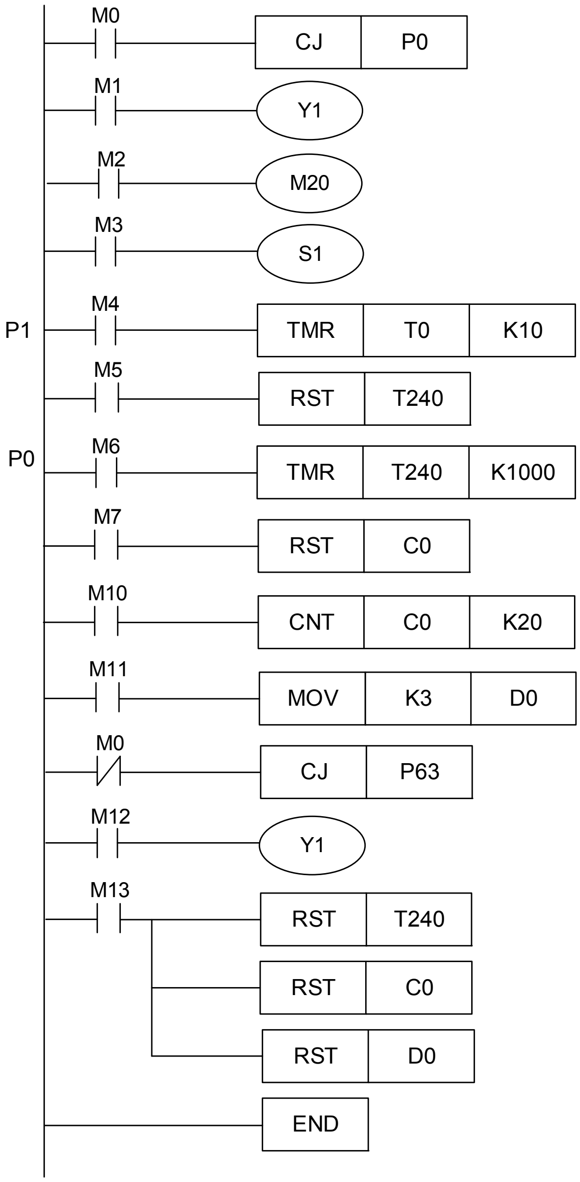
The state changes of each device:

Device	Contact state before CJ instruction is executed	Contact state during CJ instruction is executed	Output coil state during CJ instruction is executed
Y, M, A	M1, M2, M3 OFF	M1, M2, M3 OFF→ON	Y1[*1], M20, S1 OFF
	M1, M2, M3 ON	M1, M2, M3 ON→OFF	Y1[*1], M20, S1 ON
10, 100 ms Timer	M4 OFF	M4: OFF→ON	Timer T0 is not enabled.
	M4 ON	M4: ON→OFF	Timer T0 keeps on timing, M0: ON →OFF, timing to T0 → ON
C0 ~ C77	M7, M10 OFF	M10 ON/OFF trigger	Counter C0 is not enabled.
	M7 OFF, M10 ON/OFF trigger	M10 ON/OFF trigger	Counter C0 stops counting and stays latched. C0 resumes its counting after M0 goes OFF.
C78, C79	When the activated high speed counter, C78 or C79, encounters a CJ instruction, it will continue counting and the output contact point remains functioning.		

Device	Contact state before CJ instruction is executed	Contact state during CJ instruction is executed	Output coil state during CJ instruction is executed
Application instruction	M11 OFF	M11: OFF→ON	Application instructions are not executed.
	M11 ON	M11: ON→OFF	The skipped application instructions are not executed, but API 53 DHSCS and API 54 DHSCR remain being executed.

Note:

1. Y1 is a dual output, which is controlled by M1 when M0 = OFF and by M12 when M0 = ON.



4

■ API-01 CALL: Call subroutine

API		CALL			S			Call subroutine		NC Series					
01	-														
	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z

16-bit instruction: CALL continuous running type (2 STEP).
32-bit instruction: None.
Flag: None

Notes on the use of operands:
Operand S can assign indicator P.
The No. of P can be modified by register V and Z.
Operand S of the NC series model can assign P0 ~ P255.

Instruction description:

S: The indicator of calling subroutine.

The subroutine specified by the indicator should be placed after FEND instruction. The No. of indicator P, when used by CALL instruction, cannot be the same as the No. assigned by CJ instruction. If only CALL instruction is in use, it can call a subroutine of the same indicator No. repetetively without time limitation. Subroutine can be nested for maximum five calling layers including the initial CALL instruction. (Subroutine called in the sixth layer will not be executed.)

■ API-02 SRET: End subroutine

API		SRET	-	End subroutine	NC Series									
02	-													
	Bit device			Word device										
X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z

16-bit instruction: SRET continuous running type (1 STEP).
32-bit instruction: None.
Flag: None

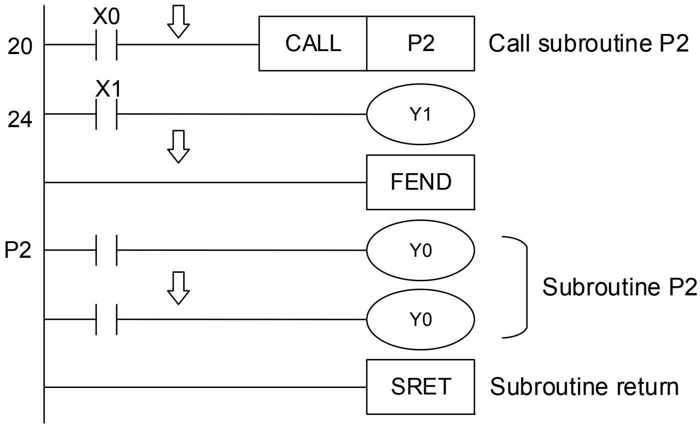
Notes on the use of operands:
No operand.
No contact to drive the instruction is required.

Instruction description:

SRET instruction refers to the termination of subroutine. The program returns to main program from SRET and then executes the sequential instruction next to CALL instruction.

Example 1:

When X0 = ON, CALL instruction is executed and the program jumps to the subroutine specified by indicator P2. Once SRET instruction is executed, the program returns to address 24 and continue its execution.

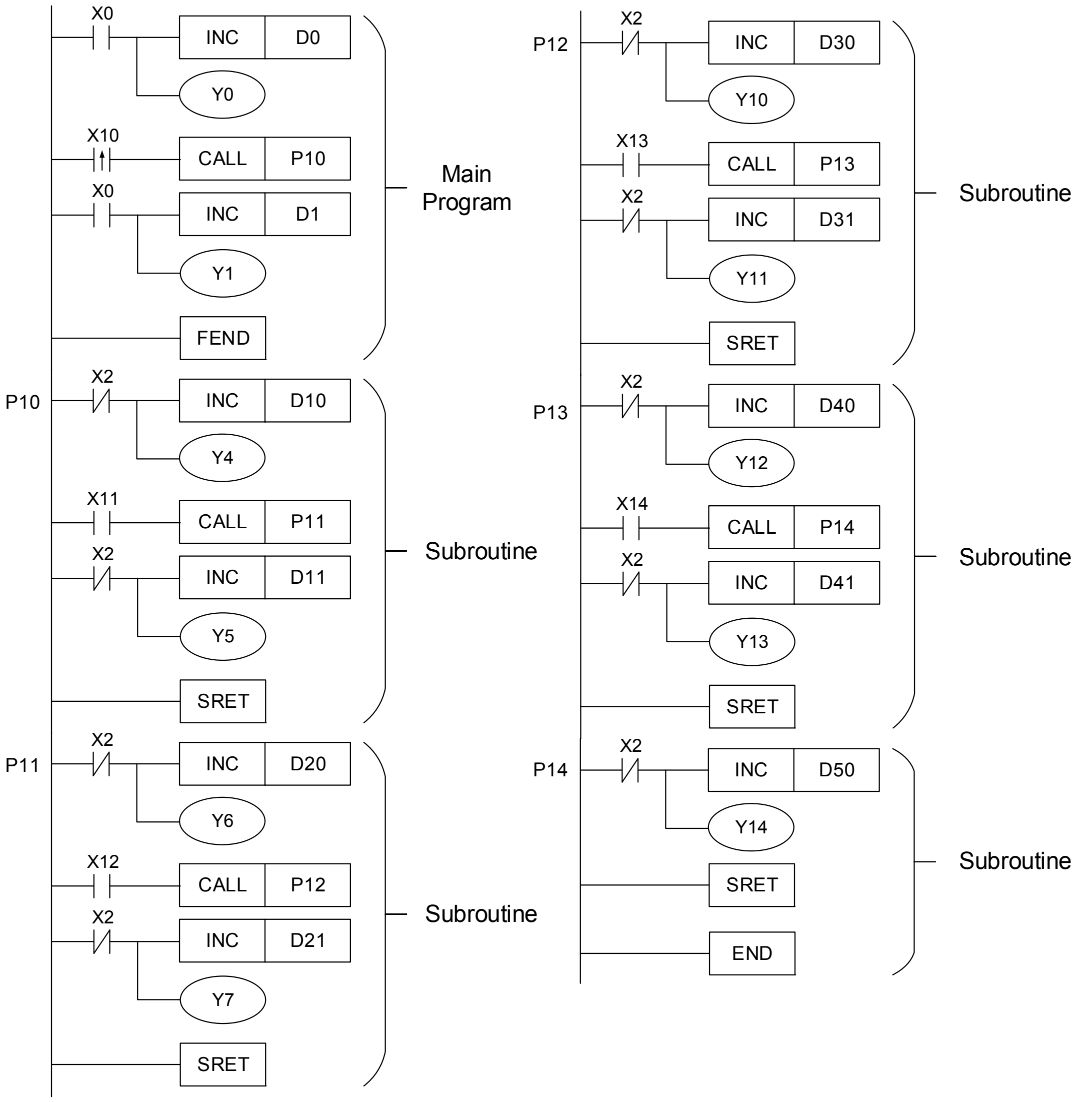


Example 2:

1. When X10 goes ON from OFF, CALL P10 instruction is rising edge-triggered and the program jumps to the subroutine specified by P10.
2. When X11 = ON, CALL P11 is executed and the program jumps to the subroutine specified by P11.
3. When X12 = ON, CALL P12 is executed and the program jumps to the subroutine specified by P12.
4. When X13 = ON, CALL P13 is executed and the program jumps to the subroutine specified by P13.
5. When X14 = ON, CALL P14 is executed and the program jumps to the subroutine specified by P14. Once SRET instruction is executed, the program will return to previous subroutine P※ and continue its execution.

4

6. When SRET is executed in subroutine P10, it will return to the main program.



API-03 IRET: Return from interruption

API		IRET	-	Return from interruption	NC Series									
03	-													
	Bit device				Word device									
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z

16-bit instruction: IRET continuous running type (1 STEP).
32-bit instruction: None.
Flag: None

Notes on the use of operands:
No operand.
No contact to drive the instruction is required.

Instruction description:

Interruption return refers to interrupt the subroutine. After the interruption is completed, the program returns to the main program from IRET instruction and continues executing the next instruction where the main program was interrupted.

API-04 EI: Enable interruption

API		EI	-	Enable interruption	NC Series									
04	-													
	Bit device				Word device									
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z

16-bit instruction: EI continuous running type (1 STEP).
32-bit instruction: None.
Flag: M2864 ~ M2873, M2880 ~ M2891, M2896 ~ M2907. See API-05 DI supplementary notes.

Notes on the use of operands:
No operand.
No contact to drive the instruction is required.
Pulse width of interruption signal must be greater than 200 us.
For ranges of I coding of each model, see API-05 DI supplementary notes.

API-05 DI: Disable interruption

API		DI	-	Disable interruption	NC Series									
05	-													
	Bit device				Word device									
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z

16-bit instruction: DI continuous running type (1 STEP).
32-bit instruction: None
Flag: None

Notes on the use of operands:
No operand.
No contact to drive the instruction is required.

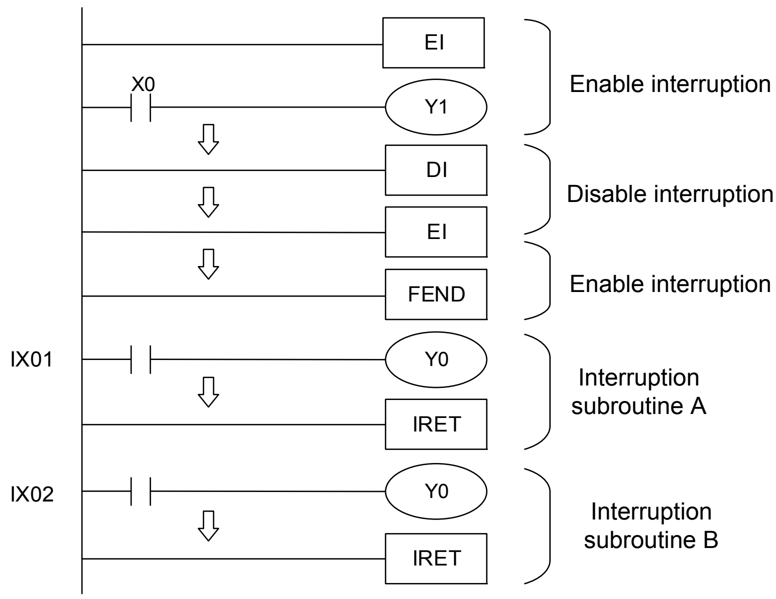
Instruction description:

1. EI instruction enables interrupting subroutines, including external and high speed counter interruptions.
2. An interruption subroutine is allowed between EI and DI instruction. DI instruction is not required if there is no interruption disabling section in the program.

3. For NC series models, if the interruption disabling special relays M2864 ~ M2873, M2880 ~ M2891, and M2896 ~ M2907 are not enabled, the corresponding interruption requests will not be executed even in the section allowed for interruptions.
4. The indicator (I) used by interruptions must be placed after the FEND instruction.
5. While an interruption subroutine is running, other interruptions are not allowed.
6. When there is more than one interruption, the priority will go to the currently running one. When several interruptions occur simultaneously, the priority will be given to the interruption with a smaller indicator No.
7. An interruption request between DI and EI instructions cannot be executed immediately. This instruction is kept in memory and will be executed when it is in a section allowed for interruption.
8. When an interruption indicator is used, do not repeatedly use the high speed counter driven by the same input contact X.
9. If immediate I/O action is required during the interruption, place a REF instruction in the program to update I/O state.

Example 1:

During the operation of MLC, when the program scans the section from instruction EI to DI and X1 = ON or X2 = ON, interruption subroutine A or B will be executed. After the IRET instruction is executed, the program returns to the main program and resumes its execution.



Supplementary notes:

No. of indicator I in NC series models:

1. OnBoard interruptions: 8 points, (IX0□, X0), (IX0□, X1), (IX0□, X2), (IX0□, X3), (IX0□, X4), (IX0□, X5), (IX0□, X6) and (IX0□, X7).
2. High speed counter interruption: 2 points, IC00 and IC01. (Work with API 32 DHSCS instruction to generate interruption signal.)

3. There are total 32 Remote I/O interruptions, IR00 ~ IR31 that are corresponding to the Input X of Remote X256 ~ 287 respectively. Each IR interruption corresponds to one of the 32 Input X of Remote I/O Card 0.
4. In MLC system, the interruptions are executed in FIFO (first in, first out) order.

The flags that disable the insertion of interruption indicators in NC series models:

Flag	Function description
M2880	IX00 interrupt input (On Board X0). (1: Enable; 0: Disable)
M2881	IX01 interrupt input (On Board X1).
M2882	IX02 interrupt input (On Board X2).
M2883	IX03 interrupt input (On Board X3).
M2884	IX04 interrupt input (On Board X4).
M2885	IX05 interrupt input (On Board X5).
M2886	IX06 interrupt input (On Board X6).
M2887	IX07 interrupt input (On Board X7).
M2888	IC00 interrupt input (Hardware Counter 0).
M2889	IC01 interrupt input (Hardware Counter 1).
M2896	IR00 interrupt input (X256 of Remote IO module)
M2897	IR01 interrupt input (X257 of Remote IO module)
M2898	IR02 interrupt input (X258 of Remote IO module).
M2899	IR03 interrupt input (X259 of Remote IO module)
M2900	IR04 interrupt input (X260 of Remote IO module)
M2901	IR05 interrupt input (X261 of Remote IO module)
M2902	IR06 interrupt input (X262 of Remote IO module)
M2903	IR07 interrupt input (X263 of Remote IO module)
M2904	IR08 interrupt input (X264 of Remote IO module)
M2905	IR09 interrupt input (X265 of Remote IO module)
M2906	IR10 interrupt input (X266 of Remote IO module)
M2907	IR11 interrupt input (X267 of Remote IO module)
M2908	IR12 interrupt input (X268 of Remote IO module)
M2909	IR13 interrupt input (X269 of Remote IO module)
M2910	IR14 interrupt input (X270 of Remote IO module)
M2911	IR15 interrupt input (X271 of Remote IO module)
M2912	IR16 interrupt input (X272 of Remote IO module)
M2913	IR17 interrupt input (X273 of Remote IO module)
M2914	IR18 interrupt input (X274 of Remote IO module)
M2915	IR19 interrupt input (X275 of Remote IO module)
M2916	IR20 interrupt input (X276 of Remote IO module)
M2917	IR21 interrupt input (X277of Remote IO module)
M2918	IR22 interrupt input (X278 of Remote IO module)
M2919	IR23 interrupt input (X279 of Remote IO module)
M2920	IR24 interrupt input (X280 of Remote IO module)
M2921	IR25 interrupt input (X281 of Remote IO module)
M2922	IR26 interrupt input (X282 of Remote IO module)
M2923	IR27 interrupt input (X283 of Remote IO module)
M2924	IR28 interrupt input (X284 of Remote IO module)
M2925	IR29 interrupt input (X285 of Remote IO module)
M2926	IR30 interrupt input (X286 of Remote IO module)
M2927	IR31 interrupt input (X287 of Remote IO module)

4

API-06 FEND: End main program

API		FEND		-			End main program		NC Series					
06	-													
	Bit device				Word device									
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z

16-bit instruction: FEND continuous running type (1 STEP).
32-bit instruction: None.
Flag: None

Notes on the use of operands:
No operand.
No contact to drive the instruction is required.

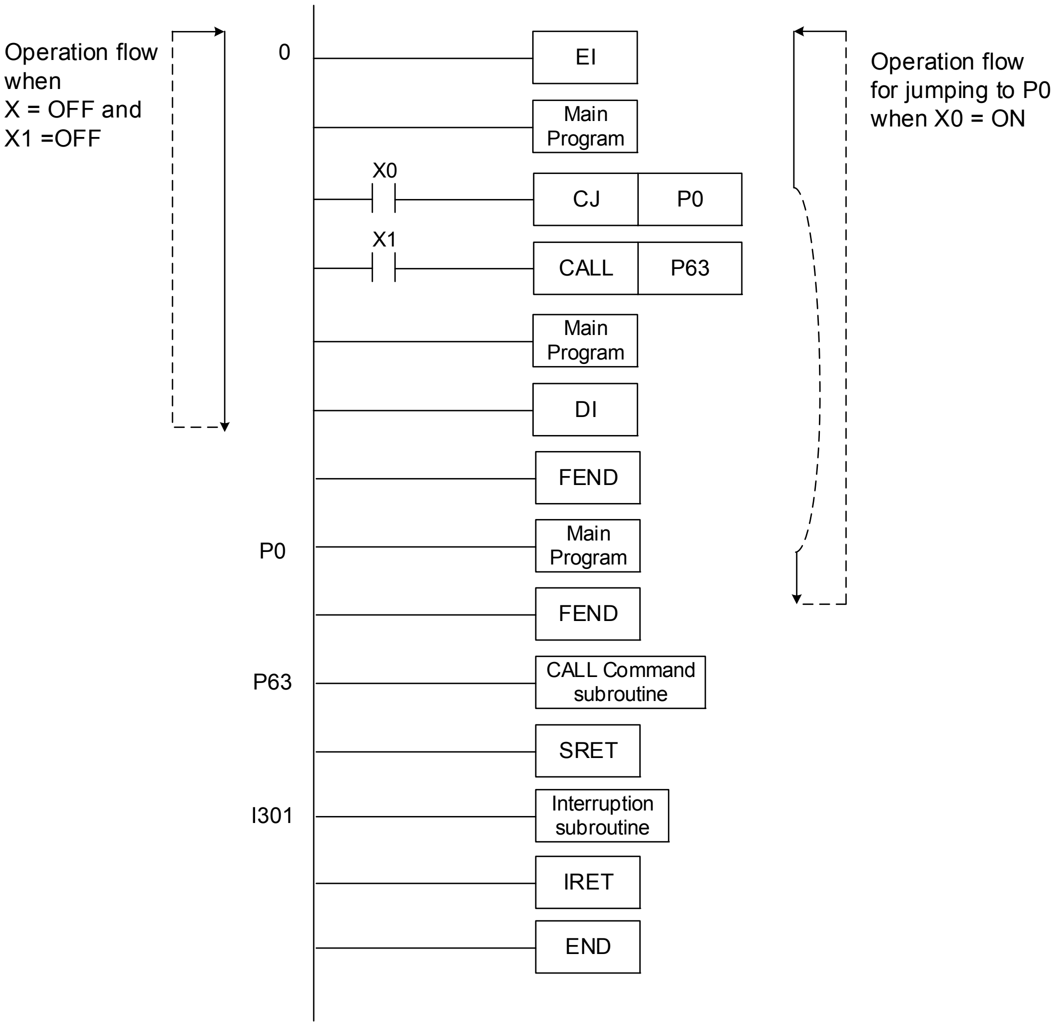
Instruction description:

FEND instruction indicates the termination of the main program. After being executed by MLC, FEND functions the same as END instruction. The subroutine of CALL instruction must be placed after FEND instruction and ended with SRET instruction. The interruption program must be written after FEND instruction and ended with IRET instruction. When more than one FEND instructions are in use, place the subroutine and interruption service program between the loop from the final FEND to END instruction.

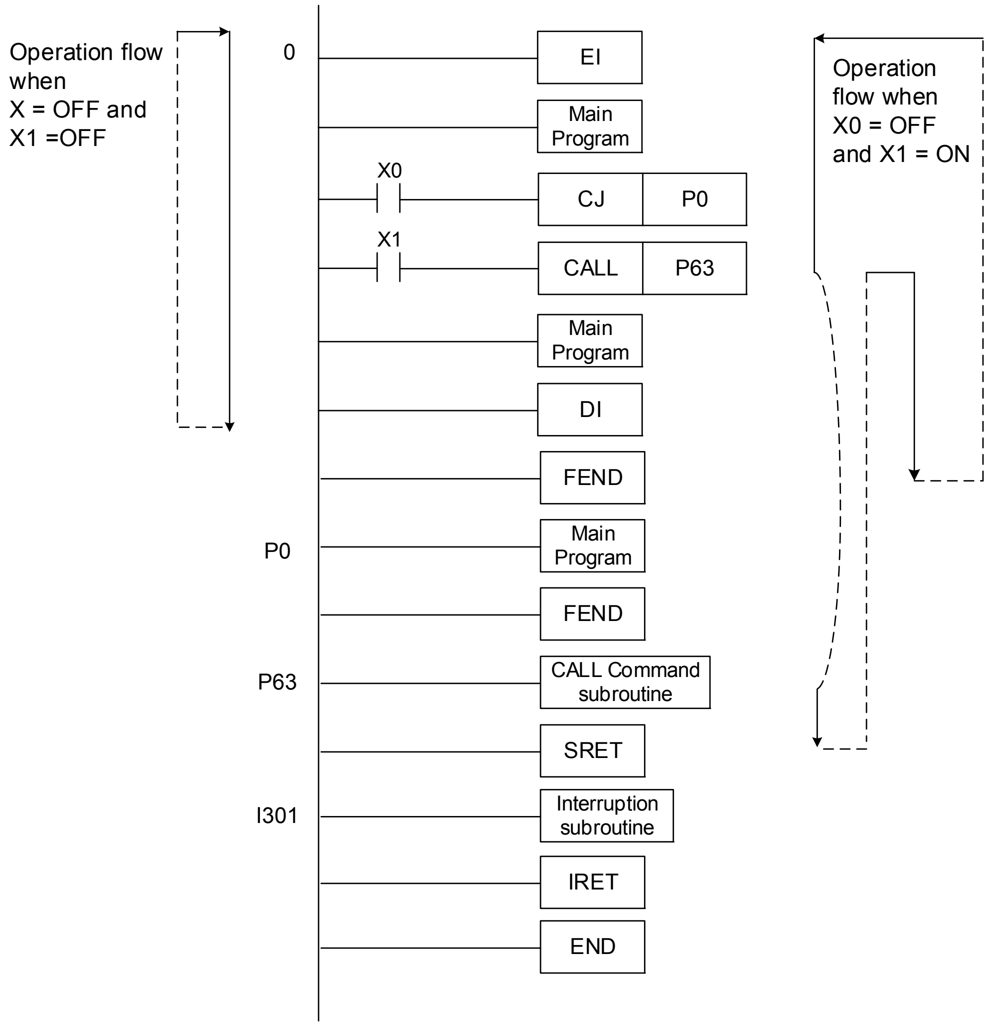
Program error will occur in the following cases:

1. After the execution of CALL instruction, FEND instruction is executed before the corresponding SRET instruction.
2. After the execution of FOR instruction, FEND instruction is executed before the corresponding NEXT instruction.

Operation flow of CJ instruction:



Operation flow of the CALL instruction:



■ API-07 FOR: Nest loops start

API		FOR				S			Nest loops start		NC Series			
07	-													
	Bit device				Word device									
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z

16-bit instruction: FOR continuous running type (3 STEP).
32-bit instruction: None.
Flag: None

Notes on the use of operands:
No contact to drive the instruction is required.
See specification of each model for the valid range of each device.

Instruction description:

S indicates the number of times the loop is to be executed.

■ API-08 NEXT: Nest loops end

API		NEXT				-			Nest loops end		NC Series			
08	-													
	Bit device				Word device									
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z

16-bit instruction: NEXT continuous running type (1 STEP).
32-bit instruction: None.
Flag: None

Notes on the use of operands:
No operand.
No contact to drive the instruction is required.

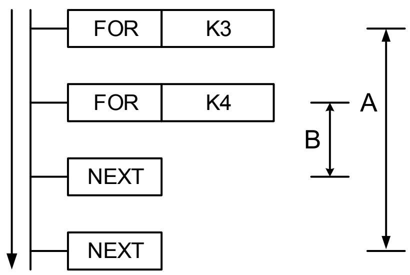
Instruction description:

1. FOR instruction indicates FOR ~ NEXT loop executing for N times, and then the program exits the loop for the next sequential instruction.
2. The range of the repeating times: N = K1 ~ K32, 767. N is regarded as K1 when N ≤ 1.
3. To skip the FOR ~ NEXT loop, use CJ instruction to jump to the given program.
4. Program error will occur in the following cases:
   a. NEXT instruction is executed before FOR instruction.
   b. The FOR instruction lacks a corresponding NEXT instruction.
   c. There is a NEXT instruction issued after FEND or END instruction.
   d. The number of FOR instructions is different from the number of NEXT instructions.
5. The FOR ~ NEXT loops can nest for up to 5 layers. Be aware that the more layers there are, the more time is required for MLC scanning.

4

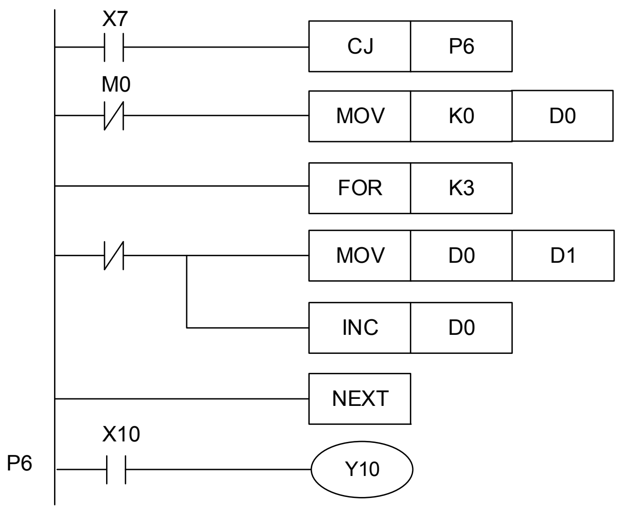
Example 1:

Program A repeats three times, and then continues executing the program after the final NEXT instruction. During each execution of program A, program B is executed for 4 times. Therefore, program B is executed 3 × 4 = 12 times in total.



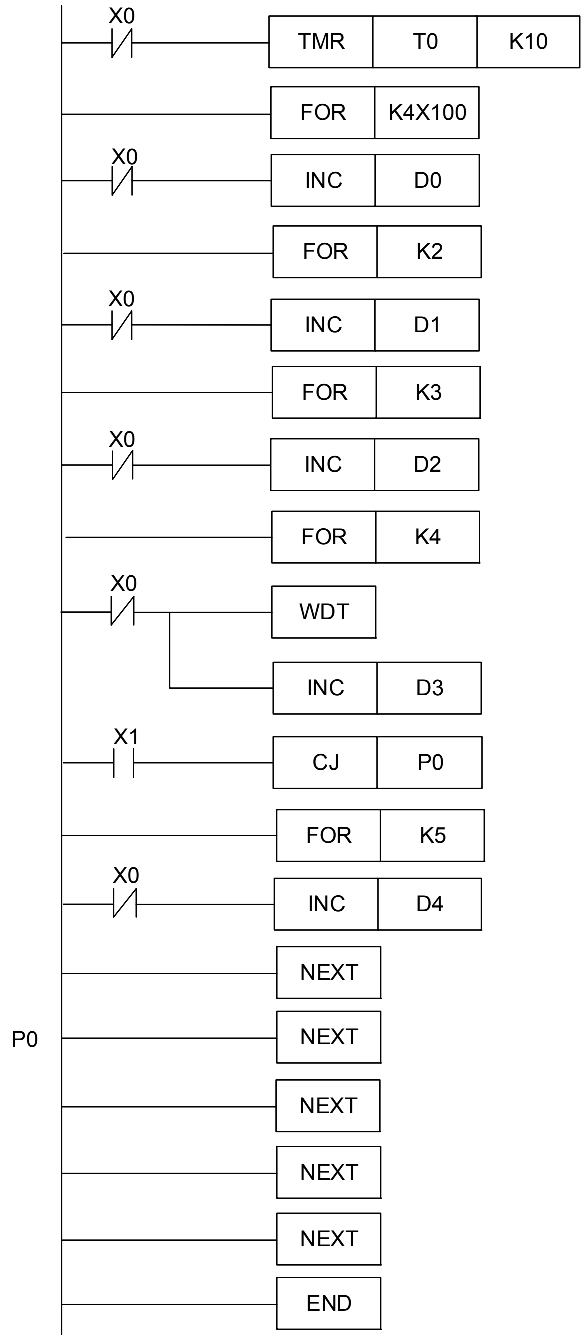
Example 2:

When X7 = OFF, MLC executes the program between FOR and NEXT instructions. When X7 = ON, execute CJ instruction and the program jumps to P6, ignoring the program between FOR and NEXT.



Example 3:

Users can use a CJ instruction to skip a FOR ~ NEXT loop. When X1 = ON, CJ instruction can be used to skip the most inner layer of FOR ~ NEXT loop and the program jumps to P0.



4.2 Transmission and comparison instructions

■ API-09 MOV: Move data

API		MOV				S, D			Move data		NC Series				
09	D														
	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S					*		*	*	*	*	*	*	*	*	*
D								*	*	*	*	*	*	*	*

16-bit instruction: MOV continuous running type (4 STEP).
32-bit instruction: DMOV continuous running type (6 STEP).
Flag: None

Notes on the use of operands:
If operands S and D are used in register Z, only 16-bit instruction is applicable.
Please see chapter 1 for details about the applicable range of each device.

Instruction description:

S: Source of data D: Destination of data

When MOV instruction is executed, the data contained in S will be directly moved to D. If the instruction is not executed, the contents in D will remain unchanged. To move the 32-bit operation result (such as application instruction MUL) and 32-bit current value of high speed counter, DMOV is required.

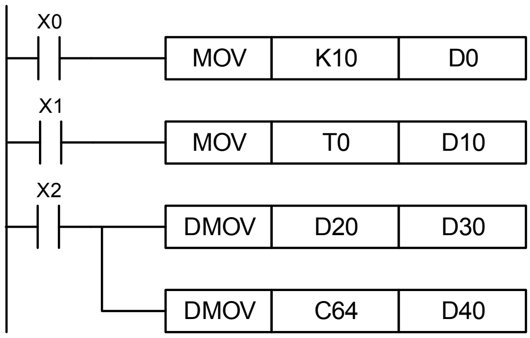
Example:

Move 16-bit data with MOV instruction:

a. When X0 = OFF, contents in D10 remain unchanged. If X0 = ON, data contained in K10 will be moved to register D10.
b. When X1 = OFF, contents in D10 remain unchanged. If X1 = ON, the current value of T0 will be moved to register D10.

Move 32-bit data with DMOV instruction:

When X2 = OFF, contents in (D31, D30) and (D41, D40) remain unchanged. If X2 = ON, the current value of (D21, D20) will be moved to data register (D31, D30); meanwhile, the current value of C64 will be moved to data register (D41, D40).



■ API-10 CML: Inverting transmission

API		CML				S, D			Inverting transmission	NC Series					
10	D														
	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S					*								*		
D													*		

16-bit instruction: CML continuous running type (4 STEP).
32-bit instruction: DCML continuous running type (5 STEP)
Flag: None

Notes on the use of operands:
If operands S and D are used in register Z, only 16-bit instruction is applicable.
Please see chapter 1 for details about the applicable range of each device.

Instruction description:

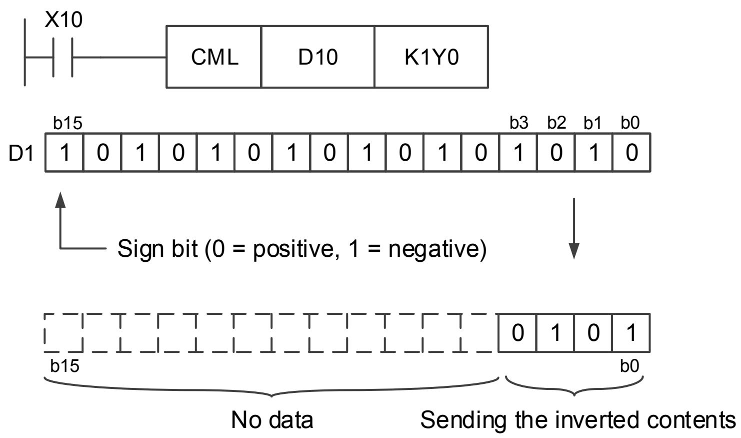
S: Source of data D: Destination device

The contents in S are inverted (0→1, 1→0) and sent to D. If the content is a K constant, it will be automatically inverted to BIN value.

Example 1:

CML instruction can be used for inverting content.

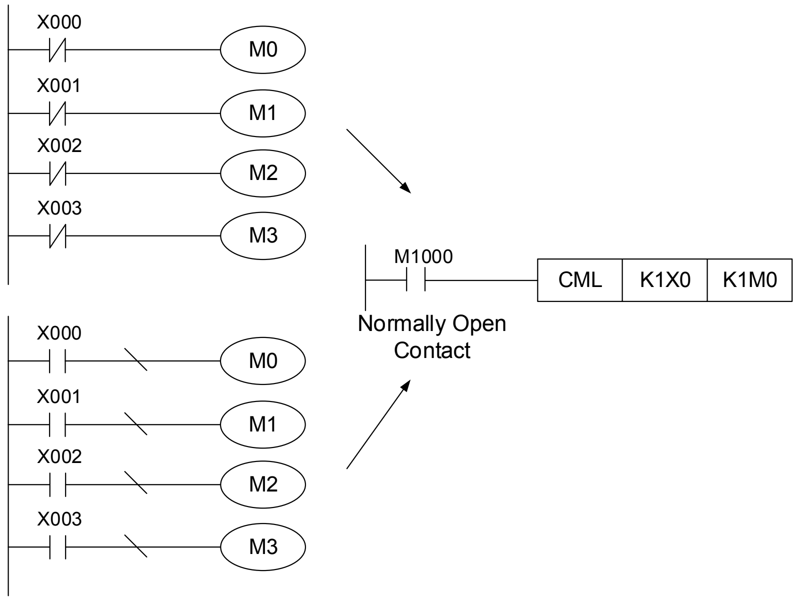
When X10 = ON, b0 ~ b3 in D1 are inverted and sent to Y0 ~ Y3.



4

Example 2:

The circuit shown to the left in the figure below can also be presented with a CML instruction (see right below).



■ API-11 BCD: BIN to BCD conversion

API		BCD			S, D			BIN to BCD conversion		NC Series				
11	D													
	Bit device				Word device									
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S													∗		
D													∗		

16-bit instruction: BCD continuous running type (4 STEP).
32-bit instruction: DBCD continuous running type (4 STEP)
Flag: M2930 (operation error); D1467 (error code)

Notes on the use of operands:
If operands S and D are used in register Z, only 16-bit instruction is applicable.
Please see chapter 1 for details about the applicable range of each device.

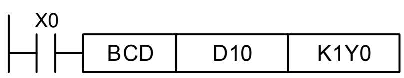
Instruction description:

S: Source of data D: Conversion result

The content in S (BIN value) is converted to BCD value and saved in D. When the result of BCD conversion exceeds K0 ~ K9,999, and M2930 = ON, D1467 register will record error code 0001. If the DBCD conversion result exceeds K0 ~ K99,999,999, and M2930 = ON, D1467 register will record error code 0001. MLC arithmetic operations and the execution of INC and DEC instructions are performed in BIN format. Thus, use BCD instruction to convert BIN values to BCD values if the user needs to see values displayed in decimal format.

Example:

When X0 = ON, BIN values in D10 are converted to BCD values, and the single digits of the conversion result will be saved in K4Y0 (Y0 ~ Y3), the four bit-devices. When D10 = 001E (Hex) = 0030 (decimal), the execution outcome is Y0 ~ Y3 = 0000(BIN).



■ API-12 BIN: BCD to BIN conversion

API		BCD	S, D	BCD to BIN conversion	NC Series
12	D				

	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S													*		
D													*		

16-bit instruction: BIN continuous running type (4 STEP).
32-bit instruction: DBIN continuous running type (4 STEP)
Flag: None.

Notes on the use of operands:
If operands S and D are used in register Z, only 16-bit instruction is applicable.
Please see chapter 1 for details about the applicable range of each device.

Instruction description:

S: Source of data D: Conversion result

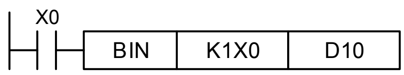
The contents in S (BCD value) are converted to BIN values and saved in D.

Valid range of contents in S: BCD (0 ~ 9,999), DBCD (0 ~ 99,999,999).

This instruction is not required for constant K and H as they are converted into BIN format automatically.

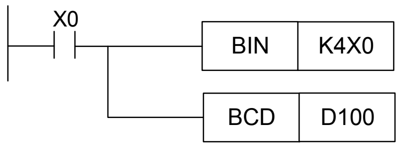
Example:

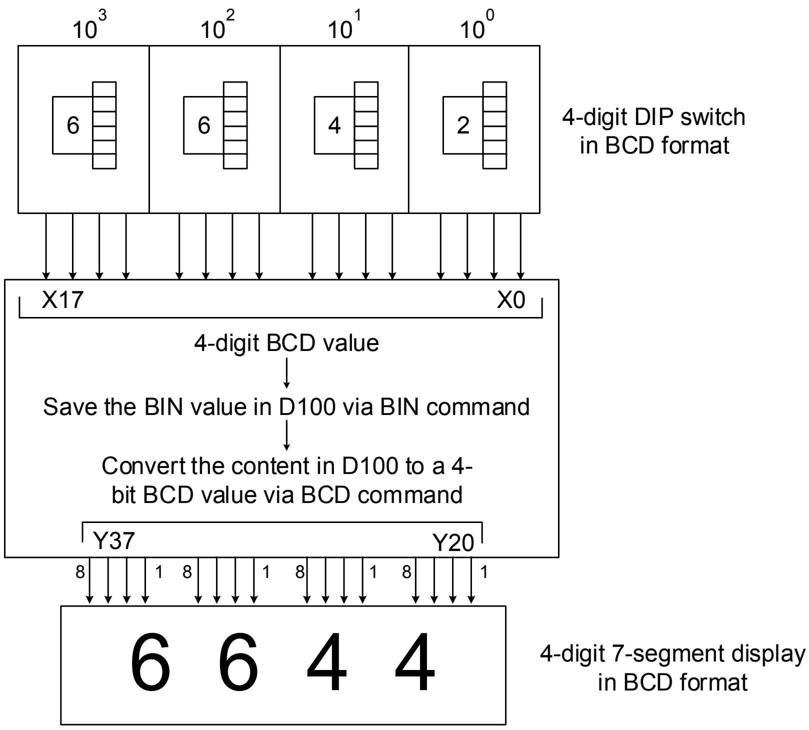
When X0 = ON, BCD values in K1X0 are converted to BIN values, and the outcome will be saved in D10.



Supplementary notes:

1. When MLC needs to read an external DIP switch state in BCD format, BIN instruction has to be adopted to convert the acquired data to BIN values and save the conversion result in MLC.
2. When MLC needs to display its stored data on a 7-segment display in BCD format, BCD instruction has to be adopted to convert the internal data to BCD values and send the result to the 7-segment display.
3. When X0 = ON, the BCD values in K4X0 are converted to BIN values and sent to D100, and then the received BIN values in D100 are converted to BCD values and sent to K4Y20 (see figure below).





4.3 Arithmetic and logic operation instructions

■ API-13 ADD: BIN Addition

API		ADD				S1, S2, D			BIN Addition		NC Series				
13	D														
	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S_1					*						*	*	*		
S_2					*						*	*	*		
D											*	*	*		

16-bit instruction: ADD continuous running type (6 STEP).
32-bit instruction: DADD continuous running type (8 STEP).
Flag: M2824 (zero flag), M2825 (borrow flag), M2826 (carry flag). See the supplementary notes below.

Notes on the use of operands:
If operands S_1, S_2 and D are used in register Z, only 16-bit instruction is applicable.
Please see chapter 1 for details about the applicable range of each device.

Instruction description:

S_1: Summand S_2: Addend D: Sum

This instruction adds data sources S_1 and S_2 in BIN format and saves the result in D. The very first bit of each data is a sign bit 0 (+) or 1 (-). This enables algebraic addition operations, such as 3 + (-9) = -6.

Flag changes in BIN addition:

16-bit BIN addition:

1. If the addition result is 0, zero flag M2824 is ON;
2. If the addition result is less than -32,768, borrow flag M2825 is ON;
3. If the addition result is larger than 32,767, carry flag M2826 is ON.

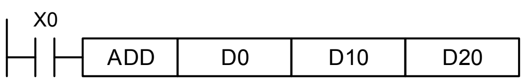
32-bit BIN addition:

1. If the addition result is 0, zero flag M2824 is ON;
2. If the addition result is less than -2,147,483,648, the borrow flag M2825 is ON;
3. If the addition result is larger than 2,147,483,647, the carry flag M2826 is ON.

Example 1:

16-bit BIN addition:

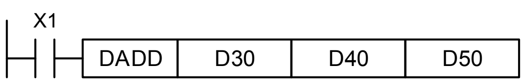
When X0 = ON, add summand D0 and addend D10 and save the result in D20.



Example 2:

32-bit BIN addition:

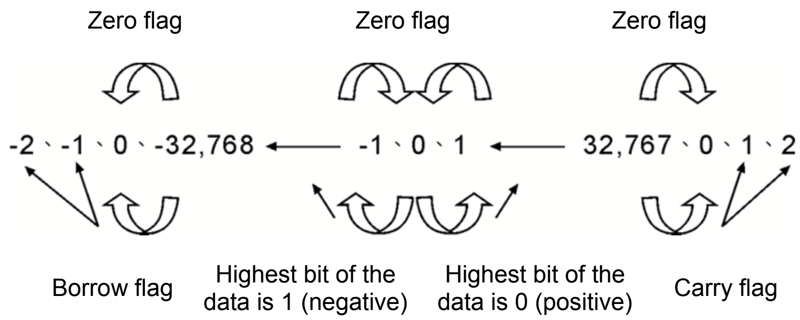
When X1 = ON, add summand (D31, D30) and addend (D41, D40) and save the result in (D51, D50). (D30, D40 and D50 are the lower 16-bit data, whereas D31, D41 and D51 are the higher 16-bit data.)



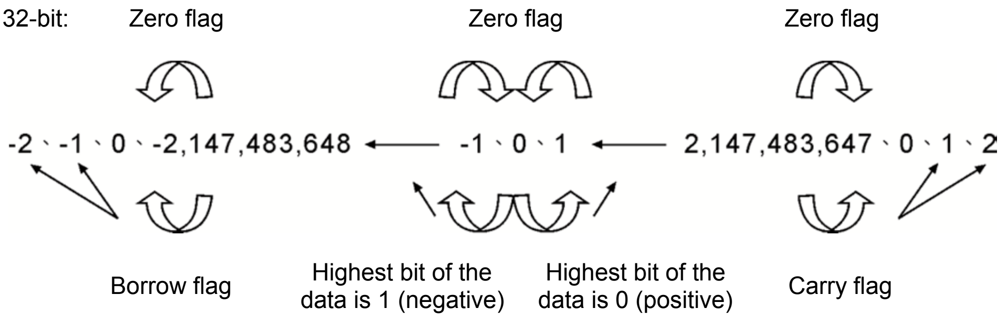
Supplementary notes:

Relations of the flags and the positive/negative sign of the values:

16-bit:



32-bit:



■ API-14 SUB: BIN Subtraction

API		SUB	S_1, S_2, D	BIN Subtraction	NC Series
14	D				

	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S_1					*						*	*	*		
S_2					*						*	*	*		
D											*	*	*		

16-bit instruction: SUB continuous running type (6 STEP).

32-bit instruction: DSUB continuous running type (8 STEP).

Flag: M2824 (zero flag), M2825 (borrow flag), M2826 (carry flag). See API-13 ADD supplementary notes.

Notes on the use of operands:

If operands S_1, S_2 and D are used in register Z, only 16-bit instruction is applicable.

Please see chapter 1 for details about the applicable range of each device.

Instruction description:

S_1: Minuend S_2: Subtrahend D: Difference

This instruction subtracts data sources S_1 and S_2 in BIN format and saves the result in D. The very first bit of each data is a sign bit 0 (+) or 1 (-), which enables algebraic subtraction operations.

Flag changes in BIN subtraction:

16-bit BIN subtraction:

1. If the subtraction result = 0, zero flag M2824 is ON;
2. If the subtraction result < -32,768, borrow flag M2825 is ON;
3. If the subtraction result > 32,767, carry flag M2826 is ON.

32-bit BIN subtraction:

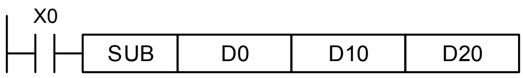
1. If the subtraction result = 0, zero flag M2824 is ON;
2. If the subtraction result < -2,147,483,648, the borrow flag M2825 is ON;
3. If the subtraction result > 2,147,483,647, the carry flag M2826 is ON.

See API-13 ADD supplementary notes for the relations of subtraction flags and the positive/negative sign of the values.

Example 1:

16-bit BIN subtraction:

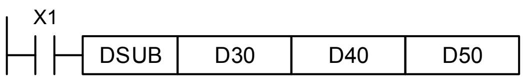
When X0 = ON, subtract the value of D10 from D0 and save the result in D20.



Example 2

32-bit BIN addition:

When X1 = ON, subtract the value of (D41, D40) from (D31, D30) and save the result in (D51, D50). (D30, D40 and D50 are the lower 16-bit data, whereas D31, D41 and D51 are the higher 16-bit data.)



■ API-15 MUL: BIN Multiplication

API		MUL				S1, S2, D			BIN Multiplication		NC Series				
15	D														
	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S_1					*						*	*	*		
S_2					*						*	*	*		
D											*	*	*		

16-bit instruction: MUL continuous running type (6 STEP).
32-bit instruction: DMUL continuous running type (8 STEP).
Flag: None.

Notes on the use of operands:
If operands S_1, S_2 and D are used in register V, only 16-bit instruction is applicable.
In 16-bit instruction, operand D takes consecutive 2 devices.
In 32-bit instruction, operand D takes consecutive 4 devices.
Please see chapter 1 for details about the applicable range of each device.

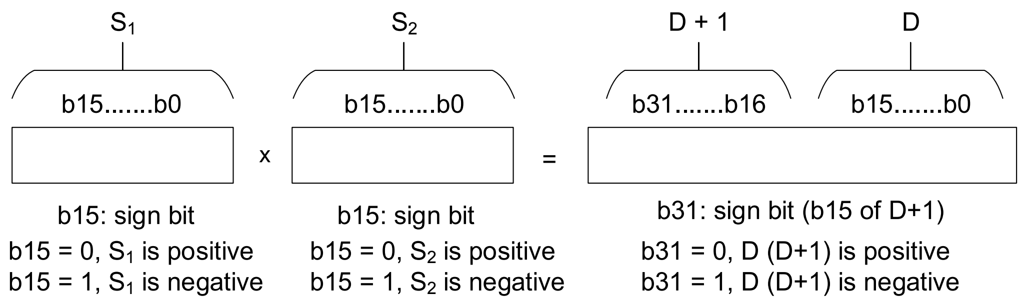
Instruction description:

S_1: Multiplicand S_2: Multiplier D: Product

This instruction multiplies values in data source S_1 and S_2 in signed binary and saves the product in D. When applying 16-bit and 32-bit operations, please pay close attention to the positive/negative signs of S_1, S_2 and D.

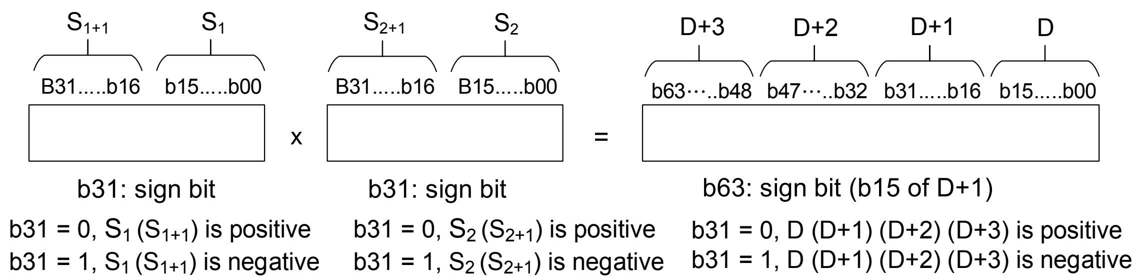
16-bit BIN multiplication:

If D serves as a bit device, K1 ~ K4 can be assigned to form a 16-bit data, occupying consecutive 2 groups of 16-bit data.



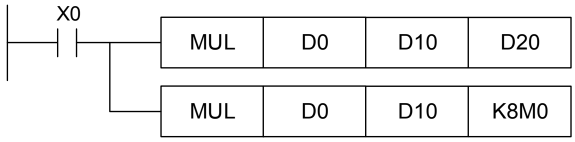
32-bit BIN multiplication:

If D serves as a bit device, K1 ~ K8 can be assigned to form a 32-bit data, storing lower 32-bit data only.



Example:

When X0 = ON, 16-bit D0 is multiplied by 16-bit D10 to obtain a 32-bit product. (multiply 16-bit register D0 and D10 and save the result in 32-bit register.) The higher 16 bits are stored in D21 and the lower 16 bits are stored in D20. ON/OFF of the most left bit indicates the positive/negative state of the value.



■ API-16 DIV: BIN Division

API		DIV	S_1, S_2, D	BIN Division	NC Series
16	D				

	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S_1					*						*	*	*		
S_2					*						*	*	*		
D											*	*	*		

16-bit instruction: DIV continuous running type (7 STEP).
32-bit instruction: DDIV continuous running type (13 STEP).
Flag: None.

Notes on the use of operands:
If operands S_1, S_2 and D are used in register V, only 16-bit instruction is applicable.
In 16-bit instruction, operand D takes consecutive 2 devices.
In 32-bit instruction, operand D takes consecutive 4 devices.
Please see chapter 1 for details about the applicable range of each device.

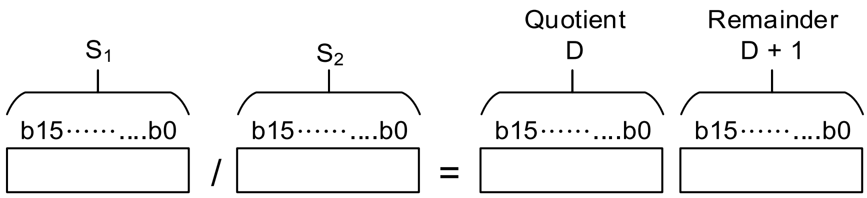
Instruction description:

S_1: Dividend S_2: Divisor D: Quotient and remainder

This instruction divides data source S_1 by S_2 in BIN format and saves the result in D. When applying 16-bit and 32-bit operations, please pay close attention to the positive/negative signs of S_1, S_2 and D. If the divisor is 0, this instruction will not be executed. And M2828 will be ON along with error code 0002 (hex) recorded in D1467.

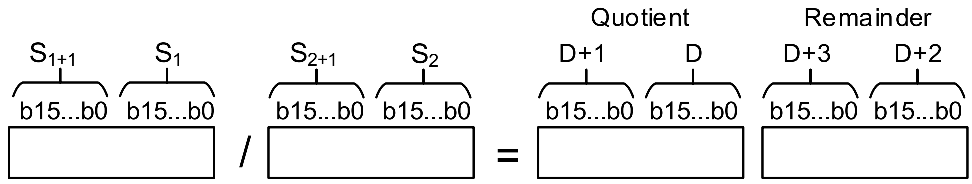
16-bit BIN division:

If D serves as a bit device, K1 ~ K4 can be assigned to form a 16-bit data, occupying consecutive 2 groups and bringing forth the quotient and remainder.



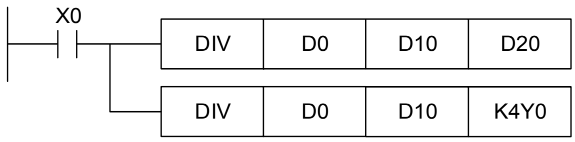
32-bit BIN division:

If D serves as a bit device, K1 ~ K8 can be assigned to form a 32-bit data, bringing forth the quotient with no remainder.



Example:

When X0 = ON, D0 is divided by D10 and the quotient is saved in D20 and remainder saved in D21. ON/OFF of the most left bit indicates the positive/negative state of the result value.



■ API-17 INC: Plus one (BIN)

API		INC		D			Plus one (BIN)	NC Series							
17	D														
	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
D											*	*	*		

16-bit instruction: INC continuous running type (3 STEP).
32-bit instruction: DINC continuous running type (3 STEP).
Flag: None.

Notes on the use of operands:
If operand D is used in register V, only 16-bit instruction is applicable.

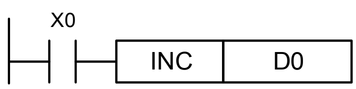
Instruction description:

D: Destination device

When INC instruction is executed, the value in the specified device D will plus 1 in every scan cycle of the program. In 16-bit operation, 32,767 plus 1 is -32,768. In 32-bit operation, 2,147,483,647 plus 1 is -2,147,483,648. The operation result of this instruction will not affect flag M2824 ~ M2826.

Example:

When X0 is OFF then ON, value in D0 will increase by 1 automatically.



4

■ API-18 DEC: Minus one (BIN)

API		DEC			D			Minus one (BIN)		NC Series					
18	D														
	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
D											*	*	*		

16-bit instruction: DEC continuous running type (3 STEP).
32-bit instruction: DDEC continuous running type (3 STEP).
Flag: None.

Notes on the use of operands:
If operand D is used in register V, only 16-bit instruction is applicable.

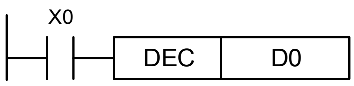
Instruction description:

D: Target device

When DEC instruction is executed, the value in the specified device D will minus 1 in every scan cycle of the program. In 16-bit operation, -32,768 minus 1 is 32,767. In 32-bit operation, -2,147,483,648 minus 1 is 2,147,483,647. The operation result of this instruction will not affect flag M2824 ~ M2826.

Example:

When X0 is OFF then ON, the value in D0 will decrease by 1 automatically.



4

■ API-19 WAND: AND operation

API		WAND				S_1, S_2, D			AND operation		NC Series				
19	D														
	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S_1					*								*		
S_2					*								*		
D													*		

16-bit instruction: WAND continuous running type (6 STEP).
32-bit instruction: DWAND continuous running type (8 STEP).
Flag: None.

Notes on the use of operands:
If operands S_1, S_2 and D are used in register Z, only 16-bit instruction is applicable.

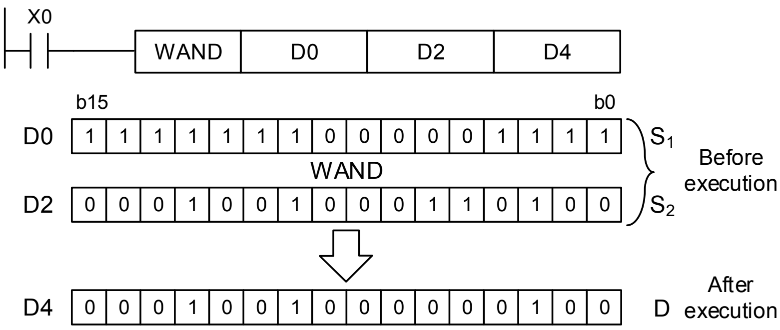
Instruction description:

S_1: Source data device 1 S_2: Source data device 2 D: Operation result

Do AND operation on data source S_1 and S_2 and save the result in D. According to the logic AND operation rule, the operation result in D will be 0 if any of S_1 or S_2 is 0. (Any value in AND operation shows 0, the result is 0.)

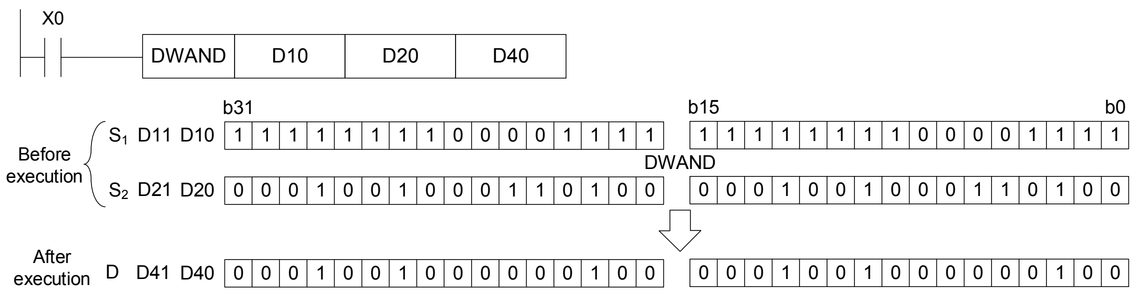
Example 1:

When X0 = ON, the 16-bit D0 and D2 perform WAND operation and the result will be saved in D4.



Example 2:

When X1 = ON, the 32-bit (D11, D10) and (D21, D20) perform DWAND operation and the result will be saved in (D41, D40).



4

■ API-20 WOR: OR operation

API				
20	D	WOR	S_1, S_2, D	OR operation	NC Series

	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S_1					*								*		
S_2					*								*		
D													*		

16-bit instruction: WOR continuous running type (6 STEP).
32-bit instruction: DWOR continuous running type (8 STEP).
Flag: None.

Notes on the use of operands:
If operands S_1, S_2 and D are used in register Z, only 16-bit instruction is applicable.

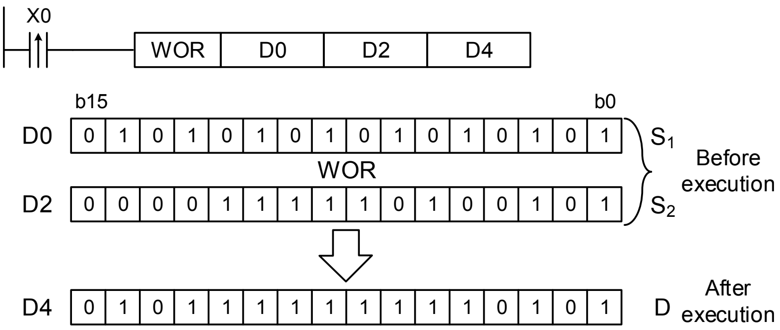
Instruction description:

S_1: Source data device 1 S_2: Source data device 2 D: Operation result

Do OR operation on data source S_1 and S_2 and save the result in D. According to the logic OR operation rule, the operation result in D will be 1 if any of S_1 or S_2 is 1. (Any value in OR operation is 1, the result is 1.)

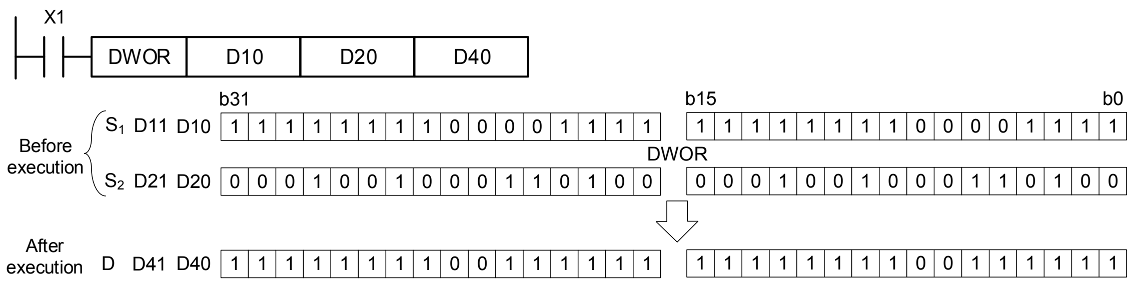
Example 1:

When X0 = ON, the 16-bit D0 and D2 perform WOR operation and the result will be saved in D4.



Example 2:

When X1 = ON, the 32-bit (D11, D10) and (D21, D20) perform DWOR operation and the result will be saved in (D41, D40).



■ API-21 WXOR: XOR / WXOR operation

API		WXOR	S_1, S_2, D	WXOR / XOR operation	NC Series
21	D				

	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S_1					*								*		
S_2					*								*		
D													*		

16-bit instruction: WXOR continuous running type (7 STEP).
32-bit instruction: DWXOR continuous running type (8 STEP).
Flag: None.

Notes on the use of operands:
If operands S_1, S_2 and D are used in register Z, only 16-bit instruction is applicable.

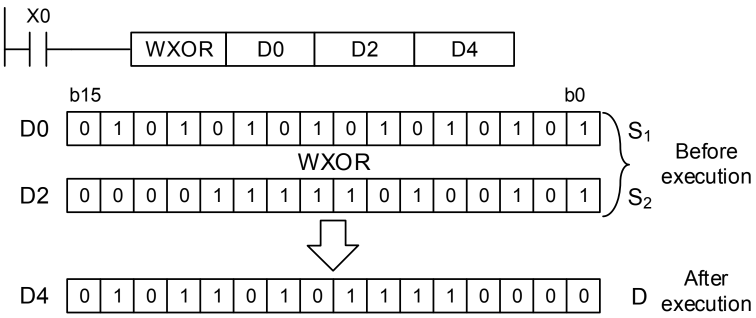
Instruction description:

S_1: Source data device 1 S_2: Source data device 2 D: Operation result

Do XOR operation on data source S_1 and S_2 and save the result in D. According to the logic XOR operation rule, if $S_1 = S_2$, the result in D is 0, and if $S_1 \neq S_2$, the result in D is 1. (In XOR operation, if both values are the same, the result will be 0, if not, the result will be 1.)

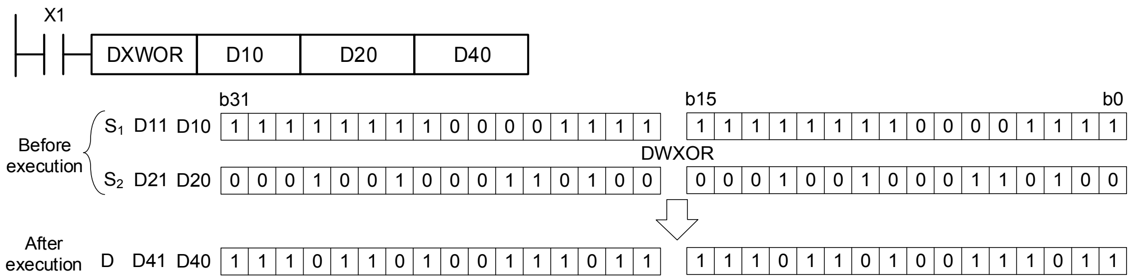
Example 1:

When X0 = ON, the 16-bit D0 and D2 perform WXOR operation and the result will be saved in D4.



Example 2:

When X1 = ON, the 32-bit (D11, D10) and (D21, D20) perform DWXOR operation and the result will be saved in (D41, D40).



■ API-22 NEG: 2's complement

API		NEG			D			2's complement			NC Series				
22	D														
	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
D													*		

16-bit instruction: NEG continuous running type (3 STEP).
32-bit instruction: DNEG continuous running type (3 STEP).
Flag: None.

Notes on the use of operands:
If operand D is used in register Z, only 16-bit instruction is applicable.

Instruction description:

D: The device requiring 2's complement

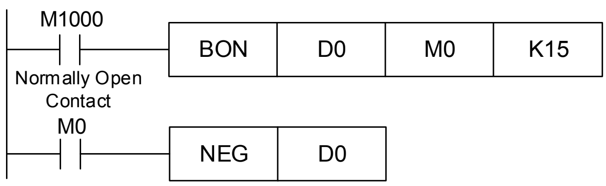
This instruction converts a negative BIN value into an absolute value.

Example 1:

When X0 is OFF then ON, every bit of the content in D10 is inverted (0→1, 1→0) and then add 1 to its value. The result will be stored in the original register D10.

Example 2:

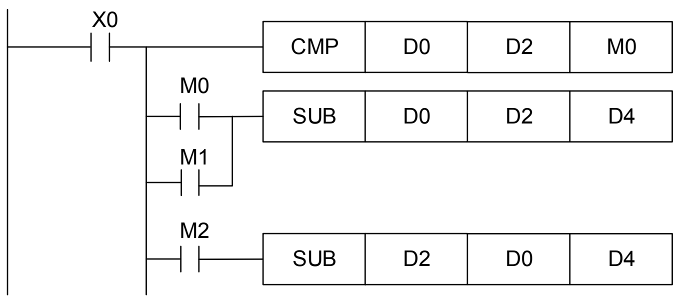
Convert a negative BIN value into an absolute value: When the 15th bit of D0 is 1, M0= ON (D0 is a negative value). When M0 = ON, NEG instruction obtains 2's compliment of D0 and further gets its absolute value.



Example 3:

Obtaining the absolute value of the difference from subtraction operation: When X0 = ON,

1. If D0 > D2, M0 is ON.
2. If D0 = D2, M1 is ON.
3. If D0 < D2, M2 is ON.
4. This ensures the value in D4 remain positive.



Supplementary notes:

Negative value and its absolute value:

The highest (most left) bit in the register is a sign bit, 0 indicates a positive value while 1 refers to a negative value. NEG instruction (API22) can be used to convert a negative value into its absolute value.

(D0) = 2

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

(D0) = 1

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

(D0) = 0

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

(D0) = -1

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

$(\overline{D0})$ +1 = 1

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

(D0) = -2

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

$(\overline{D0})$ +1 = 2

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

(D0) = -3

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

$(\overline{D0})$ +1 = 3

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

(D0) = -4

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

$(\overline{D0})$ +1 = 4

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

(D0) = -5

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

$(\overline{D0})$ +1 = 5

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

⋮

(D0) = -32,765

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

$(\overline{D0})$ +1 = 32,765

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

(D0) = -32,766

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

$(\overline{D0})$ + 1 = 32,766

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

(D0) = -32,767

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

$(\overline{D0})$+1 = 32,767

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

(D0) = -32,768

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

$(\overline{D0})$ + 1 = -32,768

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

Max. absolute value is 32,767

4

4.4 Rotate and shift instructions

■ API-23 ROR: Rotate right

API		ROR				D, n			Rotate right			NC Series		
23	D													
	Bit device				Word device									
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
D													∗		
n					∗										

16-bit instruction: ROR continuous running type (4 STEP).
32-bit instruction: DROR continuous running type (4 STEP).
Flag: M2826 (carry flag)

Notes on the use of operands:
If operand D is used in register Z, only 16-bit instruction is applicable.
If D is assigned to KnY, KnM, and KnS, only K4 (16-bit) and K8 (32-bit) are valid.
Range of n: K1 ~ K16 (16-bit), K1 ~ K32 (32-bit).

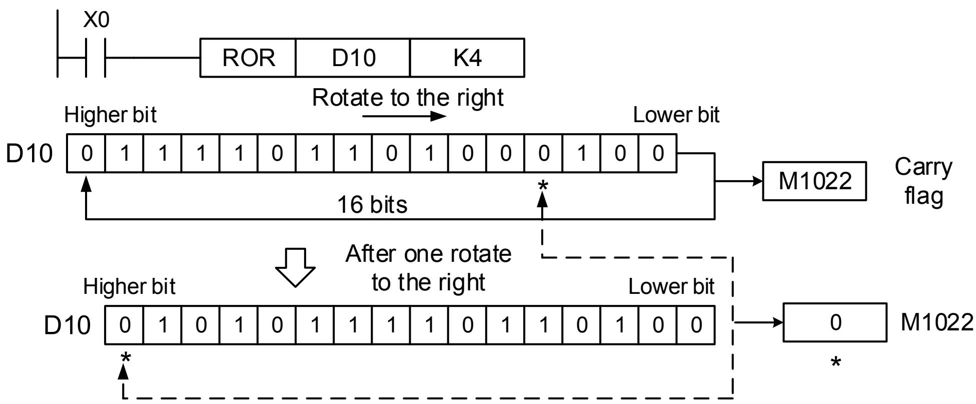
Instruction description:

D: The device to be rotated   n: Number of bits to be rotated in 1 rotation

This instruction rotates the device content assigned by D to the right for **n** bits.

Example:

When X0 changes from OFF to ON, the 16 bits in D10 rotates to the right in group of 4 bits. As shown in the figure below, the bit marked with * is sent to carry flag M1022.



■ API-24 ROL: Rotate left

API		ROL				D, n			Rotate left			NC Series			
24	D														
	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
D													*		
n					*										

16-bit instruction: ROL continuous running type (5 STEP).
32-bit instruction: DROL continuous running type (9 STEP).
Flag: M2826 (carry flag)

Notes on the use of operands:
If operand D is used in register Z, only 16-bit instruction is applicable.
If D is assigned to KnY, KnM, and KnS, only K4 (16-bit) and K8 (32-bit) are valid.
Range of n: K1 ~ K16 (16-bit), K1 ~ K32 (32-bit).

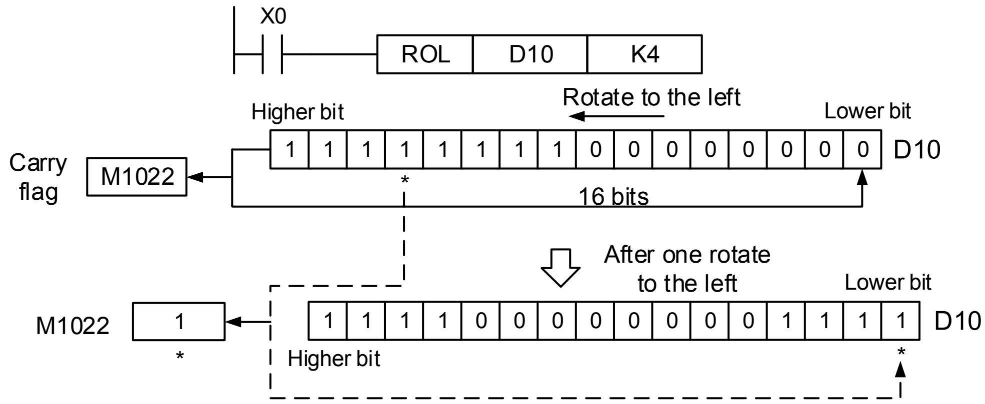
Instruction description:

D: The device to be rotated n: Number of bits to be rotated in 1 rotation

This instruction rotates the device content assigned by D to the left for **n** bits.

Example:

When X0 changes from OFF to ON, the 16 bits in D10 rotates to the left in group of 4 bits. As shown in the figure below, the bit marked with * is sent to carry flag M1022.



4.5 Data processing instructions

■ API-25 ZRST: Zone reset

API		ZRST				D1, D2		Zone reset			NC Series				
25	-														
	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
D_1		*	*	*							*	*	*		
D_2		*	*	*							*	*	*		

16-bit instruction: ZRST continuous running type (4 STEP).
32-bit instruction: None.
Flag: None.

Notes on the use of operands:

If the No. of $D_1 \leqq$ the No. of D_2, D_1 and D_2 have to specify the same type of device.

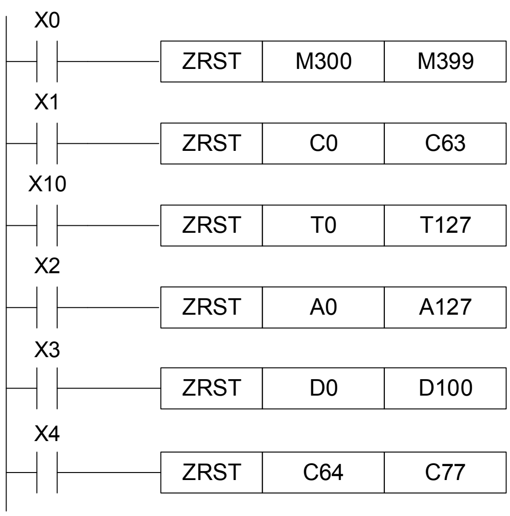
Instruction description:

D_1: Starting device of zone reset D_2: Ending device of zone reset

In NC series models, the 16-bit and 32-bit counters cannot use ZRST instruction at the same time. When the No. of D_1 is larger than the No. of D_2, only the device assigned by D2 will be reset.

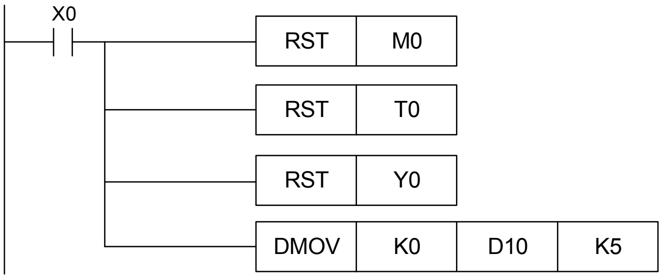
Example:

1. When X0 = ON, auxiliary relays M300 ~ M399 are reset to OFF.
2. When X1 = ON, 16-bit counters C0 ~ C63 are all reset (Set the value to 0; contacts and coils are reset to OFF).
3. When X10 = ON, timer T0 ~ T127 are all reset (write in value 0; contacts and coils are reset to OFF).
4. When X2 = ON, alarm flags A0 ~ A127 are all reset to OFF.
5. When X3 = ON, data registers D0 ~ D100 are all reset to 0.
6. When X4 = ON, 32-bit counters C64 ~ C77 are all reset (Set the value to 0; contacts and coils are reset to OFF).



Supplementary notes:

Devices, such as bit device Y, M, A and word device T, C, D, can use RST instruction independently. Likewise, instruction DMOV (API 09) can be used to send K0 to word device T, C, D or bit register KnY, KnM, KnA for resetting purpose (see figure below).



■ API-26 DECO: Decoder

API		DECO	S, D, n	Decoder	NC Series
26	-				

	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S	*	*	*	*	*								*		
D		*	*	*									*		
n					*										

16-bit instruction: DECO continuous running type (6 STEP).
32-bit instruction: None.
Flag: None.

Notes on the use of operands:
Range of n: n = 1 ~ 8 when D is a bit device; n = 1 ~ 4 when D is a word device.

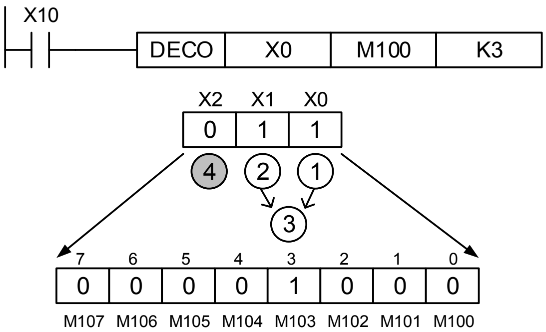
Instruction description:

S: Source device for decoding D: Device for saving the decoded result n: Length of decoded bits

The lower **n** bits of S are decoded and the results with **2^n** bits length are saved in D.

Example 1:

1. When D is a bit device, n = 1 ~ 8. Error will occur if n = 0 or n > 8.
2. When n = 8, this instruction can decode up to 2^8 = 256 points. (Please be aware of the devices' storage range after decoding, and do not use any device repeatedly.)



a. When X10 is OFF then ON, DECO instruction decodes values in X0 ~ X2 to M100 ~ M107.

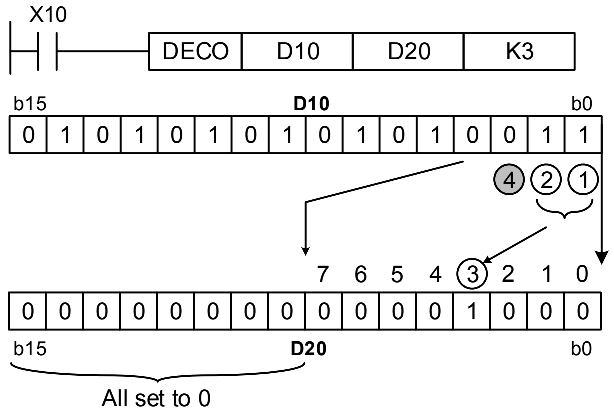
b. If the data source is 1 + 2 = 3, the third bit M103 of M100 is set to 1.

c. After DECO instruction is complete and X10 is OFF, the content that has been decoded will remain its state.

4

Example 2:

1. When D is a word device, n = 1 ~ 4. Error will occur if n = 0 or n > 4.
2. When n = 4, this instruction can decode up to 2^4 = 16 points.



a. When X10 is OFF then ON, DECO instruction decodes values in (b2 ~ b0) of D10 to (b7 ~ b0) of D20. The bits (b15 ~ b8) in D20 that have not been used are all set to 0.
b. The lower 3 bits of D10 are decoded and saved in the lower 8 bits of D20. The higher 8 bits are all set to 0.
c. When DECO instruction is complete and X10 = OFF, output result of the decoding will keep being processed.

■ API-27 ENCO: Encoder

API		ENCO			S, D, n			Encoder			NC Series				
27	-														
	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S	*	*	*	*	*								*		
D													*		
n					*										

16-bit instruction: ENCO continuous running type (6 STEP).
32-bit instruction: None.
Flag: None.

Notes on the use of operands:
Range of n: n = 1 ~ 8 when D is a bit device; n = 1 ~ 4 when D is a word device.

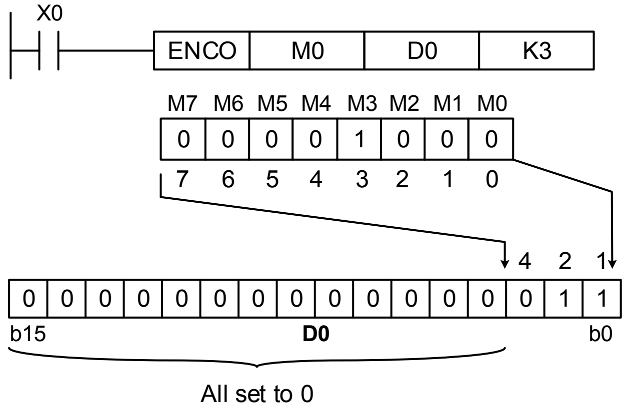
Instruction description:

S: Source device for encoding D: Device for saving encoded value n: Length of encoded bits

The lower $\mathbf{2^n}$ bits of S are decoded and the result is saved in D. If there is more than 1 bit in device S is 1, its lower bits will not be processed.

Example 1:

1. When S is a bit device, n = 1 ~ 8. Error will occur if n = 0 or n > 8.
2. When n = 8, this instruction can encode up to 2^8 = 256 points.

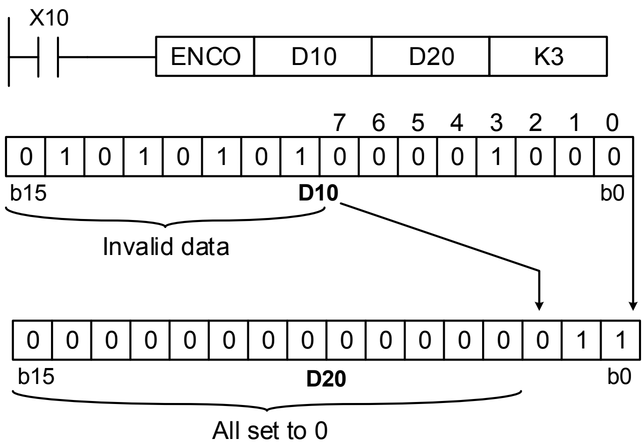


a. When X0 is OFF then ON, ENCO instruction encodes 2^3 bits data ~~in~~ (M0 ~ M7) and saves the result in the lower 3 bits (b2 ~ b0) of D0. The bits (b15 ~ b3) in D0 that have not been used are all set to 0.

b. When ENCO instruction is complete and X0 = OFF, data in D remains unchanged.

Example 2:

1. When S is a word device, n = 1 ~ 4. Error will occur if n = 0 or n > 4.
2. When n = 4, this instruction can encode up to 2^4 = 16 points.



a. When X0 is OFF then ON, ENCO instruction encodes 2^3 bits data ~~in~~ (b0 ~ b7) and saves the result in the lower 3 bits (b2 ~ b0) of D20. The bits (b15 ~ b3) in D20 that have not been used are all set to 0. (b8 ~ b15 in D10 are invalid data.)

b. When ENCO instruction is complete and X0 = OFF, data in D stays the same.

■ API-28 BON: Monitor specified bit state

API		BON				S, D, n			Monitor specified bit state			NC Series			
28	D														
	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S					*						*	*	*		
D		*	*	*											
n					*										

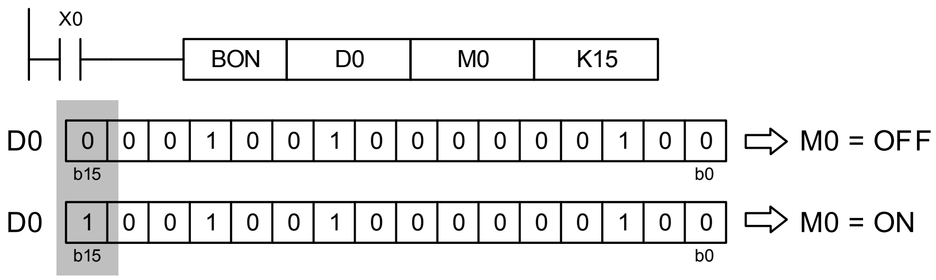
16-bit instruction: BON continuous running type (6 STEP).
32-bit instruction: DBON continuous running type (7 STEP).
Flag: None.

Notes on the use of operands:
If operand S is used in register Z, only 16-bit instruction is applicable.
Range of n: n = 0 ~15 (16-bit instruction); n = 0 ~ 31 (32-bit instruction)

Instruction description:

S: Source device D: Device for storing the result n: Monitoring bit (start at 0)

Example:



a. When X0 = ON, M0 goes ON if the 15th bit of D0 is 1. If it is 0, M0 goes OFF.

b. When X0 = OFF, M0 remains its previous state.

■ API-29 ANS: Alarm trigger

API		ANS	S, m, D	Alarm trigger	NC Series
29	-				

	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S											*				
m					*										
D				*											

16-bit instruction: ANS continuous running type (8 STEP).
32-bit instruction: None.
Flag: None.

Notes on the use of operands:
Range of S: T0 ~ T255 in NC series models.
Range of m: K1 ~ K32, 767, unit can be 100 ms or 10 ms that is determined by T(n), n = 0 ~ 255.
Range of D: A0 ~ A511 in NC series models.
T0 ~ T199 (unit: 100ms), T200 ~ T255 (unit: 10ms).

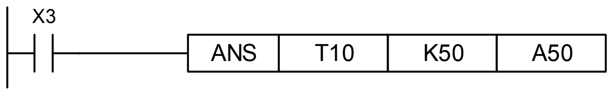
Instruction description:

S: Timer for alarm detection m: Time setting D: Alarm device

This instruction is used for triggering the alarm.

Example:

When X3 has been ON for more than 5 seconds, the alarm flag A50 goes ON. Afterwards, A50 will remain ON even if X3 goes OFF. (But T10 will be reset to OFF, and its current value is reset to 0.)



■ API-30 ANR: Alarm clear

API		ANS	-	Alarm clear	NC Series
30	-				

	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z

16-bit instruction: ANR continuous running type (1 STEP).
32-bit instruction: None.
Flag: None.

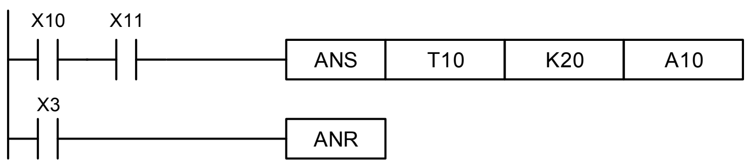
Notes on the use of operands: No operand.

Instruction description:

This instruction is used for clearing the alarm. When more than one alarm are triggered, the alarm with a smaller No. will be cleared.

Example:

1. When X10 and X11 have been ON simultaneously for more than 2 seconds, the alarm flag A10 goes ON. Afterwards, A10 will remain ON even if X10 and X11 turn OFF. (But T10 will be to OFF, and its current value is 0.)
2. When X10 and X11 are ON simultaneously for less than 2 seconds, the current value of T10 will be set to 0.
3. When X3 goes ON from OFF, the currently active alarm will be cleared. NC series models can use alarm flags A0 ~ A511.
4. When X3 goes ON from OFF again, the alarm with the second small No. will be cleared.



4.6 High-speed processing instructions

■ API-31 REF: I/O refresh

API		REF	D, n	I/O refresh	NC Series
31	-				

	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
D	*	*													
n					*										

16-bit instruction: REF continuous running type (3 STEP).
32-bit instruction: None.
Flag: None

Notes on the use of operands:
Operand D must assign X0, X10, Y0 and Y10 whose No. ends with 0. See supplementary notes.
Range of n: n = 8 ~ 256, and n has to be a multiple of 8.

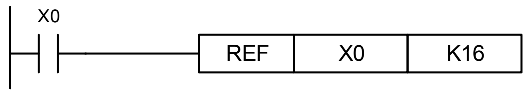
Instruction description:

D: Starting device of I/O refresh   n: Number of devices to be I/O refreshed

1. The state of all MLC input/output contacts will be updated after the program scans to END instruction. When the program starts to scan, the state of the external input contact is read and stored into the memory (for input contact). The output contact will send the content in memory (for output point) to the output device after END instruction is executed. Therefore, this instruction is applicable when the latest input/output data are required for the operation.
2. Operand D must assign X0, X10, Y0 and Y10, of which No. ends with 0. The range of **n** is 8 ~ 256 that has to be a multiple of 8. Other numbers will be regarded as errors. The application range varies with the models. Please refer to the supplementary notes below.

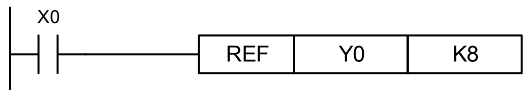
Example 1:

When X0 = ON, MLC immediately reads the state of the input contacts X0 ~ X15 and updates the input signals without any delay.



Example 2:

When X0 = ON, the 8 output signals from Y0 ~ Y7 are immediately sent to the output contacts and refreshed without waiting for the END instruction.



Supplementary notes:

NC series models can process input/output contacts of I/O and RIO, i.e. n = k8 or n= k16.

API-32 DHSCS: Compare setup (high-speed counter)

API	-	DHSCS	S_1, S_2, D	Compare setup (high-speed counter)	NC Series
32	D				

	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S_1					*								*		
S_2												*			
D		*	*	*											

16-bit instruction: None.

32-bit instruction: DHSCS continuous running type (5 STEP).

Flag: M2872 ~ M2873, for disabling high-speed counter interruption. Please refer to Example 3.

Notes on the use of operands:

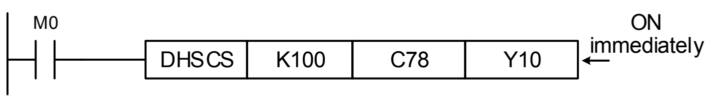
- Operand S_2 has to assign high-speed counter C78 and C79. See supplementary notes below.
- Operand D can assign IC00, IC01, and D can be modified by indirect register V, Z.
- Please see chapter 1 for details about the applicable range of each device.
- This instruction is valid for 32-bit instruction DHSCS only.
- This instruction sets up a comparison value for high speed counters.
- The high-speed counter counts by hardware. When the counter reaches the set value, the external signal will interrupt the current program. And the D operand will be ON.

Instruction description:

S_1: Comparing value S_2: No. of high speed counter D: Comparing result

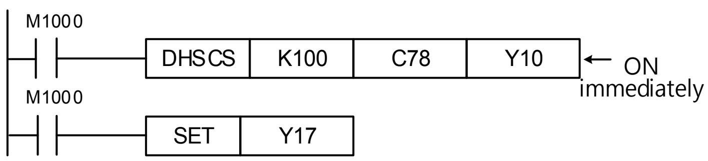
1. The high-speed counter is triggered by external high-speed input signal. When the high speed counter specified by S_2 adds 1 or subtracts 1, DHSCS instruction will carry out comparison immediately. When the counter’s current value equals the comparative value specified by S_1, device specified by D will turn ON and remain ON even if the comparison results become unequal afterwards.
2. If the devices specified by D are Y0 ~ Y23 (only On board Y) and the comparing value equals the current value of the high speed counter, the comparison result will output instantly to external output contact Y0 ~ Y23 (only On board Y). Other Y devices will still be affected by the scan cycle. M and A devices act immediately without being affected by the scan cycle.

Example 1:



When RUN instruction is executed and M0 = ON, DHSCS instruction will be executed. If the current value of C78 changes from 99 to 100 or from 101 to 100, Y10 will be set to ON. It will instantly output to external output contact and Y10 will remain ON.

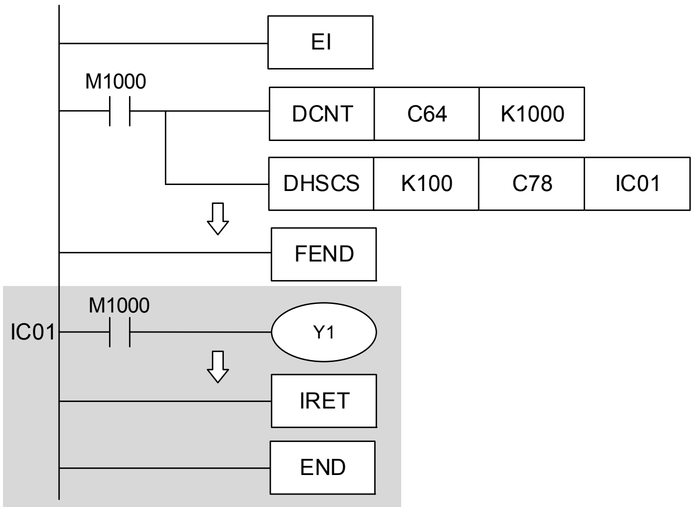
Example 2:



Differences between the Y output of DHSCS instruction and general Y output:

1. When the current value of C79 changes from 99 to 100 and from 101 to 100, Y10 of DHSCS instruction outputs immediately to the external output contacts by inserting an interruption, which is irrelevant to MLC scan cycle. However, the output time will still be delayed by the relay (10 ms) or transistor (10 us).
2. When the current value of C79 changes from 99 to 100, contact C79 will be set to ON immediately. When the program is running to SET Y17, Y17 will still be affected by the scan cycle and it will output after END instruction is executed.

Example 3:



1. Operand D of DHSCS instruction can assign IC00 and IC01 for the timing of interruption. That is, when the counter reaches its set value, interruption will occur.
2. For NC series models, there are limits of using high-speed counter for interruption: When DHSCS instruction assigns an I for interruption, the high-speed counter cannot be used in other DHSCS and DHSCR instructions. Incorrect use of high-speed counter will cause program errors.
3. For NC series models, when the active high-speed counter meets its set value, the interruption will occur. C78 serves as the first counter and the No. of the interrupt indicators are specified as IC00 or IC01.
4. When the current value of C78 changes from 99 to 100 or from 101 to 100 (applying MLC #312 parameter for counting down), the program will jump to indicator IC01 and carry out interruption service subroutine.

NC series models M2872 ~ M2873 refer to high-speed counters IC00 ~ IC01 respectively, which means interrupt IC0 is disabled if M2872 = OFF.

Interrupt No.	Interrupt disabling flag
IC00	M2872
IC01	M2873

4

■ API-33 DHSCR: Compare reset (high-speed counter)

API	-	DHSCR			S_1, S_2, D			Compare reset (high speed counter)		NC Series					
33	D														
	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S_1					*								*		
S_2												*			
D		*	*	*											

16-bit instruction: None.
32-bit instruction: DHSCR continuous running type (5 STEP).
Flag: None.

Notes on the use of operands:

- Operand S2 has to assign high-speed counter C78 and C79. See API-32 DHSCS supplementary notes.
- Operand D can assign counters C78 ~ C79 that has the same No. with the counters assigned by S2.
- This instruction is valid for 32-bit instruction DHSCR only.
- This instruction sets up a comparison value for high-speed counters.
- The high-speed counter counts by hardware. When the counter reaches the set value, the external signal will interrupt the current program. And the D operand will be OFF.

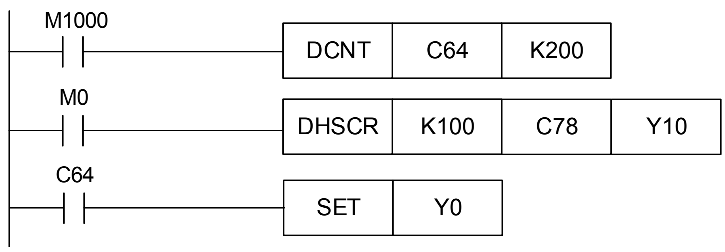
Instruction description:

S_1: Comparing value S_2: No. of high speed counter   D: Comparing result

1. The high-speed counter is triggered by external high-speed input signal. When the high speed counter specified by S_2 adds 1 or subtracts 1, DHSCR instruction will carry out comparison immediately. When the counter's current value equals the comparative value S_1, device specified by D will turn OFF and remain OFF even if the comparison results become unequal afterwards.
2. If the devices specified by D are Y0 ~ Y23 (only On board Y) and the comparing value equals the current value of the high-speed counter, the comparison result will output instantly to external output contact Y0 ~ Y23 (only On board Y). Other Y devices are still affected by the scan cycle, while M and A devices change their ON/OFF state without being affected by the scan cycle.

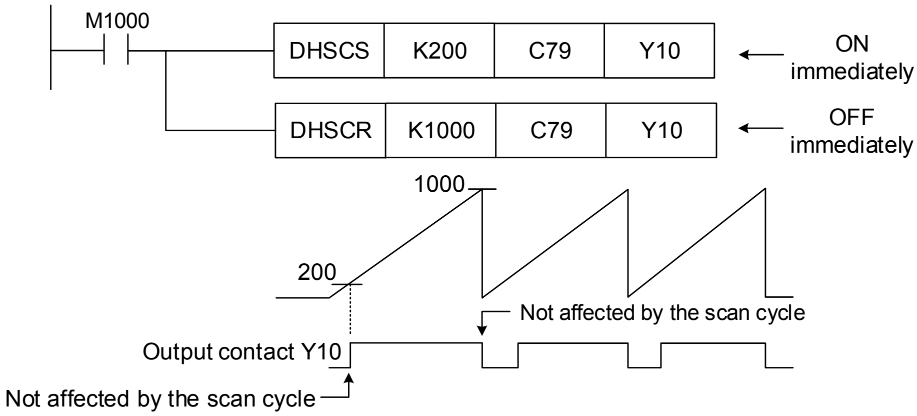
Example 1:

1. When M0 = ON and C78's current value changes from 99 to 100 or from 101 to 100, Y10 will be cleared and set to OFF.
2. When C64's current value changes from 199 to 200, the contact of C64 will be set to ON that will make Y0 = ON. However, the output will be delayed by the program scan time.
3. Y10 immediately reset its state after reaching its set value. And D can assign high-speed counters of the same No. See Example 2.



Example 2:

If DHSCR instruction assigns the same high-speed counter and C79's current value changes from 999 to 1,000 or from 1,001 to 1,000, the contact of C79 will be cleared and set to OFF.



4.7 Convenience instructions

■ API-34 ALT: ON/OFF alternate

API		ALT				D			ON/OFF alternate	NC Series					
34	-														
	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
D		＊	＊	＊											

16-bit instruction: ALT continuous running type (3 STEP).
32-bit instruction: None.
Flag: None

Notes on the use of operands:
Please see chapter 1 for details about the applicable range of each device.

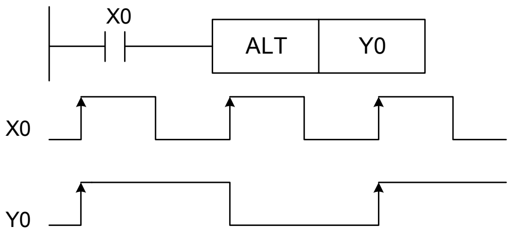
Instruction description:

D: Target device

This instruction is usually used as an execution instruction (ALT).

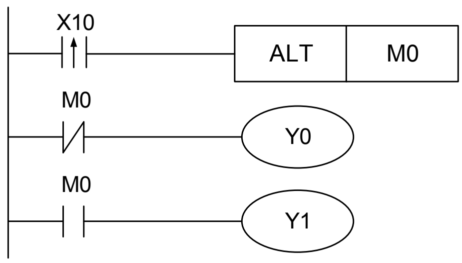
Example 1:

When X0 changes from OFF to ON for the first time, Y0 turns ON. When X0 goes ON for the second time, Y0 turns OFF.



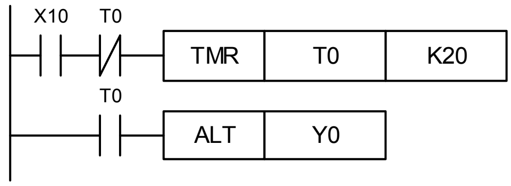
Example 2:

Using a single switch to enable and disable control. At the beginning, M0 = OFF., thus Y0 = ON and Y1 = OFF.. When X10 switches between ON / OFF., M0 = ON, thus Y1 = ON and Y0 = OFF.. For the second time of X10 ON / OFF. switching, M0 = OFF., thus Y0 = ON and Y1 = OFF..



Example 3:

ALT instruction can be used to enable Y0 flashing. When X10 = ON, T0 changes its ON/OFF state every 2 seconds. Then, ALT enables Y0 output to switch between ON and OFF every time T0 changes its state.



4.8 Contact type comparing instructions

■ API-39 ~ 44 LD※ : Contact type compare

API		LD※	S_1, S_2	Contact type compare LD※	NC Series
39~44	D				

	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S_1					*						*	*	*		
S_2					*						*	*	*		

16-bit instruction: LD※ continuous running type (4 STEP).

32-bit instruction: DLD※ continuous running type (6 STEP).

Flag: None

Notes on the use of operands:
Please see chapter 1 for details about the applicable range of each device.

※: =, >, <, <>, ≦, ≧

Instruction description:

S_1: Data source device 1 S_2: Data source device 2

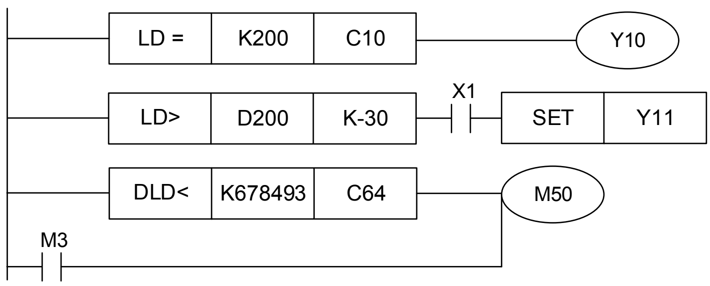
1. This instruction compares contents stored in S_1 and S_2. Take API 39(LD=) instruction as an example: if the comparison result satisfies the condition, the contact turns on.
2. LD※ instruction may connect to a bus bar directly. See the table below:

API No.	16-bit instruction	32-bit instruction	Turn-on condition	Not turn-on condition
39	LD =	DLD =	$S_1 = S_2$	$S_1 \neq S_2$
40	LD >	DLD >	$S_1 > S_2$	$S_1 \leqq S_2$
41	LD <	DLD <	$S_1 < S_2$	$S_1 \geqq S_2$
42	LD < >	DLD < >	$S_1 \neq S_2$	$S_1 = S_2$
43	LD < =	DLD < =	$S_1 \leqq S_2$	$S_1 > S_2$
44	LD > =	DLD > =	$S_1 \geqq S_2$	$S_1 < S_2$

3. Use 32-bit instruction (DLD※) to compare 32-bit counters (C64 ~ C77). Program error will occur if 16-bit instruction (LD※) is used, and ERROR light indicator on the main board will be flashing.

Example:

1. When content in C10 equals K200, Y10 turns ON.
2. When content in D200 is greater than K-30 and X1 = ON, Y11 turns ON and remains.
3. When content in C64 is less than K678, 493 or M3 = ON, M50 turns ON .



■ API-45 ~ 50 AND※ : Contact type compare

API		AND※				S1, S2			Contact type compare AND※		NC Series				
45~50	D														
	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S_1					*						*	*	*		
S_2					*						*	*	*		

16-bit instruction: AND※ continuous running type (4 STEP).

32-bit instruction: DAND※ continuous running type (6 STEP).

Flag: None

Notes on the use of operands:
Please see chapter 1 for details about the applicable range of each device.
※: =, >, <, <>, ≦, ≧

Instruction description:

S_1: Data source device 1 S_2: Data source device 2

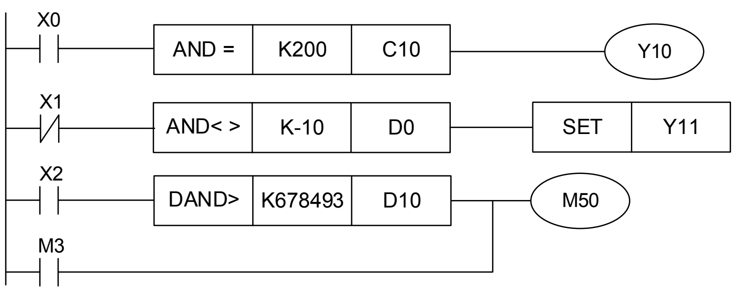
1. This instruction compares contents stored in S_1 and S_2. Take API 45(AND=) instruction as an example: When the comparing result satisfies the condition, the contact turns on.
2. AND※ is a comparing instruction that serial connects to a contact, as shown below:

API No.	16-bit instruction	32-bit instruction	Turn-on condition	Not turn-on condition
45	AND =	DAND =	$S_1 = S_2$	$S_1 \neq S_2$
46	AND >	DAND >	$S_1 > S_2$	$S_1 \leqq S_2$
47	AND <	DAND <	$S_1 < S_2$	$S_1 \geqq S_2$
48	AND < >	DAND < >	$S_1 \neq S_2$	$S_1 = S_2$
49	AND < =	DAND < =	$S_1 \leqq S_2$	$S_1 > S_2$
50	AND > =	DAND > =	$S_1 \geqq S_2$	$S_1 < S_2$

3. Use 32-bit instruction (DAND※) to compare 32-bit counters (C64 ~ C77). Program error will occur if 16-bit instruction (AND※) is used, and ERROR light indicator on the main board will be flashing.

Example:

1. When X0 = ON and content in C10 equals K200, Y10 turns ON.
2. When X1 = OFF and content in D0 does not equal K-30, Y11 turns ON and remains.
3. When X2 = ON and content in 32-bit register D10 (D11) is less than K678,493 or M3 = ON, M50 will turns ON.



■ API-51 ~ 56 OR※ : Contact type compare

API		OR※	S1, S2	Contact type compare OR※	NC Series
51~56	D				

	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S_1					*						*	*	*		
S_2					*						*	*	*		

16-bit instruction: OR※ continuous running type (4 STEP).

32-bit instruction: DOR※ continuous running type (6 STEP).

Flag: None

Notes on the use of operands:
Please see chapter 1 for details about the applicable range of each device.
※: =, >, <, <>, ≦, ≧

Instruction description:

S_1: Data source device 1 S_2: Data source device 2

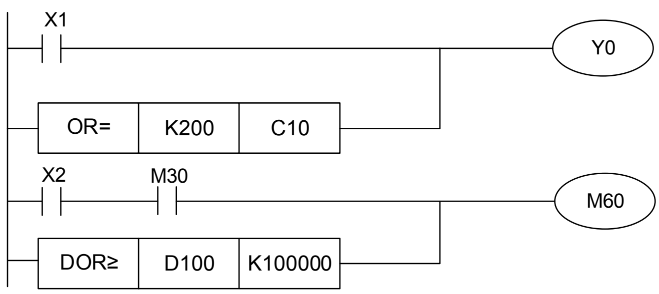
1. This instruction compares contents stored in S1 and S2. Take API 51(OR=) instruction as an example: When the comparing result satisfies the condition, the contact turns on.
2. OR※ is a comparing instruction that parallel connects to a contact, as shown below:

API No.	16-bit instruction	32-bit instruction	Turn-on condition	Not turn-on condition
51	OR =	DOR =	$S_1 = S_2$	$S_1 \neq S_2$
52	OR >	DOR >	$S_1 > S_2$	$S_1 \leqq S_2$
53	OR <	DOR <	$S_1 < S_2$	$S_1 \geqq S_2$
54	OR < >	DOR < >	$S_1 \neq S_2$	$S_1 = S_2$
55	OR < =	DOR < =	$S_1 \leqq S_2$	$S_1 > S_2$
56	OR > =	DOR > =	$S_1 \geqq S_2$	$S_1 < S_2$

3. Use 32-bit instruction (DOR※) to compare 32-bit counters (C64 ~ C77). Program error will occur if 16-bit instruction (OR※) is used, and ERROR light indicator on the main board will be flashing.

Example:

1. When X1 = ON and content in C10 equals K200, Y0 turns ON.
2. When both X2 and M30 are ON, or content in 32-bit register D100(D101) is greater or equals K100, 000, then M60 turns ON.



■ API-57 VRT: Variable table

API		VRT			S, n, D			Variable table			NC Series				
57	D														
	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S	*	*	*								*	*			
n					*										
D													*		

16-bit instruction: VRT continuous running type (70 STEP).
32-bit instruction: DVRT continuous running type (134 STEP).
Flag: None.

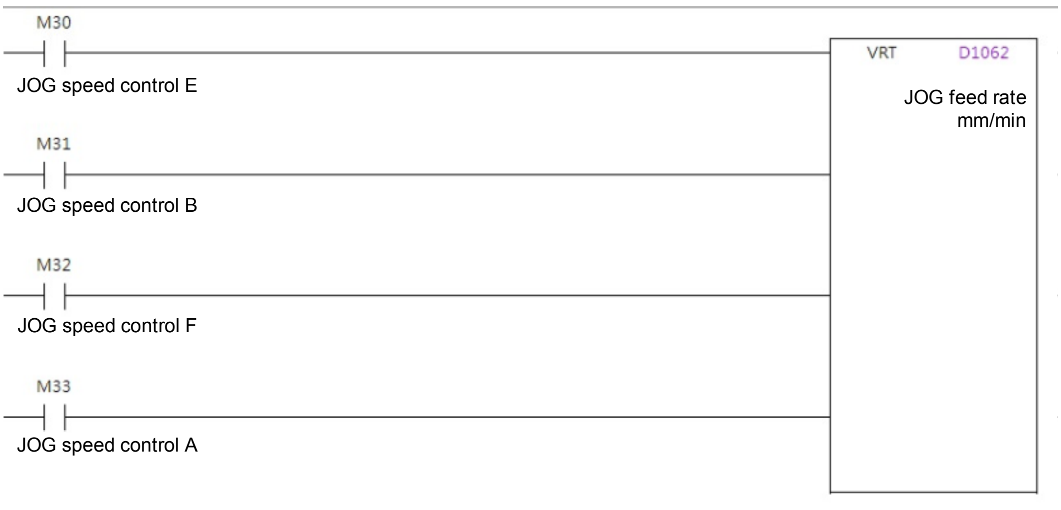
Notes on the use of operands: No operand.

Instruction description:

S: Source device to be switched n: Number of source devices D: Result

This instruction assigns the initial source device specified by S, and then designates multiple sequential source devices. When the device switches the state, the value will be changed according to the variable table and saved in register D, X, Y, M, T or C. Please note that register D, X, Y, M T or C can be specified as the source device.

Example 1:



Variable table:

	+0	+1	+2	+3	+4
0	0	20	32	50	79
5	126	200	320	500	790
10	1260	2000	3200	5000	7900
15	12600				

When M30 = ON, M31 = ON, M32 = OFF and M33 = OFF, M30 ~ M33 is 3 in binary format, and its correlating value in variable table is 50. Thus, D1062 = 50.

4.9 Floating point operation instructions

■ API-58 FADD: Binary floating point addition

API	-	FADD	S_1, S_2, D	Binary floating point addition	NC Series
58					

	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S_1						*							*		
S_2						*							*		
D													*		

16-bit instruction: None.
32-bit instruction: FADD continuous running type (7 STEP).
Flag: M2824 (zero flag), M2825 (borrow flag), M2826 (carry flag)

Notes on the use of operands:
Please see chapter 1 for details about the applicable range of each device.

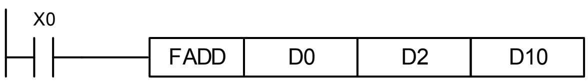
Instruction description:

S_1: Summand S_2: Addend D: Sum

1. This instruction adds S_1 and S_2 and saves the result in register specified by D. The addition is executed in binary floating point format.
2. If S1 or S2 is specified with constant K or F, this instruction will convert the constant to binary floating point for addition.
3. S_1 and S_2 can assign the register with the same No. In such case, if a instruction of continuous running type is executed, the register will conduct an addition for each scan cycle as long as the contact is ON.
4. When the absolute value of the addition result is greater than the maximum value of floating point, the carry flag M2826 turns ON.
5. When the absolute value of the addition result is smaller than the minimum value of floating point, the borrow flag M2825 turns ON.
6. If the addition result is 0, the zero flag M2824 is ON.

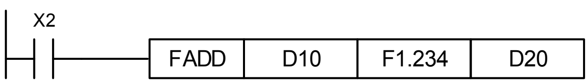
Example 1:

When X0 = ON, binary floating point (D1, D0) and (D3, D2) are added together and the sum is saved in (D11, D10).



Example 2:

When X2 = ON, binary floating point (D11, D10) is added to F1.234 (automatically converted to binary floating point), and the sum is saved in (D21, D20).



4

■ API-59 FSUB: Binary floating point subtraction

API	-	FSUB	S_1, S_2, D	Binary floating point subtraction	NC Series
59	D				

	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S_1					*								*		
S_2					*								*		
D													*		

16-bit instruction: None.
32-bit instruction: FSUB continuous running type (7 STEP).
Flag: M2824 (zero flag), M2825 (borrow flag), M2826 (carry flag)

Notes on the use of operands:
Please see chapter 1 for details about the applicable range of each device.

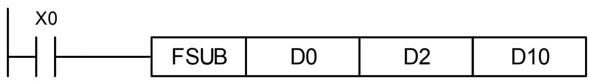
Instruction description:

S_1: Minuend S_2: Subtrahend D: Difference

1. This instruction subtracts S_2 from S_1, and saves the result in register specified by D. The subtraction is executed in binary floating point format.
2. If S1 or S2 is specified with constant K or F, this instruction will convert the constant to binary floating point for subtraction.
3. S_1 and S_2 can assign the register with the same No. In such case, if a instruction of continuous running type is executed, the register will conduct a subtraction for each scan cycle as long as the contact is ON.
4. When the absolute value of the subtraction result is greater than the maximum value of floating point, the carry flag M2826 turns ON.
5. When the absolute value of the subtraction result is smaller than the minimum value of floating point, the borrow flag M2825 turns ON.
6. If the subtraction results in 0, the zero flag M2824 turns ON.

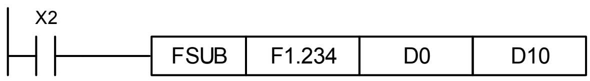
Example 1:

When X0 = ON, FSUB instruction subtracts the binary floating point (D3、D2) from (D1, D0) and the difference is saved in (D11, D10).



Example 2:

When X2 = ON, the binary floating point (D1, D0) is subtracted from F1.234 (automatically converted to binary floating point), and the remainder is saved in (D11, D10).



■ API-60 FMUL: Binary floating point multiplication

API		FMUL	S_1, S_2, D	Binary floating point multiplication	NC Series
60	D				

	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S_1					*								*		
S_2					*								*		
D													*		

16-bit instruction: None.
32-bit instruction: FMUL continuous running type (7 STEP).
Flag: M2824 (zero flag), M2825 (borrow flag), M2826 (carry flag)

Notes on the use of operands:
Please see chapter 1 for details about the applicable range of each device.

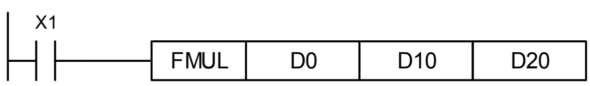
Instruction description:

S_1: Multiplicand S_2: Multiplier   D: Product

1. This instruction multiplies S_1 by S_2, and saves the result in D. The multiplication is executed in binary floating point format.
2. If S_1 or S_2 is specified with constant K or F, this instruction will convert the constant to binary floating point for multiplication.
3. S_1 and S_2 can assign the register with the same No. In such case, if a instruction of continuous running type is executed, the register will conduct a multiplication for each scan cycle as long as the contact is ON.
4. When the absolute value of the multiplication result is greater than the maximum value of floating point, the carry flag M2826 turns ON.
5. When the absolute value of the multiplication result is smaller than the minimum value of floating point, the borrow flag M2825 turns ON.
6. If the multiplication results in 0, the zero flag M2824 turns ON.

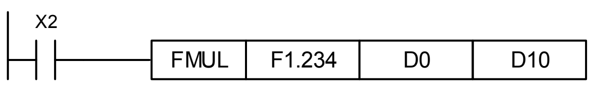
Example 1:

When X0 = ON, the binary floating point (D1, D0) is multiplied by (D11, D10) and the result is saved in (D21, D20).



Example 2:

When X2 = ON, the binary floating point (D1, D0) is multiplied by F1.234 (automatically converted to binary floating point), and the result is saved in (D11, D10).



■ API-61 FDIV: Binary floating point division

API	-	FDIV	S_1, S_2, D	Binary floating point division	NC Series
61					

	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S_1					*								*		
S_2					*								*		
D													*		

16-bit instruction: None.

32-bit instruction: FDIV continuous running type (7 STEP).

Flag: M2824 (zero flag), M2825 (borrow flag), M2826 (carry flag)

Notes on the use of operands:

Please see chapter 1 for details about the applicable range of each device.

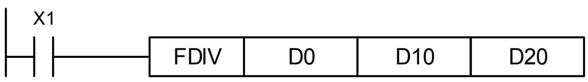
Instruction description:

S_1: Dividend S_2: Divisor D: Quotient and remainder

1. This instruction divides S_1 by S_2, and saves the result in register specified by D. The division is executed in binary floating point format.
2. If S1 or S2 is specified with constant K or F, this instruction will convert the constant to binary floating point for division.
3. When divisor S_2 is 0, the division will be regarded as operation error, and the instruction will not be executed. M1067 and M1068 will be ON along with error code H'0E19 recorded in D1067.
4. When the absolute value of the division result is greater than the maximum value of floating point, the carry flag M2826 turns ON.
5. When the absolute value of the division result is smaller than the minimum value of floating point, the borrow flag M2825 turns ON.
6. If the division results in 0, the zero flag M2824 turns ON.

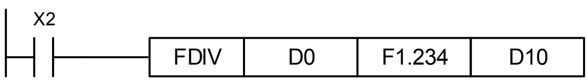
Example 1:

When X0 = ON, the binary floating point (D1, D0) is divided by (D11, D10) and the result is saved in (D21, D20).



Example 2:

When X2 = ON, the binary floating point (D1, D0) is divided by F1.234 (automatically converted to binary floating point), and the result is saved in (D11, D10).



■ API-62 FCMP: Binary floating point comparison

API	-	FCMP	S_1, S_2, D	Binary floating point comparison	NC Series
62					

	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S_1					*								*		
S_2					*								*		
D		*	*	*											

16-bit instruction: None.
32-bit instruction: FCMP continuous running type (7 STEP).
Flag: M2824 (zero flag), M2825 (borrow flag), M2826 (carry flag)

Notes on the use of operands:
Please see chapter 1 for details about the applicable range of each device.
Operand D occupies consecutive 3 points.

Instruction description:

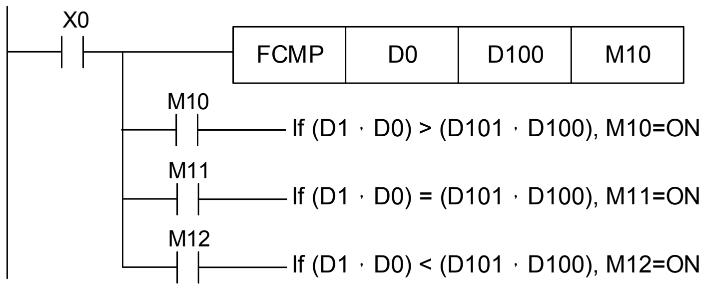
S_1: Binary floating point comparing value 1 S_2: Binary floating point comparing value 2

D: Comparing result, occupying consecutive 3 points.

This instruction compares S_1 and S_2, and the comparing result (>, =, <) is placed in register specified by D. If S_1 or S_2 specifies constant K or F, the instruction will convert the constant to binary floating point for comparison.

Example:

1. If the assigned device is M10, then M10 ~ M12 will be used.
2. When X0 = ON, execute FCMP instruction and one of M10 ~ M12 will turn ON. When X0= OFF, FCMP instruction will not be executed, M10 ~ M12 will remain the state before X0 goes OFF.
3. To have ≧, ≦, and ≠ statements, use logical combination for M10~M12.
4. RST and ZRST can be used to clear the comparing result.



■ API-63 FINT: Binary floating point convert to BIN integer

API	-	FINT	S, D	Binary floating point convert to BIN integer	NC Series
63					

	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S						*							*		
D													*		

16-bit instruction: None.
32-bit instruction: FINT continuous running type (5 STEP).
Flag: M2824 (zero flag), M2825 (borrow flag), M2826 (carry flag)

Notes on the use of operands:
Please see chapter 1 for details about the applicable range of each device.
Operand D occupies consecutive 2 points.

Instruction description:

S: Source device D: Conversion result

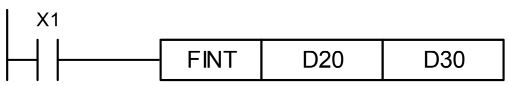
The value contained in S is converted from binary floating point to BIN integer and the result is saved in D. The floating points of the integer are discarded. FINT instruction is the opposite operation of API 64 FDOT instruction. If the conversion results in 0, the zero flag (M2824) turns ON. If the result has floating points that were discarded in the conversion, the borrow flag M2825 turns ON. If the result exceeds following range (overflow), the carry flag M2826 turns ON.

16-bit instruction: -32,768~32,767

32-bit instruction: -2,147,483,648~2,147,483,647

Example:

When X1 = ON, the binary floating point (D21, D20) is converted to BIN integer with the result saved in (D31, D30). The floating points in the result are discarded.



4

■ API-64 FDOT: BIN integer convert to binary floating point

API	-	FDOT		S, D			BIN integer convert to binary floating point			NC Series				
64														
	Bit device				Word device									
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S						∗							∗		
D													∗		

16-bit instruction: None.
32-bit instruction: FDOT continuous running type (5 STEP).
Flag: M2824 (zero flag), M2825 (borrow flag), M2826 (carry flag), M1081 (instruction function switch)

Notes on the use of operands:
Please see chapter 1 for details about the applicable range of each device.
Operand D occupies consecutive 2 points.

Instruction description:

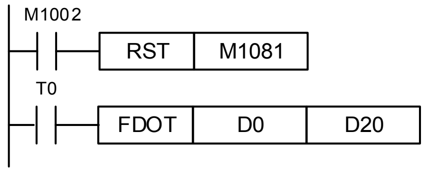
S: Source device D: Conversion result

1. When M1081 = OFF, BIN integer is converted to binary floating point. Here, the source device S in 16-bit instruction FDOT occupies 1 register, and device D stored with the conversion result occupies 2 registers.
   a. If the conversion result is greater than the maximum floating point, carry flag M2826 turns ON.
   b. If the conversion result is smaller than the minimum floating point, borrow flag M2825 turns ON.
   c. If the conversion results in 0, the borrow flag M2824 turns ON.
2. When M1081 = ON, the binary floating point is converted to BIN integer (discarding the decimal points). Here, the source device S in 16-bit instruction FLT occupies 2 register, and device D stored with the conversation result occupies 1 register. The operations are the same with INT instruction.
   a. When the conversion result exceeds the BIN integer range of D (16-bit: -32,768 ~ 32,767 and 32-bit: -2,147,483,648 ~ 2,147,483,647), D will save the maximum value or the minimum value, and carry flag M2826 will turn ON.
   b. If the result has decimal points that are discarded in the conversion, borrow flag M2825 will turn ON.
   c. If the value in S is 0, zero flag M1020 will turn ON.
   d. After the conversion, D saves the 16-bit result.

Example 1:

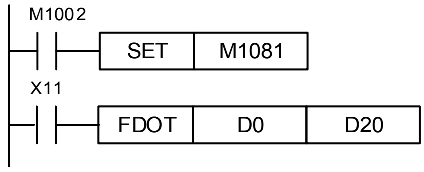
1. When M1081 = OFF, BIN integer is converted to binary floating point.
2. When X11 = ON, the instruction converts D1, D0 (composed of BIN integer) to D21, D20 (composed of binary floating point).

3. If 32-bit register D0(D1) = K100,000, X11 goes ON and the 32-bit value of the converted floating point is H4735000. And the result is saved in 32-bit register D20(D21).



Example 2:

1. When 1081 = ON, binary floating point is converted to BIN integer (discarding the decimal points.)
2. When X11 = ON, the instruction converts D1, D0 (composed of BIN integer) to D21, D20 (composed of binary floating point). If D0 (D1) = H47C35000, the value of the converted floating point is 100,000. And the result is saved in 32-bit register D20 (D21).



■ API-65 FRAD: Convert value in degree to radian

API 65	-	FRAD	S, D	Convert value in degree to radian	NC Series

	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S						*							*		
D													*		

16-bit instruction: None.
32-bit instruction: FRAD continuous running type (5 STEP).
Flag: M2824 (zero flag), M2825 (borrow flag), M2826 (carry flag)

Notes on the use of operands:
Please see chapter 1 for details about the applicable range of each device.

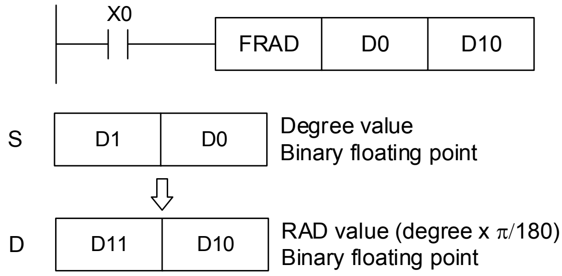
Instruction description:

S: Source device (degree) D: Conversion result (radian)

1. This instruction converts degree to radian via the equation: radian = degree × (π/180)
2. When the absolute value of the conversion result is greater than the maximum value of floating point, the carry flag M2826 turns ON.
3. When the absolute value of the conversion result is smaller than the minimum value of floating point, the borrow flag M2825 turns ON.
4. If the conversion results in 0, the zero flag M2824 turns ON.

Example:

When X0 = ON, the degree value in the format of binary floating point contained in (D1, D0) is converted to radian value in the same format, which is then saved in (D11, D10).



■ API-66 FDEG: Convert value in radian to degree

API	-	FDEG			S, D			Convert value in radian to degree		NC Series					
66															
	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
S						*							*		
D													*		

16-bit instruction: None.
32-bit instruction: FDEG continuous running type (5 STEP).
Flag: M2824 (zero flag), M2825 (borrow flag), M2826 (carry flag)

Notes on the use of operands:
Please see chapter 1 for details about the applicable range of each device.
This instruction is valid for 32-bit instruction FDEG only.

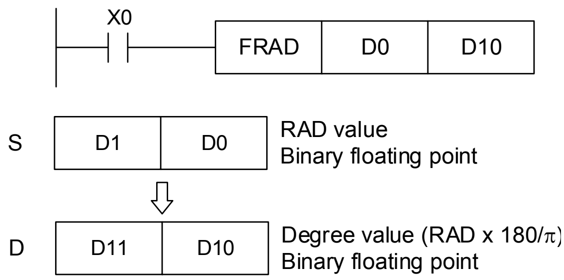
Instruction description:

S: Source device (radian) D: Conversion result (degree)

1. This instruction converts radian to degree via the equation: degree = radian × (180/π)
2. When the absolute value of the conversion result is greater than the maximum value of floating point, the carry flag M2826 turns ON.
3. When the absolute value of the conversion result is smaller than the minimum value of floating point, the borrow flag M2825 turns ON.
4. If the conversion results in 0, the zero flag M2824 turns ON.

Example:

When X0 = ON, the radian value in the format of binary floating point contained in (D1, D0) is converted to degree value in the same format, which is then saved in (D11, D10).



MLC Special M, D Commands and Functions 5

This chapter provides description about all the special M and D commands in the NC system, including the definitions, categories and functions.

5

5.1 Definitions of the special M, D in NC series

The MLC (Motion Logic Control) and NC systems are two independent systems. MLC system performs knob and button controls, mechanical operations, and other electric logic controls, while NC system manages system and servo axis related functions. The MLC special M and D serve as the I/O interface between these two systems for data exchange and signal transmission. Output mentioned in this chapter refers to the signals sent to the NC system by MLC special M and D. Input refers to the signals sent to MLC special M and D by the NC system. The M letter prefixed commands are in bit format referring to signal 0 (OFF) or 1 (ON). The D prefixed ones are in word format referring to numerical values like 1000. MLC special M and D codes are all expressed in the form of M- and D- suffixed with four digits. Data exchanges between the two systems are divided into following four groups:

1: MLC bit output from MLC to NC (special M, bit output)
2: MLC bit input from NC to MLC (special M, bit input)
3: MLC word output from MLC to NC (special D, word output)
4: MLC word input from NC to MLC (special D, word input)

5.2 Description of the special M in NC series

This section lists all the special M in NC series along with the definitions and categories.

- **List of Special M**

Function name	Special M code	Description	Device type
HMI output 1	M1024	Pass the special M state to system # variable, paring with variable #1801.	R/W
HMI output 2	M1025	Pass the special M state to system # variable, paring with variable #1802.	R/W
HMI output 3	M1026	Pass the special M state to system # variable, paring with variable #1803.	R/W
HMI output 4	M1027	Pass the special M state to system # variable, paring with variable #1804.	R/W
HMI output 5	M1028	Pass the special M state to system # variable, paring with variable #1805.	R/W
HMI output 6	M1029	Pass the special M state to system # variable, paring with variable #1806.	R/W
HMI output 7	M1030	Pass the special M state to system # variable, paring with variable #1807.	R/W
HMI output 8	M1031	Pass the special M state to system # variable, paring with variable #1808.	R/W

Function name	Special M code	Description	Device type
HMI output 9	M1032	Pass the special M state to system # variable, paring with variable #1809.	R/W
HMI output 10	M1033	Pass the special M state to system # variable, paring with variable #1810.	R/W
HMI output 11	M1034	Pass the special M state to system # variable, paring with variable #1811.	R/W
HMI output 12	M1035	Pass the special M state to system # variable, paring with variable #1812.	R/W
HMI output 13	M1036	Pass the special M state to system # variable, paring with variable #1813.	R/W
HMI output 14	M1037	Pass the special M state to system # variable, paring with variable #1814.	R/W
HMI output 15	M1038	Pass the special M state to system # variable, paring with variable #1815.	R/W
HMI output 16	M1039	Pass the special M state to system # variable, paring with variable #1816.	R/W
HMI output 17	M1040	Pass the special M state to system # variable, paring with variable #1817.	R/W
HMI output 18	M1041	Pass the special M state to system # variable, paring with variable #1818.	R/W
HMI output 19	M1042	Pass the special M state to system # variable, paring with variable #1819.	R/W
HMI output 20	M1043	Pass the special M state to system # variable, paring with variable #1820.	R/W
HMI output 21	M1044	Pass the special M state to system # variable, paring with variable #1821.	R/W
HMI output 22	M1045	Pass the special M state to system # variable, paring with variable #1822.	R/W
HMI output 23	M1046	Pass the special M state to system # variable, paring with variable #1823.	R/W
HMI output 24	M1047	Pass the special M state to system # variable, paring with variable #1824.	R/W
HMI output 25	M1048	Pass the special M state to system # variable, paring with variable #1825.	R/W
HMI output 26	M1049	Pass the special M state to system # variable, paring with variable #1826.	R/W

Function name	Special M code	Description	Device type
HMI output 27	M1050	Pass the special M state to system # variable, paring with variable #1827.	R/W
HMI output 28	M1051	Pass the special M state to system # variable, paring with variable #1828.	R/W
HMI output 29	M1052	Pass the special M state to system # variable, paring with variable #1829.	R/W
HMI output 30	M1053	Pass the special M state to system # variable, paring with variable #1830.	R/W
HMI output 31	M1054	Pass the special M state to system # variable, paring with variable #1831.	R/W
HMI output 32	M1055	Pass the special M state to system # variable, paring with variable #1832.	R/W
System mode selection: 0. Auto execution (AUTO) 1. Edit (EDIT) 2. Manual input (MDI) 3 Hand wheel feeding (MPG) 4. Jog (JOG) 5 Fast feeding (RAPID) 6. Homing (HOME)	M1056 M1057 M1058 M1059	The NC system modes can be selected through M1056 ~ M 1059, which is represented by Bit 0 ~ Bit 3 in binary format. The binary number can be converted to decimals 0 ~ 6 referring to each system mode. For example, MPG mode is represented by decimal 3 (= binary number 0011) and its corresponding four bits in MLC are M1056 ~ M1059. Thus, the bit state of MPG mode is shown as below: M1056 = ON M1057 = ON M1058 = OFF M1059 = OFF	R/W
Single block execution	M1060	In auto mode, program stops after one block is executed.	R/W
Cycle Start	M1061	Instruct the system to start running (Cycle Start).	R/W
NC STOP	M1062	NC controller pauses after M1062 is triggered.	R/W
System STOP	M1063	The system stops operating.	R/W
System reset	M1064	When triggering M1076 or the Reset on the 1st panel, the system will start resetting and M1064 will be triggered.	R
Dummy execution	M1065	After M1065 is triggered, the movement speed F of G01 in auto mode will be set as the feed rate in register D1062.	R/W
Optional stop (M01 Pause)	M1066	Optional stop key. The controller pauses when M01 is executed in the program.	R/W

Function name	Special M code	Description	Device type
Single block skip (‘/’)	M1067	The program will skip the block with symbol '/' when this function is enabled.	R/W
Mechanical lock of each axis	M1068	Lock axis X, Y, and Z movement.	R/W
Lock axis Z	M1069	Lock axis Z from movement.	R/W
Relieve axis limit	M1070	The limit signal of each axis will be ignored when this function is active.	R/W
Lock M, S, T codes	M1071	Lock M, S, and T codes. The program will skip M, S, T codes in the execution.	R/W
DMCNET successfully connected	M1072	M1072 signal will be sent when the system has been successfully connected to DMCNET. This signal only confirms the success connection, which does not indicate the servo state (Servo ON or OFF).	R
G31 MLC Input contact	M1073	When G31 input is set to 0 in parameter 307, this special M can be used as G31 input signal.	R/W
Macro call initial preparation	M1074	The initial input of macro calling (only works in auto mode and with correct macro ID)	R/W
Macro call activation	M1075	Activate macro calling.	R/W
System reset	M1076	When M1076 is triggered, NC system will be reset. (MLC > NC)	R/W
MPG simulation	M1080	When executing the program, MPG can be used to control the speed of movement trails.	R/W
Trigger flag of synchronous control	M1088	Set this flag to ON to enable the system’s synchronous function.	R/W
The slave axis X follows the master axis	M1089	Set axis X as the following axis in the synchronous function.	R/W
The slave axis Y follows the master axis	M1090	Set axis Y as the following axis in the synchronous function.	R/W
The slave axis Z follows the master axis	M1091	Set axis Z as the following axis in the synchronous function.	R/W
The slave axis A follows the master axis	M1092	Set axis A as the following axis in the synchronous function.	R/W
The slave axis B follows the master axis	M1093	Set axis B as the following axis in the synchronous function.	R/W
The slave axis C follows the master axis	M1094	Set axis C as the following axis in the synchronous function.	R/W

Function name	Special M code	Description	Device type
Trigger flag of transfer command controls	M1098	Set this flag to ON to enable the transfer command.	R/W
Axis X receives command from master axis	M1099	Set axis X to receive the command in the transfer command function.	R/W
Axis Y receives command from master axis	M1100	Set axis Y to receive the command in the transfer command function.	R/W
Axis Z receives command from master axis	M1101	Set axis Z to receive the command in the transfer command function.	R/W
Axis A receives command from master axis	M1102	Set axis A to receive the command in the transfer command function.	R/W
Axis B receives command from master axis	M1103	Set axis B to receive the command in the transfer command function.	R/W
Axis C receives command from master axis	M1104	Set axis C to receive the command in the transfer command function.	R/W
Panel MPG pulse +	M1118	This is the trigger signal for forward movement when using the keys on the secondary control panel for MPG function. See D1040 for reference.	R/W
Panel MPG pulse -	M1119	This is the trigger signal for backward movement when using the keys on the secondary control panel for MPG function. See D1040 for reference.	R/W
Spindle moves forward	M1120	The flag for spindle moving forward.	R/W
Spindle moves backward	M1121	The flag for spindle moving backward.	R/W
Spindle gear selection	M1122 M1123	The selection of spindle gear ratio is presented by the combination of M1122 (Bit 0) and M1123 (Bit 1), and the bit range is 0 ~ 3 representing the four gear ratio (parameter P422 ~ P429). For example: MPG mode is represented by decimal 3 (= binary number 0011) and its corresponding four bits in MLC are M1056 ~ M1123. Thus, the bit state of MPG mode is shown as below: M1122 = ON M1123 = ON	R/W
Spindle positioning control	M1124	The flag for spindle positioning control.	R/W
Spindle returns from tapping	M1125	The flag for spindle returning from tapping.	R/W

5

5

Function name	Special M code	Description	Device type
MST Code executed flag	M1152	When M1152 is triggered, NC system will be informed that M, S or T codes have completed their execution.	R/W
Tool magazine 1 moves forward	M1168	Tool magazine 1 moves forward. When M1168 is triggered, the standby tool pot (D1373) adds 1 to its value.	R/W
Tool magazine 1 moves backward	M1169	Tool magazine 1 moves backward. When M1169 is triggered, the standby tool pot (D1373) subtracts 1 from its value.	R/W
Tool 1 exchange	M1170	Exchange tool data in tool magazine 1.	R/W
Tool magazine 1 resets	M1171	When M1171 is triggered, the tool No. data in tool magazine 1 will be reset (work with M code in auto mode).	R/W
Tool magazine 2 moves forward	M1172	Tool magazine 2 moves forward. When M1172 is triggered, the standby tool pot (D1377) adds 1 to its value.	R/W
Tool magazine 2 moves backward	M1173	Tool magazine 2 moves backward. When M1173 is triggered, the standby tool pot (D1377) subtracts 1 from its value.	R/W
Tool 2 exchange	M1174	Exchange tool data in tool magazine 2.	R/W
Tool magazine 2 resets	M1175	When M1175 is triggered, the tool No. data in tool magazine 2 will be reset (work with M code in auto mode).	R/W
Trigger axis X movement (MLC axis)	M1184	The flag triggers MLC control on axis X.	R/W
Trigger axis Y movement (MLC axis)	M1185	The flag triggers MLC control on axis Y.	R/W
Trigger axis Z movement (MLC axis)	M1186	The flag triggers MLC control on axis Z.	R/W
Trigger axis A movement (MLC axis)	M1187	The flag triggers MLC control on axis A.	R/W
Trigger axis B movement (MLC axis)	M1188	The flag triggers MLC control on axis B.	R/W
Trigger axis C movement (MLC axis)	M1189	The flag triggers MLC control on axis C.	R/W
Trigger spindle movement (MLC axis)	M1193	The flag triggers MLC control on spindle.	R/W
Axis X forward jog control	M1216	The flag triggers axis X to jog in forward direction	R/W

Function name	Special M code	Description	Device type
Axis Y forward jog control	M1217	The flag triggers axis Y to jog in forward direction.	R/W
Axis Z forward jog control	M1218	The flag triggers axis Z to jog in forward direction.	R/W
Axis A forward jog control	M1219	The flag triggers axis A to jog in forward direction.	R/W
Axis B forward jog control	M1220	The flag triggers axis B to jog in forward direction.	R/W
Axis C forward jog control	M1221	The flag triggers axis C to jog in forward direction.	R/W
Axis X backward jog control	M1226	The flag triggers axis X to jog in backward direction.	R/W
Axis Y backward jog control	M1227	The flag triggers axis Y to jog in backward direction.	R/W
Axis Z backward jog control	M1228	The flag triggers axis Z to jog in backward direction.	R/W
Axis A backward jog control	M1229	The flag triggers axis A to jog in backward direction.	R/W
Axis B backward jog control	M1230	The flag triggers axis B to jog in backward direction.	R/W
Axis C backward jog control	M1231	The flag triggers axis C to jog in backward direction.	R/W
Axis X homing control	M1236	The flag triggers axis X to do homing.	R/W
Axis Y homing control	M1237	The flag triggers axis Y to do homing.	R/W
Axis Z homing control	M1238	The flag triggers axis Z to do homing.	R/W
Axis A homing control	M1239	The flag triggers axis A to do homing.	R/W
Axis B homing control	M1240	The flag triggers axis B to do homing.	R/W
Axis C homing control	M1241	The flag triggers axis C to do homing.	R/W
Relieve the 1st software limit of Axis X	M1248	Trigger flag of removing the 1st software limit on axis X.	R/W
Relieve the 1st software limit of Axis Y	M1249	Trigger flag of removing the 1st software limit on axis Y.	R/W
Relieve the 1st software limit of Axis Z	M1250	Trigger flag of removing the 1st software limit on axis Z.	R/W
Relieve the 1st software limit of Axis A	M1251	Trigger flag of removing the 1st software limit on axis A.	R/W
Relieve the 1st software limit of Axis B	M1252	Trigger flag of removing the 1st software limit on axis B.	R/W
Relieve the 1st software limit of Axis C	M1253	Trigger flag of removing the 1st software limit on axis C.	R/W
Lock axis X	M1257	Trigger flag of locking axis X movement.	R/W
Lock axis Y	M1258	Trigger flag of locking axis Y movement.	R/W

Function name	Special M code	Description	Device type
Lock axis Z	M1259	Trigger flag of locking axis Z movement.	R/W
Lock axis A	M1260	Trigger flag of locking axis A movement.	R/W
Lock axis B	M1261	Trigger flag of locking axis B movement.	R/W
Lock axis C	M1262	Trigger flag of locking axis C movement.	R/W
Axis X Servo Off	M1266	The flag triggers axis X to switch to Servo Off state.	R/W
Axis Y Servo Off	M1267	The flag triggers axis Y to switch to Servo Off state.	R/W
Axis Z Servo Off	M1268	The flag triggers axis Z to switch to Servo Off state.	R/W
Axis A Servo Off	M1269	The flag triggers axis A to switch to Servo Off state.	R/W
Axis B Servo Off	M1270	The flag triggers axis B to switch to Servo Off state.	R/W
Axis C Servo Off	M1271	The flag triggers axis C to switch to Servo Off state.	R/W
HMI input 1	M2080	Change the special M state via system # variable, paring with variable #1864.	R
HMI input 2	M2081	Change the special M state via system # variable, paring with variable #1865.	R
HMI input 3	M2082	Change the special M state via system # variable, paring with variable #1866.	R
HMI input 4	M2083	Change the special M state via system # variable, paring with variable #1867.	R
HMI input 5	M2084	Change the special M state via system # variable, paring with variable #1868.	R
HMI input 6	M2085	Change the special M state via system # variable, paring with variable #1869.	R
HMI input 7	M2086	Change the special M state via system # variable, paring with variable #1870.	R
HMI input 8	M2087	Change the special M state via system # variable, paring with variable #1871.	R
HMI input 9	M2088	Change the special M state via system # variable, paring with variable #1872.	R
HMI input 10	M2089	Change the special M state via system # variable, paring with variable #1873.	R
HMI input 11	M2090	Change the special M state via system # variable, paring with variable #1874.	R
HMI input 12	M2091	Change the special M state via system # variable, paring with variable #1875.	R
HMI input 13	M2092	Change the special M state via system # variable, paring with variable #1876.	R

Function name	Special M code	Description	Device type
HMI input 14	M2093	Change the special M state via system # variable, paring with variable #1877.	R
HMI input 15	M2094	Change the special M state via system # variable, paring with variable #1878.	R
HMI input 16	M2095	Change the special M state via system # variable, paring with variable #1879.	R
HMI input 17	M2096	Change the special M state via system # variable, paring with variable #1880.	R
HMI input 18	M2097	Change the special M state via system # variable, paring with variable #1881.	R
HMI input 19	M2098	Change the special M state via system # variable, paring with variable #1882.	R
HMI input 20	M2099	Change the special M state via system # variable, paring with variable #1883.	R
HMI input 21	M2100	Change the special M state via system # variable, paring with variable #1884.	R
HMI input 22	M2101	Change the special M state via system # variable, paring with variable #1885.	R
HMI input 23	M2102	Change the special M state via system # variable, paring with variable #1886.	R
HMI input 24	M2103	Change the special M state via system # variable, paring with variable #1887.	R
HMI input 25	M2104	Change the special M state via system # variable, paring with variable #1888.	R
HMI input 26	M2105	Change the special M state via system # variable, paring with variable #1889.	R
HMI input 27	M2106	Change the special M state via system # variable, paring with variable #1890.	R
HMI input 28	M2107	Change the special M state via system # variable, paring with variable #1891.	R
HMI input 29	M2108	Change the special M state via system # variable, paring with variable #1892.	R
HMI input 30	M2109	Change the special M state via system # variable, paring with variable #1893.	R
HMI input 31	M2110	Change the special M state via system # variable, paring with variable #1894.	R

Function name	Special M code	Description	Device type
HMI input 32	M2111	Change the special M state via system # variable, paring with variable #1895.	R
Machine started and system is ready	M2112	NC system is ready.	R
System alarm message	M2113	Alarm occurs in the NC system.	R
System emergency stop	M2114	System stops immediately after the EMG key is pressed. Then, this signal will be sent.	R
Servo enabled	M2115	This signal is sent when the servo is ready.	R
HSI1	M2142	High speed input point 1 (G31 skip signal input)	R
HSI2	M2143	High speed input point 2 (G31 skip signal input)	R
Port 1 axis positive hardware limit	M2144	This signal is sent when positive hardware limit of Port 1 axis is triggered.	R
Port 1 axis negative hardware limit	M2145	This signal is sent when negative hardware limit of Port 1 axis is triggered.	R
Port 1 axis home signal	M2146	This signal is sent when homing of Port 1 axis is triggered.	R
Port 2 axis positive hardware limit	M2148	This signal is sent when positive hardware limit of Port 2 axis is triggered.	R
Port 2 axis negative hardware limit	M2149	This signal is sent when negative hardware limit of Port 2 axis is triggered.	R
Port 2 axis home signal	M2150	This signal is sent when homing signal of Port 2 axis is triggered.	R
Port 3 axis positive hardware limit	M2152	This signal is sent when positive hardware limit of Port 3 axis is triggered.	R
Port 3 axis negative hardware limit	M2153	This signal is sent when negative hardware limit of Port 3 axis is triggered.	R
Port 3 axis home signal	M2154	This signal is sent when homing signal of Port 3 axis is triggered.	R
Port 4 axis positive hardware limit	M2156	This signal is sent when positive hardware limit of Port 4 axis is triggered.	R
Port 4 axis negative hardware limit	M2157	This signal is sent when negative hardware limit of Port 4 axis is triggered.	R
Port 4 axis home signal	M2158	This signal is sent when homing signal of Port 4 axis is triggered.	R
Port 5 positive hardware limit	M2160	This signal is sent when positive hardware limit of Port 5 axis is triggered.	R

Function name	Special M code	Description	Device type
Port 5 axis negative hardware limit	M2161	This signal is sent when negative hardware limit of Port 5 axis is triggered.	R
Port 5 axis home signal	M2162	This signal is sent when homing signal of Port 5 axis is triggered is triggered.	R
Port 6 positive hardware limit	M2164	This signal is sent when positive hardware limit of Port 6 axis is triggered.	R
Port 6 axis negative hardware limit	M2165	This signal is sent when negative hardware limit pf Port 6 axis is triggered.	R
Port 6 axis home signal	M2166	This signal is sent when homing signal of Port 6 axis is triggered.	R
MST Code executed flag	M2208	When M codes are executed in the program, NC system will set this signal to ON. It will be set to OFF when MST Code completed flag (M1152) is triggered. The M codes mentioned here do not include M00, M01, M02, M30, M98, M99 or the M code specified as macro.	R
S Code execution flag	M2209	When S codes are executed in the program, NC system will set this signal to ON. It will be set to OFF when MST Code completed flag (M1152) is triggered. This function will not work when the S code is specified as macro.	R
T Code execution flag	M2210	When the T codes are executed in the program, NC system will set this signal to ON. It will be set to OFF when MST Code completed flag (M1152) is triggered. This flag will not be triggered when the T code has been used for macro calling. Flag M2210 is related to the station ID in tool magazine. The flag can be triggered only when the value of T code is within the range of station ID specified in the tool magazine.	R
Reset tool data in tool magazine 1 completed	M2212	This signal will be sent when the tool magazine is reset via M1171. (Required to work with M code in auto mode).	R
Reset tool data in tool magazine 2 completed	M2213	This signal will be sent when the tool magazine is reset via M1175. (Required to work with M code in auto mode).	R
Macro call initialization completed	M2224	Macro call initial function completed.	R

Function name	Special M code	Description	Device type
Activating flag	M2225	Flag M2225 activates the execution of macro call.	R
Error flag	M2226	Flag M2226 indicates that error occurred in the macro calling.	R
Synchronous function in execution	M2227	NC system sends this signal when the synchronous function is in execution.	R
Transfer function in execution	M2228	NC system sends this signal when the transfer function is in execution.	R
System reset	M2229	This signal is sent when the system is reset.	R
Channel alarm message	M2240	Irregularity occurs in NC channel.	R
Auto execution (AUTO)	M2241	NC system sends this signal in AUTO mode.	R
Edit (EDIT)	M2242	NC system sends this signal in EDIT mode.	R
Manual input(MDI)	M2243	NC system sends this signal in MDI mode.	R
Hand wheel feeding (MPG)	M2244	NC system sends this signal in MPG mode.	R
Jog (JOG)	M2245	NC system sends this signal in JOG mode.	R
Fast feed (RAPID)	M2246	NC system sends this signal in RAPID mode.	R
Homing (HOME)	M2247	NC system sends this signal in HOME mode.	R
Single block execution	M2249	NC system sends this signal when the program stops after executing single block.	R
Cycle Start	M2250	NC system sends this signal when the program starts running.	R
Pause	M2251	NC system sends this signal when the system is paused.	R
M00 program stops	M2252	NC system sends this signal when M00 is executed.	R
M01 optional pause	M2253	NC system sends this signal when M01 is executed.	R
M02 program ends	M2254	NC system sends this signal when M02 is executed.	R
M30 program ends and returns	M2255	NC system sends this signal when M30 is executed.	R
Spindle reaches the target speed	M2256	This signal is sent when the spindle reaches the target speed.	R
Spindle reaches zero speed	M2257	This signal is sent when the spindle reaches zero speed.	R
Spindle positioning completed	M2258	This signal is sent when the spindle reaches the target position.	R
Spindle is in rigid tapping mode.	M2259	This signal is sent when the spindle executes rigid tapping.	R
Rigid tapping interruption	M2260	This signal is sent in rigid tapping interruption.	R
Program ends	M2271	This signal is sent when the machining program ends.	R

Function name	Special M code	Description	Device type
Axis X homing completed	M2272	This signal is sent when axis X completes homing.	R
Axis Y homing completed	M2273	This signal is sent when axis Y completes homing.	R
Axis Z homing completed	M2274	This signal is sent when axis Z completes homing.	R
Axis A homing completed	M2275	This signal is sent when axis A completes homing.	R
Axis B homing completed	M2276	This signal is sent when axis B completes homing.	R
Axis C homing completed	M2277	This signal is sent when axis C completes homing.	R
Spindle homing completed	M2281	This signal is sent when the spindle completes homing.	R
Axis X positioned at the second reference point	M2286	This signal is sent when axis X reaches the second reference point.	R
Axis Y positioned at the second reference point	M2287	This signal is sent when axis Y reaches the second reference point.	R
Axis Z positioned at the second reference point	M2288	This signal is sent when axis Z reaches the second reference point.	R
Axis A positioned at the second reference point	M2289	This signal is sent when axis A reaches the second reference point.	R
Axis B positioned at the second reference point	M2290	This signal is sent when axis B reaches the second reference point.	R
Axis C positioned at the second reference point	M2291	This signal is sent when axis C reaches the second reference point.	R
G00 teach triggered	M2292	This signal is sent when using G00 in teach mode.	R
G01 teach triggered	M2293	This signal is sent when using G01 in teach mode.	R
G00 teach record completed	M2294	This signal is sent when G00 is used in teach mode and the teaching path is recorded.	R
G01 teach record completed	M2295	This signal is sent when G01 is used in teach mode and the teaching path is recorded.	R
Axis X positioning completed (MLC axis)	M2304	This signal is sent when axis X reaches the reference point and axis X is controlled by MLC.	R
Axis Y positioning completed (MLC axis)	M2305	This signal is sent when axis Y reaches the reference point and axis Y is controlled by MLC.	R
Axis Z positioning completed (MLC axis)	M2306	This signal is sent when axis Z reaches the reference point and axis Z is controlled by MLC.	R
Axis A positioning completed (MLC axis)	M2307	This signal is sent when axis A reaches the reference point and axis A is controlled by MLC.	R
Axis B positioning completed (MLC axis)	M2308	This signal is sent when axis B reaches the reference point and axis B is controlled by MLC.	R

5

Function name	Special M code	Description	Device type
Axis C positioning completed (MLC axis)	M2309	This signal is sent when axis C reaches the reference point and axis C is controlled by MLC.	R
Axis X is moving	M2320	This signal is sent when axis X is moving in any mode.	R
Axis Y is moving	M2321	This signal is sent when axis Y is moving in any mode.	R
Axis Z is moving	M2322	This signal is sent when axis Z is moving in any mode.	R
Axis A is moving	M2323	This signal is sent when axis A is moving in any mode.	R
Axis B is moving	M2324	This signal is sent when axis B is moving in any mode.	R
Axis C is moving	M2325	This signal is sent when axis C is moving in any mode.	R
IX00 interrupt input	M2880	IX00 interrupt input (On Board X0). (1: Enable; 0: Disable)	R/W
IX01 interrupt input	M2881	IX01 interrupt input (On Board X1).	R/W
IX02 interrupt input	M2882	IX02 interrupt input (On Board X2).	R/W
IX03 interrupt input	M2883	IX03 interrupt input (On Board X3).	R/W
IX04 interrupt input	M2884	IX04 interrupt input (On Board X4).	R/W
IX05 interrupt input	M2885	IX05 interrupt input (On Board X5).	R/W
IX06 interrupt input	M2886	IX06 interrupt input (On Board X6).	R/W
IX07 interrupt input	M2887	IX07 interrupt input (On Board X7).	R/W
IC00 interrupt input	M2888	IC00 interrupt input (Hardware Counter 0).	R/W
IC01 interrupt input	M2889	IC01 interrupt input (Hardware Counter 1).	R/W
IR00 interrupt input	M2896	IR00 interrupt input (X256 of Remote IO module)	R/W
IR01 interrupt input	M2897	IR01 interrupt input (X257 of Remote IO module)	R/W
IR02 interrupt input	M2898	IR02 interrupt input (X258 of Remote IO module).	R/W
IR03 interrupt input	M2899	IR03 interrupt input (X259 of Remote IO module)	R/W
IR04 interrupt input	M2900	IR04 interrupt input (X260 of Remote IO module)	R/W
IR05 interrupt input	M2901	IR05 interrupt input (X261 of Remote IO module)	R/W
IR06 interrupt input	M2902	IR06 interrupt input (X262 of Remote IO module)	R/W
IR07 interrupt input	M2903	IR07 interrupt input (X263 of Remote IO module)	R/W
IR08 interrupt input	M2904	IR08 interrupt input (X264 of Remote IO module)	R/W
IR09 interrupt input	M2905	IR09 interrupt input (X265 of Remote IO module)	R/W
IR10 interrupt input	M2906	IR10 interrupt input (X266 of Remote IO module)	R/W
IR11 interrupt input	M2907	IR11 interrupt input (X267 of Remote IO module)	R/W
IR12 interrupt input	M2908	IR12 interrupt input (X268 of Remote IO module)	R/W
IR13 interrupt input	M2909	IR13 interrupt input (X269 of Remote IO module)	R/W
IR14 interrupt input	M2910	IR14 interrupt input (X270 of Remote IO module)	R/W
IR15 interrupt input	M2911	IR15 interrupt input (X271 of Remote IO module)	R/W
IR16 interrupt input	M2912	IR16 interrupt input (X272 of Remote IO module)	R/W
IR17 interrupt input	M2913	IR17 interrupt input (X273 of Remote IO module)	R/W

Function name	Special M code	Description	Device type
IR18 interrupt input	M2914	IR18 interrupt input (X274 of Remote IO module)	R/W
IR19 interrupt input	M2915	IR19 interrupt input (X275 of Remote IO module)	R/W
IR20 interrupt input	M2916	IR20 interrupt input (X276 of Remote IO module)	R/W
IR21 interrupt input	M2917	IR21 interrupt input (X277of Remote IO module)	R/W
IR22 interrupt input	M2918	IR22 interrupt input (X278 of Remote IO module)	R/W
IR23 interrupt input	M2919	IR23 interrupt input (X279 of Remote IO module)	R/W
IR24 interrupt input	M2920	IR24 interrupt input (X280 of Remote IO module)	R/W
IR25 interrupt input	M2921	IR25 interrupt input (X281 of Remote IO module)	R/W
IR26 interrupt input	M2922	IR26 interrupt input (X282 of Remote IO module)	R/W
IR27 interrupt input	M2923	IR27 interrupt input (X283 of Remote IO module)	R/W
IR28 interrupt input	M2924	IR28 interrupt input (X284 of Remote IO module)	R/W
IR29 interrupt input	M2925	IR29 interrupt input (X285 of Remote IO module)	R/W
IR30 interrupt input	M2926	IR30 interrupt input (X286 of Remote IO module)	R/W
IR31 interrupt input	M2927	IR31 interrupt input (X287 of Remote IO module)	R/W
Lock user permission	M2934	M2934 can be used to lock the user permission. To use this special M, please set parameter 10015 (methods of granting permission) to 1, which is to restrict the user permission.	R/W
Lock program editing	M2935	Prevent the program in controllers from being edited.	R/W

- **Special M for HMI output**

Variable #1801~#1832 can be used in a machining program to read the signal state of MLC “HMI output points.” Variable #1801 ~ #1832 are paired with MLC Interface output points M1024 ~ M1055 respectively. For example, #1801 is paired with M1024, and so forth, for total 32 pairs. If M1024 output is ON, the variable in the NC program #1801 will be 1, and this value will be 0 if M1024 output is OFF.

Global Bit (MLC > NC)

Function name	Special M code	Variable ID	Function name	Special M code	Variable ID
HMI output 1	M1024	#1801	HMI output 17	M1040	#1817
HMI output 2	M1025	#1802	HMI output 18	M1041	#1818
HMI output 3	M1026	#1803	HMI output 19	M1042	#1819
HMI output 4	M1027	#1804	HMI output 20	M1043	#1820
HMI output 5	M1028	#1805	HMI output 21	M1044	#1821
HMI output 6	M1029	#1806	HMI output 22	M1045	#1822
HMI output 7	M1030	#1807	HMI output 23	M1046	#1823
HMI output 8	M1031	#1808	HMI output 24	M1047	#1824
HMI output 9	M1032	#1809	HMI output 25	M1048	#1825

5

Function name	Special M code	Variable ID	Function name	Special M code	Variable ID
HMI output 10	M1033	#1810	HMI output 26	M1049	#1826
HMI output 11	M1034	#1811	HMI output 27	M1050	#1827
HMI output 12	M1035	#1812	HMI output 28	M1051	#1828
HMI output 13	M1036	#1813	HMI output 29	M1052	#1829
HMI output 14	M1037	#1814	HMI output 30	M1053	#1830
HMI output 15	M1038	#1815	HMI output 31	M1054	#1831
HMI output 16	M1039	#1816	HMI output 32	M1055	#1832

■ **Special M for NC axes**

When the special M signals in this section are triggered, NC system will be instructed to conduct axis actions. For example, the forward jog of axis X will be enabled when M1216 is set to ON. The table below lists the flag signals for the action controls of each NC axis:

Function name	Special M code	Function name	Special M code
Mechanical lock of each axis	M1068	Axis X Servo Off	M1266
Lock axis Z	M1069	Axis Y Servo Off	M1267
Relieve axis limit	M1070	Axis Z Servo Off	M1268
Trigger flag of synchronous control	M1088	Axis A Servo Off	M1269
The slave axis X follows the master axis	M1089	Axis B Servo Off	M1270
The slave axis Y follows the master axis	M1090	Axis C Servo Off	M1271
The slave axis Z follows the master axis	M1091	Port 1 positive hardware limit	M2144
The slave axis A follows the master axis	M1092	Port 1 axis negative hardware limit	M2145
The slave axis B follows the master axis	M1093	Port 1 axis home signal	M2146
The slave axis C follows the master axis	M1094	Port 2 axis positive hardware limit	M2148
Trigger flag of transfer command controls	M1098	Port 2 axis negative hardware limit	M2149
Axis X receives command from master axis	M1099	Port 2 axis home signal	M2150
Axis Y receives command from master axis	M1100	Port 3 axis positive hardware limit	M2152
Axis Z receives command from master axis	M1101	Port 3 axis negative hardware limit	M2153

Function name	Special M code	Function name	Special M code
Axis A receives command from master axis	M1102	Port 3 axis home signal	M2154
Axis B receives command from master axis	M1103	Port 4 axis positive hardware limit	M2156
Axis C receives command from master axis	M1104	Port 4 axis negative hardware limit	M2157
Axis X forward jog control	M1216	Port 4 axis home signal	M2158
Axis Y forward jog control	M1217	Port 5 positive hardware limit	M2160
Axis Z forward jog control	M1218	Port 5 axis negative hardware limit	M2161
Axis A forward jog control	M1219	Port 5 axis home signal	M2162
Axis B forward jog control	M1220	Port 6 positive hardware limit	M2164
Axis C forward jog control	M1221	Port 6 axis negative hardware limit	M2165
Axis X backward jog control	M1226	Port 6 axis home signal	M2166
Axis Y backward jog control	M1227	Axis X homing completed	M2272
Axis Z backward jog control	M1228	Axis Y homing completed	M2273
Axis A backward jog control	M1229	Axis Z homing completed	M2274
Axis B backward jog control	M1230	Axis A homing completed	M2275
Axis C backward jog control	M1231	Axis B homing completed	M2276
Axis X homing control	M1236	Axis C homing completed	M2277
Axis Y homing control	M1237	Axis X positioned at the second reference point	M2286
Axis Z homing control	M1238	Axis Y positioned at the second reference point	M2287
Axis A homing control	M1239	Axis Z positioned at the second reference point	M2288
Axis B homing control	M1240	Axis A positioned at the second reference point	M2289
Axis C homing control	M1241	Axis B positioned at the second reference point	M2290
Relieve the 1st software limit of Axis X	M1248	Axis C positioned at the second reference point	M2291
Relieve the 1st software limit of Axis Y	M1249	Axis X is moving	M2320
Relieve the 1st software limit of Axis Z	M1250	Axis Y is moving	M2321
Relieve the 1st software limit of Axis A	M1251	Axis Z is moving	M2322

5

Function name	Special M code	Function name	Special M code
Relieve the 1st software limit of Axis B	M1252	Axis A is moving	M2323
Relieve the 1st software limit of Axis C	M1253	Axis B is moving	M2324
Lock axis X	M1257	Axis C is moving	M2325
Lock axis Y	M1258	-	-
Lock axis Z	M1259	-	-
Lock axis A	M1260	-	-
Lock axis B	M1261	-	-
Lock axis C	M1262	-	-

- **Special M for spindle control**

Please refer to the following special M list for the controlling of spindle actions.

Function name	Special M code	Function name	Special M code
Spindle moves forward	M1120	Spindle reaches the target speed	M2256
Spindle moves backward	M1121	Spindle reaches zero speed	M2257
Spindle gear selection	M1122 M1123	Spindle positioning completed	M2258
Spindle positioning control	M1124	Spindle is in rigid tapping mode	M2259
Spindle returns from tapping	M1125	Spindle homing completed	M2281

- **Special M for HMI input**

Variable #1864~#1895 can be used in a NC program to read the signal state of MLC “HMI input points”. Variable 1864~#1895 are paired with MLC Interface output points M2080~M2111 respectively. For example, #1864 is paired with M2080, and so forth, for total 32 pairs. If #1864 = 1 in NC program, M2028 in MLC will be ON; and if # 1864 = 0, M2028 will be OFF.

Function name	Special M code	Variable ID	Function name	Special M code	Variable ID
HMI input 1	M2080	#1864	HMI input 17	M2096	#1880
HMI input 2	M2081	#1865	HMI input 18	M2097	#1881
HMI input 3	M2082	#1866	HMI input 19	M2098	#1882
HMI input 4	M2083	#1867	HMI input 20	M2099	#1883
HMI input 5	M2084	#1868	HMI input 21	M2100	#1884
HMI input 6	M2085	#1869	HMI input 22	M2101	#1885
HMI input 7	M2086	#1870	HMI input 23	M2102	#1886
HMI input 8	M2087	#1871	HMI input 24	M2103	#1887
HMI input 9	M2088	#1872	HMI input 25	M2104	#1888
HMI input 10	M2089	#1873	HMI input 26	M2105	#1889
HMI input 11	M2090	#1874	HMI input 27	M2106	#1890

Function name	Special M code	Variable ID	Function name	Special M code	Variable ID
HMI input 12	M2091	#1875	HMI input 28	M2107	#1891
HMI input 13	M2092	#1876	HMI input 29	M2108	#1892
HMI input 14	M2093	#1877	HMI input 30	M2109	#1893
HMI input 15	M2094	#1878	HMI input 31	M2110	#1894
HMI input 16	M2095	#1879	HMI input 32	M2111	#1895

■ **Special M for M, S, T codes**

When M, S, and T codes are executed in a program, NC system will send the corresponding special M to MLC. For example, when M03 is executed in a program, M2208 in MLC will be set to ON accordingly. Followings are the special M flags corresponding to M, S, T codes.

Function name	Special M code	Function name	Special M code
Lock M, S, T codes	M1071	S Code execution flag	M2209
MST Code executed flag	M1152	T Code execution flag	M2210
MST Code Execution flag	M2208		

■ **Special M for tool magazines**

Function name	Special M code	Function name	Special M code
Tool magazine 1 moves forward	M1168	Tool magazine 2 moves backward	M1173
Tool magazine 1 moves backward	M1169	Tool 2 exchange	M1174
Tool 1 exchange	M1170	Tool magazine 2 resets	M1175
Tool magazine 1 resets	M1171	Reset tool data in tool magazine 1 completed	M2212
Tool magazine 2 moves forward	M1172	Reset tool data in tool magazine 2 completed	M2213

■ **Special M for MLC axes**

Function name	Special M code	Function name	Special M code
Trigger axis X movement (MLC axis).	M1184	Axis X positioning completed (MLC axis)	M2304
Trigger axis Y movement (MLC axis).	M1185	Axis Y positioning completed (MLC axis)	M2305
Trigger axis Z movement (MLC axis).	M1186	Axis Z positioning completed (MLC axis)	M2306
Trigger axis A movement (MLC axis).	M1187	Axis A positioning completed (MLC axis)	M2307
Trigger axis B movement (MLC axis).	M1188	Axis B positioning completed (MLC axis)	M2308

5

Function name	Special M code	Function name	Special M code
Trigger axis C movement (MLC axis)	M1189	Axis C positioning completed (MLC axis)	M2309
Trigger spindle movement (MLC axis)	M1193	-	-

■ Special M for mode switching

Function name	Special M code	Function name	Special M code
System mode selection: 0. Auto execution (AUTO) 1. Edit (EDIT) 2. Manual input (MDI) 3 Hand wheel feeding (MPG) 4. Jog (JOG) 5 Fast feeding (RAPID) 6. Homing (HOME)	M1056 M1057 M1058 M1059	Hand wheel feeding (MPG)	M2244
Auto execution (AUTO)	M2241	Jog (JOG)	M2245
Edit (EDIT)	M2242	Fast feed (RAPID)	M2246
Manual input(MDI)	M2243	Homing (HOME)	M2247

■ Special M for machining actions

Function name	Special M code	Function name	Special M code
Single block execution	M1060	Cycle Start	M2250
Cycle Start	M1061	Pause	M2251
Dummy execution	M1065	M00 program stops	M2252
Optional stop (M01 Pause)	M1066	M01 optional pause	M2253
Single block skip (‘/’)	M1067	M02 program ends	M2254
MPG simulation	M1080	M30 program ends and returns	M2255
Single block execution	M2249	Program ends	M2271

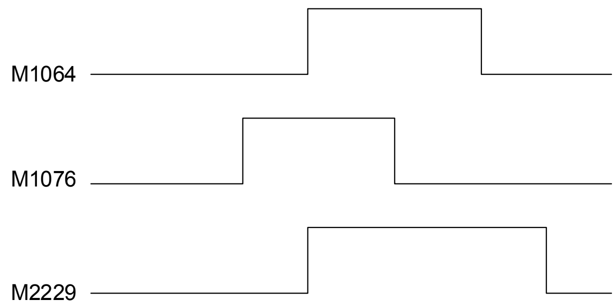
■ Special M for system actions

Function name	Special M code	Function name	Special M code
NC STOP	M1062	Synchronous function in execution	M2227
System STOP	M1063	Transfer function in execution	M2228
System reset	M1064	System reset completed	M2229
System reset	M1076	Channel alarm message	M2240
Machine started and system is ready	M2112	Lock user permission	M2934
System alarm message	M2113	Lock program editing	M2935

Function name	Special M code	Function name	Special M code
System emergency stop	M2114	-	-

Note: The action sequences of M1064, M1076, and M2229:

When system reset is enabled, M1076 will be set to ON instructing the system to reset; M1076 can be set to OFF when the reset has started. In the reset process, M1064 and M2229 are set to ON. When the reset has completed, M1064 will be set to OFF, and M2229 will be set to OFF at the last second.



- **Special M for DMCNET**

Function name	Special M code	Function name	Special M code
DMCNET successfully connected	M1072	Servo enabled	M2115

- **Special M for G31**

Function name	Special M code	Function name	Special M code
G31 MLC Input contact	M1073	HSI2	M2143
HSI1	M2142		

- **Special M for calling macro with one key**

Function name	Special M code	Function name	Special M code
Macro call initial preparation	M1074	Activating flag	M2225
Macro call activation	M1075	Error flag	M2226
Macro call initialization completed	M2224		

- **Special M for software panel**

Function name	Special M code	Function name	Special M code
Panel MPG pulse +	M1118	Panel MPG pulse -	M1119

- **Special M for teach mode**

Function name	Special M code	Function name	Special M code
G00 teach triggered	M2292	G00 teach record completed	M2294
G01 teach triggered	M2293	G01 teach record completed	M2295

5

■ **Special M for MLC input interruption**

Function name	Special M code	Function name	Special M code
IX00 interrupt input	M2880	IR11 interrupt input	M2907
IX01 interrupt input	M2881	IR12 interrupt input	M2908
IX02 interrupt input	M2882	IR13 interrupt input	M2909
IX03 interrupt input	M2883	IR14 interrupt input	M2910
IX04 interrupt input	M2884	IR15 interrupt input	M2911
IX05 interrupt input	M2885	IR16 interrupt input	M2912
IX06 interrupt input	M2886	IR17 interrupt input	M2913
IX07 interrupt input	M2887	IR18 interrupt input	M2914
IC00 interrupt input	M2888	IR19 interrupt input	M2915
IC01 interrupt input	M2889	IR20 interrupt input	M2916
IR00 interrupt input	M2896	IR21 interrupt input	M2917
IR01 interrupt input	M2897	IR22 interrupt input	M2918
IR02 interrupt input	M2898	IR23 interrupt input	M2919
IR03 interrupt input	M2899	IR24 interrupt input	M2920
IR04 interrupt input	M2900	IR25 interrupt input	M2921
IR05 interrupt input	M2901	IR26 interrupt input	M2922
IR06 interrupt input	M2902	IR27 interrupt input	M2923
IR07 interrupt input	M2903	IR28 interrupt input	M2924
IR08 interrupt input	M2904	IR29 interrupt input	M2925
IR09 interrupt input	M2905	IR30 interrupt input	M2926
IR10 interrupt input	M2906	IR31 interrupt input	M2927

5.3 Description of the special D in NC series

This section lists all the special M in NC series along with the definitions and categories.

5

- **List of Special D**

Function name	Special D code	Description	Device type
Number of the processed products	D1022	It can be set in the Process screen or by MLC input.	R/W
Number of the processing target	D1023	It can be set in the Process screen or by MLC input.	R/W
HMI output register 1	D1024	Pass the special D value to system # variable, paring with variable #1833.	R/W
HMI output register 2	D1025	Pass the special D value to system # variable, paring with variable #1834.	R/W
HMI output register 3	D1026	Pass the special D value to system # variable, paring with variable #1835.	R/W
HMI output register 4	D1027	Pass the special D value to system # variable, paring with variable #1836.	R/W
HMI output register 5	D1028	Pass the special D value to system # variable, paring with variable #1837.	R/W
HMI output register 6	D1029	Pass the special D value to system # variable, paring with variable #1838.	R/W
HMI output register 7	D1030	Pass the special D value to system # variable, paring with variable #1839.	R/W
HMI output register 8	D1031	Pass the special D value to system # variable, paring with variable #1840.	R/W
HMI output register 9	D1032	Pass the special D value to system # variable, paring with variable #1841.	R/W
HMI output register 10	D1033	Pass the special D value to system # variable, paring with variable #1842.	R/W
HMI output register 11	D1034	Pass the special D value to system # variable, paring with variable #1843.	R/W
HMI output register 12	D1035	Pass the special D value to system # variable, paring with variable #1844.	R/W
HMI output register 13	D1036	Pass the special D value to system # variable, paring with variable #1845.	R/W
HMI output register 14	D1037	Pass the special D value to system # variable, paring with variable #1846.	R/W

Function name	Special D code	Description	Device type
HMI output register 15	D1038	Pass the special D value to system # variable, paring with variable #1847.	R/W
HMI output register 16	D1039	Pass the special D value to system # variable, paring with variable #1848.	R/W
MPG operation mode No.	D1040	This function is to set the MPG operation mode. When D1040 is set to 0, it is for external MPG. If D1040 is set to 10, the MPG is controlled by the secondary control panel with M1118 and M1119 as the trigger signals.	R/W
MPG operation channel selection	D1041	D1041 helps to designate the MPG operation channel. The default value is 0.	R/W
Set MPG pulse magnification	D1042	D1042 is to set MPG pulse magnification, ×1, ×10, and ×100. And it usually works with the actual MPG. Multiply the minimum unit 0.001mm by the pulse magnification. E.g. 1×0.001 = 0.001mm/cnt.	R/W
Axis selection of MPG movement	D1043	You can select the axis to be moving via MPG operation. It is set that 0 = axis X, 1 = axis Y, and 2 = axis Z.	R/W
Adjustment of cutting feed rate	D1056	Set the adjustment ratio of the cutting feed rate (F) in NC programs. If F is set to 1000 and the current value of D1056 is 50, it means the actual command speed is F500 mm/min (1000 x 50%).	R/W
Speed adjustment of rapid movement	D1058	Set the adjustment ratio of G00 value (rapid movement). For example, if the speed of rapid movement is 6000, and D1058 is set to 50, it means the actual speed of G00 will be 3000 mm/min (= 6000 x 50%).	R/W
Spindle speed adjustment rate	D1060	Set the adjustment ratio of the S value specified in the program. For example, if S1000 is given in the program and D1060 is set to 30, it means the actual spindle speed is S300 r/min.	R/W
Set the speed of Jog and Dry run	D1062	Set movement speed F for dry run in JOG or AUTO mode. For example, set special D to 50 indicates F50 (mm/min) with a range of 0 ~ 65535 mm/min.	R/W
Axis X position command (MLC axis)	D1064	Set axis X as MLC axis and specify its target position. Unit: mm, inch	R/W
Axis Y position command (MLC axis)	D1066	Set axis Y as MLC axis and specify its target position. Unit: mm, inch	R/W

Function name	Special D code	Description	Device type
Axis Z position command (MLC axis)	D1068	Set axis Z as MLC axis and specify its target position. Unit: mm, inch	R/W
Axis A position command (MLC axis)	D1070	Set axis A as MLC axis and specify its target position. Unit: mm, inch	R/W
Axis B position command (MLC axis)	D1072	Set axis B as MLC axis and specify its target position. Unit: mm, inch	R/W
Axis C position command (MLC axis)	D1074	Set axis C as MLC axis and specify its target position. Unit: mm, inch	R/W
Axis X speed (MLC axis)	D1082	Set axis X as MLC axis and specify its moving speed. Unit: mm, inch/min.	R/W
Axis Y speed (MLC axis)	D1084	Set axis Y as MLC axis and specify its moving speed. Unit: mm, inch/min.	R/W
Axis Z speed (MLC axis)	D1086	Set axis Z as MLC axis and specify its moving speed. Unit: mm, inch/min.	R/W
Axis A speed (MLC axis)	D1088	Set axis A as MLC axis and specify its moving speed. Unit: mm, inch/min.	R/W
Axis B speed (MLC axis)	D1090	Set axis B as MLC axis and specify its moving speed. Unit: mm, inch/min.	R/W
Axis C speed (MLC axis)	D1092	Set axis C as MLC axis and specify its moving speed. Unit: mm, inch/min.	R/W
Spindle speed (MLC axis)	D1100	Set the spindle as MLC axis and specify its moving speed. Unit: mm, inch/min.	R/W
Calling macro file name	D1111	Specify the call macro file name as O9xxx. For example, D1111 writes K9100. When executing D1111, system will call macro named O9100.	R/W
HMI input register 1	D1336	Set the special D value via system # variable, paring with variable #1896.	R
HMI input register 2	D1337	Set the special D value via system # variable, paring with variable #1897.	R
HMI input register 3	D1338	Set the special D value via system # variable, paring with variable #1898.	R
HMI input register 4	D1339	Set the special D value via system # variable, paring with variable #1899.	R
HMI input register 5	D1340	Set the special D value via system # variable, paring with variable #1900.	R
HMI input register 6	D1341	Set the special D value via system # variable, paring with variable #1901.	R

Function name	Special D code	Description	Device type
HMI input register 7	D1342	Set the special D value via system # variable, paring with variable #1902.	R
HMI input register 8	D1343	Set the special D value via system # variable, paring with variable #1903.	R
HMI input register 9	D1344	Set the special D value via system # variable, paring with variable #1904.	R
HMI input register 10	D1345	Set the special D value via system # variable, paring with variable #1905.	R
HMI input register 11	D1346	Set the special D value via system # variable, paring with variable #1906.	R
HMI input register 12	D1347	Set the special D value via system # variable, paring with variable #1907.	R
HMI input register 13	D1348	Set the special D value via system # variable, paring with variable #1908.	R
HMI input register 14	D1349	Set the special D value via system # variable, paring with variable #1909.	R
HMI input register 15	D1350	Set the special D value via system # variable, paring with variable #1910.	R
HMI input register 16	D1351	Set the special D value via system # variable, paring with variable #1911.	R
M code data	D1368	When M code is executed in a program, the corresponding device of D1368 will be triggered. For example, when executing M3 command, the value of D1368 is 3. The M codes mentioned here do not include M00, M01, M02, M30, M98, M99 and the M code used for macro call.	R
S code data	D1369	When S code is executed in a program, the S code value will be saved in register D1369. It will not be triggered when applying the S code. (Unit: rpm)	R
T code data (command)	D1370	When T code is executed in a program, the T code value will be saved in register D1370. It will not be triggered when the T code is specified for macro. D1370 data is related to the station ID in the tool magazine. The T code specified by the program will correctly display in D1370 only if it is within the range of station ID specified in the tool magazine.	R

Function name	Special D code	Description	Device type
Standby tool No. (tool magazine 1)	D1371	The Register Magazine in tool magazine 1 displays the tool No. corresponding to the standby tool pot (D1373).	R
Tool pot offset (tool magazine 1)	D1372	It is used to save the tool pot offset between the positions specified in D1370 (T code data) and D1371 (standby tool No.) in tool magazine 1. When the tool magazine is moving forward and backward during tool exchange (M1172/1173), the current tool magazine needs to rotate according to the value in D1372 for compensating the offset.	R
Standby tool pot (tool magazine 1)	D1373	The standby tool pot No. in tool magazine 1.	R
Tool No. in use (tool magazine 1)	D1374	The tool No. that is currently in use in tool magazine 1.	R
Standby tool No. (tool magazine 2)	D1375	The Register Magazine in tool magazine 2 displays the tool No. corresponding to the standby tool pot (D1377).	R
Tool pot offset (tool magazine 2)	D1376	It is used to save the offset between the positions specified in D1370 (T code data) and D1375 (standby tool No.) in tool magazine 2. When the tool magazine is moving forward and backward during tool exchange (M1172/1173), the current tool magazine needs to rotate according to the value in D1376 for compensating the offset.	R
Standby tool pot (Tool magazine 2)	D1377	The standby tool pot No. in tool magazine 2.	R
Tool No. in use (tool magazine 2)	D1378	The tool No. that is currently in use in tool magazine 2.	R
Feed rate	D1379	Access the feed rate during cutting.	R
Spindle speed	D1380	Access spindle speed.	R
Current G code value when using G01,G02,G03	D1383	When using G01, G02 or G03, the value varies with the currently using G code. For example: G01 = 1, G02 = 2, G03 = 3	R
Axis X mechanical coordinates	D1384	The current mechanical coordinates of axis X.	R
Axis Y mechanical coordinates	D1386	The current mechanical coordinates of axis Y.	R

5

Function name	Special D code	Description	Device type
Axis Z mechanical coordinates	D1388	The current mechanical coordinates of axis Z.	R

■ **Special D for HMI output**

Variable #1833~#1848 can be used in a machining program to access the register values of MLC "HMI output registers". Variable 1833~#1848 are paired with MLC HMI output registers D1024 ~D1039 respectively. For example, #1833 is paired with D1024, and so forth, for total 16 pairs. If the output value of D1024 in MLC is 100, the value of #1833 in NC program will be 100 accordingly. That is, the value of #1833 varies with the value of register D1024.

Please refer to the table below for MLC output registers and the corresponding variables in NC system (MLC > NC):

Function name	Special D code	Variable ID	Function name	Special D code	Variable ID
HMI output register 1	D1024	#1833	HMI output register 9	D1032	#1841
HMI output register 2	D1025	#1834	HMI output register 10	D1033	#1842
HMI output register 3	D1026	#1835	HMI output register 11	D1034	#1843
HMI output register 4	D1027	#1836	HMI output register 12	D1035	#1844
HMI output register 5	D1028	#1837	HMI output register 13	D1036	#1845
HMI output register 6	D1029	#1838	HMI output register 14	D1037	#1846
HMI output register 7	D1030	#1839	HMI output register 15	D1038	#1847
HMI output register 8	D1031	#1840	HMI output register 16	D1039	#1848

■ **Special D for MPG operations**

Function name	Special D code	Function name	Special D code
MPG operation mode ID	D1040	Set MPG pulse magnification	D1042
MPG operation channel selection	D1041	Axis selection of MPG movement	D1043

Special D for HMI input

Variable #1896~#1911 can be used in a machining program to write in the signal value of MLC "HMI input register". Variable #1896 ~ #1911 are paired with MLC HMI output registers D1336 ~ D1351 respectively. For example, #1896 is paired with D1336, and so forth, for total 16 pairs. If #1896 = 101 in NC program, the value of D1336 in MLC is 101 as well. That is, D1336 in MLC varies with variable #1896 in NC system.

Please refer to the table below for MLC input registers and the corresponding variables in NC system (NC > MLC):

Function name	Special D code	Variable ID	Function name	Special D code	Variable ID
HMI input register 1	D1336	#1896	HMI input register 9	D1344	#1904
HMI input register 2	D1337	#1897	HMI input register 10	D1345	#1905
HMI input register 3	D1338	#1898	HMI input register 11	D1346	#1906
HMI input register 4	D1339	#1899	HMI input register 12	D1347	#1907
HMI input register 5	D1340	#1900	HMI input register 13	D1348	#1908
HMI input register 6	D1341	#1901	HMI input register 14	D1349	#1909
HMI input register 7	D1342	#1902	HMI input register 15	D1350	#1910
HMI input register 8	D1343	#1903	HMI input register 16	D1351	#1911

Special D for NC axes

These special D signals are transmitted from NC to MLC, which are used for accessing the mechanical coordinates.

Function name	Special D code	Function name	Special D code
Axis X mechanical coordinates	D1384	Axis Z mechanical coordinates	D1388
Axis Y mechanical coordinates	D1386		

Special D input for feed rate and speed

Function name	Special D code	Function name	Special D code
Adjustment of cutting feed rate	D1056	Spindle speed adjustment rate	D1060
Speed adjustment of rapid movement	D1058	Set the speed of Jog and Dry run	D1062
Feed rate	D1379	Spindle speed	D1380

Special D for MLC axes

Function name	Special D code	Function name	Special D code
Axis X position command (MLC axis)	D1064	Axis X speed (MLC axis)	D1082

5

Function name	Special D code	Function name	Special D code
Axis Y position command (MLC axis)	D1066	Axis Y speed (MLC axis)	D1084
Axis Z position command (MLC axis)	D1068	Axis Z speed (MLC axis)	D1086
Axis A position command (MLC axis)	D1070	Axis A speed (MLC axis)	D1088
Axis B position command (MLC axis)	D1072	Axis B speed (MLC axis)	D1090
Axis C position command (MLC axis)	D1074	Axis C speed (MLC axis)	D1092
-	-	Spindle speed (MLC axis)	D1100

- **Special D for workpiece quantity**

Function name	Special M code	Function name	Special M code
Number of the processed products	D1022	Number of the processing target	D1023

- **Special D for calling macro with one key**

Function name	Special M code
Calling macro file name	D1111

- **Special D for M, S, T codes**

Function name	Special M code	Function name	Special M code
M code data	D1368	T code data (command)	D1370
S code data	D1369		

- **Special D for tool magazines**

Function name	Special M code	Function name	Special M code
Standby tool No. (tool magazine 1)	D1371	Standby tool No. (tool magazine 2)	D1375
Tool pot offset (tool magazine 1)	D1372	Tool pot offset (tool magazine 2)	D1376
Standby tool pot (tool magazine 1)	D1373	Standby tool pot (tool magazine 2)	D1377
Tool No. in use (tool magazine 1)	D1374	Tool No. in use (tool magazine 2)	D1378

MLC Application Examples 6

This chapter introduces the common MLC applications, including analog spindle gear switch, return from tapping interruption, and call macro by one key.

6.1 Application of analog spindle gear switch

6

For the machines with spindle gear switch function, MLC can be used to control external mechanism for gear switch. The gear ratio of the current spindle after the gear switch will be transferred to MLC system for calculating and giving instructions. An application example is given as below to introduce to the basic spindle gear switch in MLC. As the machines are of different operating conditions, the example will be based on following hypothesizes:

M Code expression	Output contact	Input contact	M1122 = Bit0 M1123 = Bit1	Spindle gear switch parameter(Shift gear ratio)
M69 neutral	Y256 = Neutral	X256 switch to neutral point	-	-
M70 switch to gear 1	Y257 switch to gear 1	X257 switch to gear 1	00	Gear 1 (422 numerator / 423 denominator)
M71 switch to gear 2	Y258 switch to gear 2	X258 switch to gear 2	01	Gear 2 (424 numerator / 425 denominator)
M72 switch to gear 3	Y259 switch to gear 3	X259 switch to gear 3	10	Gear 3 (426 numerator / 427 denominator)
M73 switch to gear 4	Y260 switch to gear 4	X260 switch to gear 4	11	Gear 4 (428 numerator / 429 denominator)

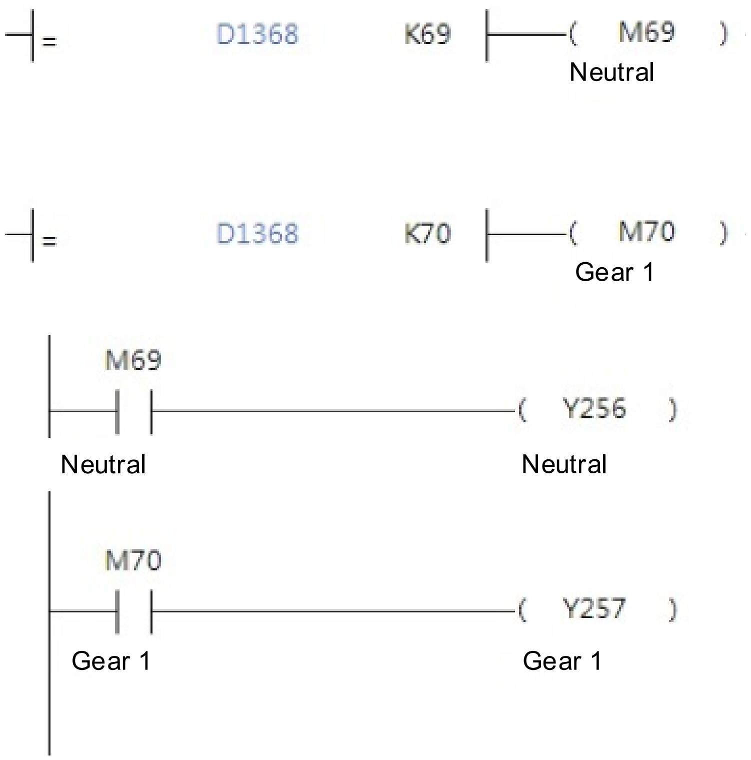
Example:

When the system reads G code instruction “M3S1000”, the spindle will be instructed to move forward with 1000 rpm. Assuming that the spindle is at gear 2 (parameter 424/425) with gear ratio at 1/2, the analog voltage will double (x2) and motor speed will be at S2000 rpm, and the speed of end point on spindle will be at S1000 rpm after mechanical deceleration.

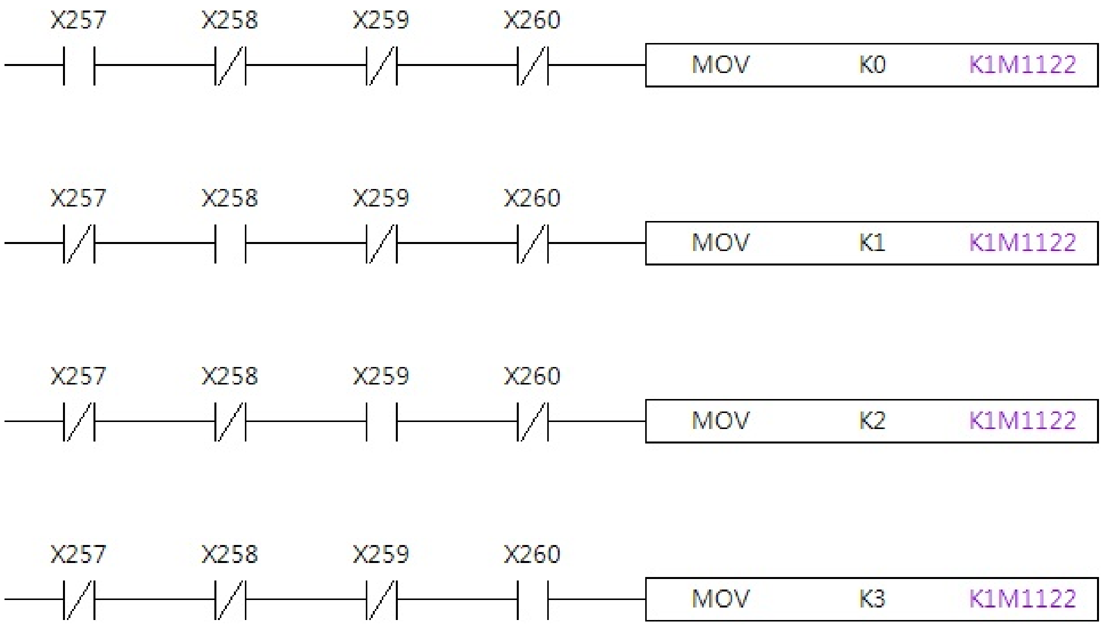
Description of special M, diagram, and Marco with examples:

Function name	Special M code	Function name	Special M code
Spindle moves forward	M1120	Select gear ratio bit 1	M1123
Spindle moves backward	M1121	Spindle positioning control	M1124
Select gear ratio bit 0	M1122	-	-

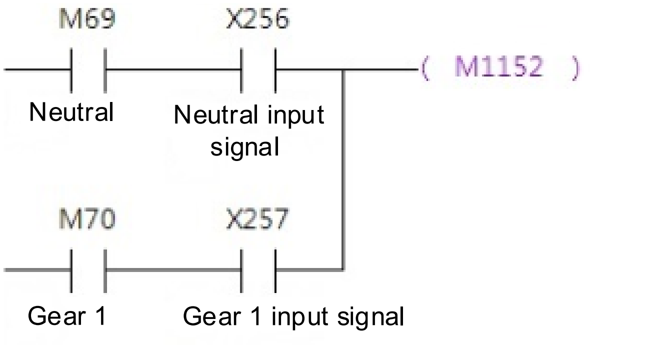
When the system reads M69 and M70 in NC program, MLC will acquire M code according to the value of D1368 and execute the diagram below:



Change the gear ratio of the system by setting M1122 and M1123 to ON or OFF.



M code execution completed flag (M1152) is on as the corresponded DI is triggered. When executing M69 and M70, both will be set to ON and remains so until M1152 is triggered. Thus, M1152 will be triggered when M69 and X256 both on or M70 and X257 are on. The sequence of MLC gear switch is: set M69 or M70 to ON, and then trigger Y256 or Y257. Until the external devices complete their tasks, trigger X256 or X257. Then, M1152 will be triggered by M69 or M70.



6

Macro example of spindle gear switch:

```
#1 = 500            (Define gear range by speed)
#2 = 4000
#3 = 8000
#4 = 12000

#6 = 100            (Define the speed of gear switch)
IF[#19<#1]GOTO10    (#19: speed)
IF[#19<#2]GOTO11
IF[#19<#3]GOTO12
IF[#19<#4]GOTO13

GOTO1000
(1st stage)
N10
#10 = 70
GOTO20

(2nd stage)
N11
#10 = 71
GOTO20

(3rd stage)
N12
#10 = 72
GOTO20

(4th stage)
N13
#10 = 73
GOTO20

N20
#11=#10-69
IF[#1833<#11]GOTO1000
(Compare MLC gear with the instruction's target gear. If they are the same, the instruction will be
skipped.)

S#6         (Start to accelerate until reaching the speed of gear switch)
M69         (Neutral)
M#10        (Inform MLC the gear No. for switching)
G4X2
M99

N1000
S#19
M99
```

6

6.2 Return from tapping interruption

Tapping is usually a coherent movement. However, when irregularity occurs in the process and the RESET key or RMG key is pressed, the tapping interruption flag (M2260) will be triggered. And the machine will halt at the current position. By then, return from tapping interruption flag (M1125) can be triggered in auto mode. Then, the system will automatically return the machine to point R.

Be aware that tapping interruption should not be applied to following situations, otherwise this function will be automatically invalid.

1. Spindle positioning canceled;
2. Program restarted;
3. Arbitrary axis moves;
4. System power ON again;
5. The emergency stop mode of parameter 307 is set to 0.

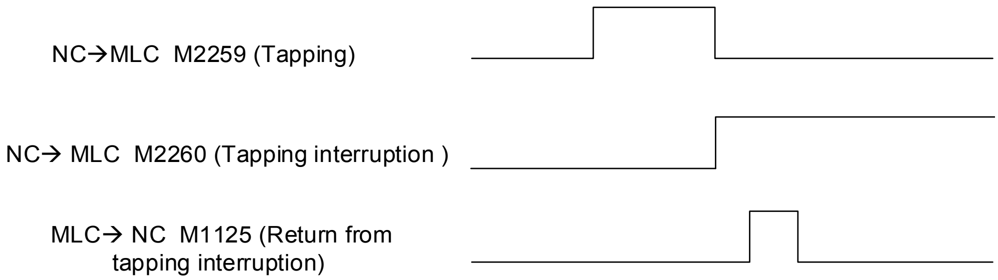
Limitations when applying tapping interruption function:

1. During the tapping process, it is prohibited to switch modes.
2. During the tapping process, it is prohibited to trigger the flag of return from tapping (M1125).
3. If the user wants to disable the function of tapping interruption (execute the program again and conduct arbitrary axis movement), the positioning function has to be removed (M1120 and M1124 are set to 0).
4. When tapping interruption flag M2260 is triggered, the spindle and axis Z will halt at the current position.
5. After the tapping interruption flag M2260 is set to ON, MPG cannot be used for homing operations.

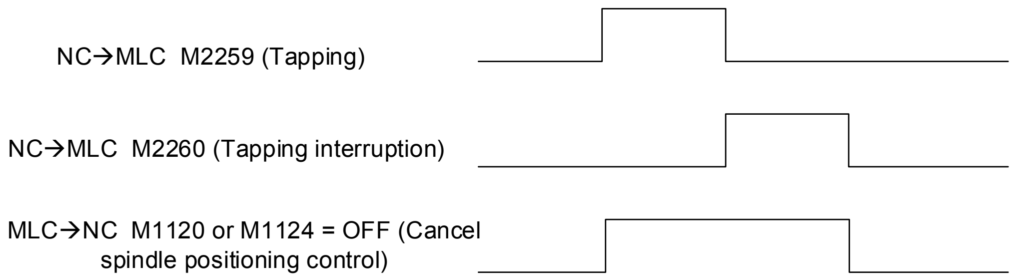
6.2.1 Sequence diagram of tapping interruption and other actions

This section will introduce the sequential relations of tapping in execution, tapping interruption and other related actions, such as return from tapping, tapping interruption canceled, tapping interruption cancelled by Cycle Start, interruption cancelled by axial movement). Please see the following four diagrams for more details.

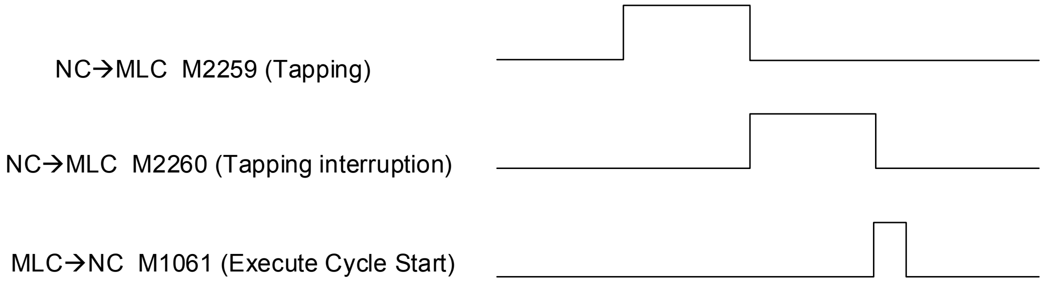
- Sequential relation of tapping in execution, tapping interruption and return from tapping. When the tapping process starts, M2259 (spindle in tapping process) will be set to ON, and it will be set to OFF when the RESET key or EMG key is pressed for tapping interruption. Meanwhile, M2260 (spindle tapping interruption) will be set to ON. By then, user can trigger M1125 (return from tapping) via MLC, and the spindle will return from tapping. M2260 (tapping interruption) will remain ON even after the tapping return process has completed.



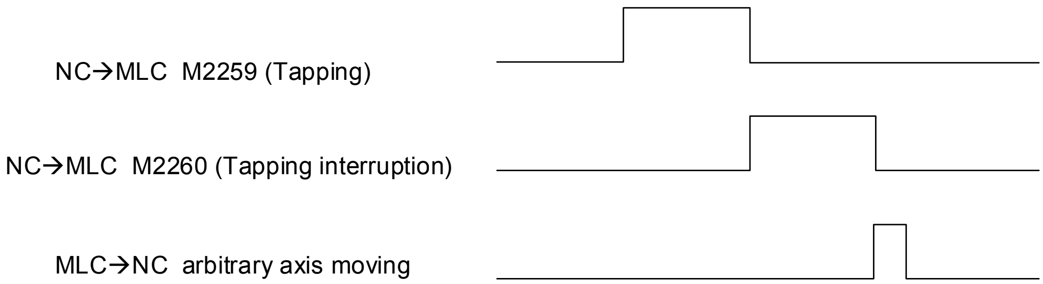
- Sequential relation of tapping in execution, tapping interruption and tapping interruption canceled.
  When the tapping process starts, the system will set M2259 to ON. When the RESET key or EMG key is pressed for tapping interruption, the system will set M2259 to OFF and set M2260 to ON. By then, user can set to OFF either M1120 (spindle moves forward) or M1124 (spindle positioning control) via MLC. Meanwhile, the system will set M2260 to OFF.



- Sequential relation of tapping in execution, tapping interruption, and tapping interruption cancel by Cycle Start.
  When the tapping process starts, the system will set M2259 to ON. When the RESET key or EMG key was pressed for tapping interruption, system will set M2259 to OFF and set M2260 to ON. By then, user can trigger M1061 (Cycle Start) again, and the system will set M2260 to OFF at the same time.



- Sequential relation of tapping in execution, tapping interruption, and tapping interruption cancel by axial movement.
  When the tapping process starts, the system will set M2259 to ON. When the RESET key or EMG key is pressed for tapping interruption, the system will set M2259 to OFF and set M2260 to ON. By then, user can trigger M1216 ~ M1219 or M1226 ~ M1229 (axial movement) through MLC. Meanwhile, the system will set M2260 to OFF.



6

6.3 Call macro by one key

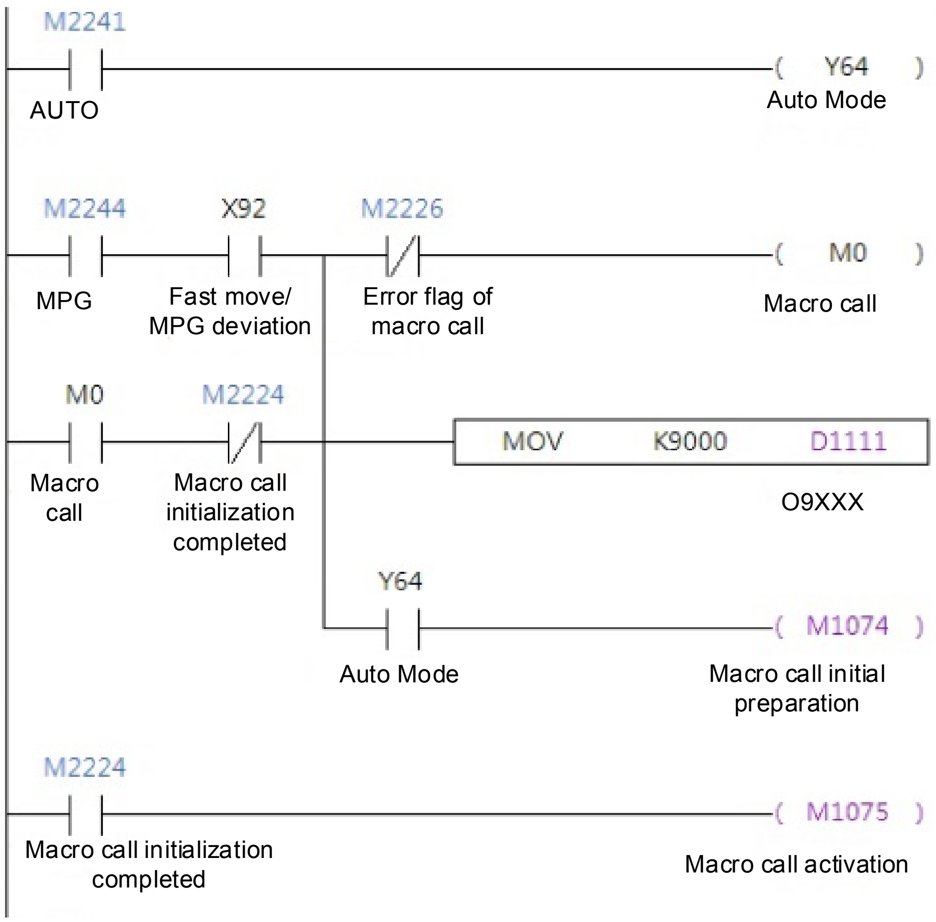
Call macro by one key means to trigger a DI from external device to call a Macro file and execute the file content. Depending on different conditions, users can edit MLC diagram to specify the macro that is requested.

The special M and D for calling macro by one key are listed in the table below.

Function name	Special M code	Function name	Special M code
Initialize for calling the macro	M1074	Activating flag	M2225
Activate the macro	M1075	Error flag	M2226
Initialization completed	M2224	-	-

Function name	Special D code
Calling macro file name	D1111

The diagram of calling macro by one key is shown as below. In this case, the X92 (quick key) is set as the DI trigger.



6

6.4 Reset system before calling macro by one key

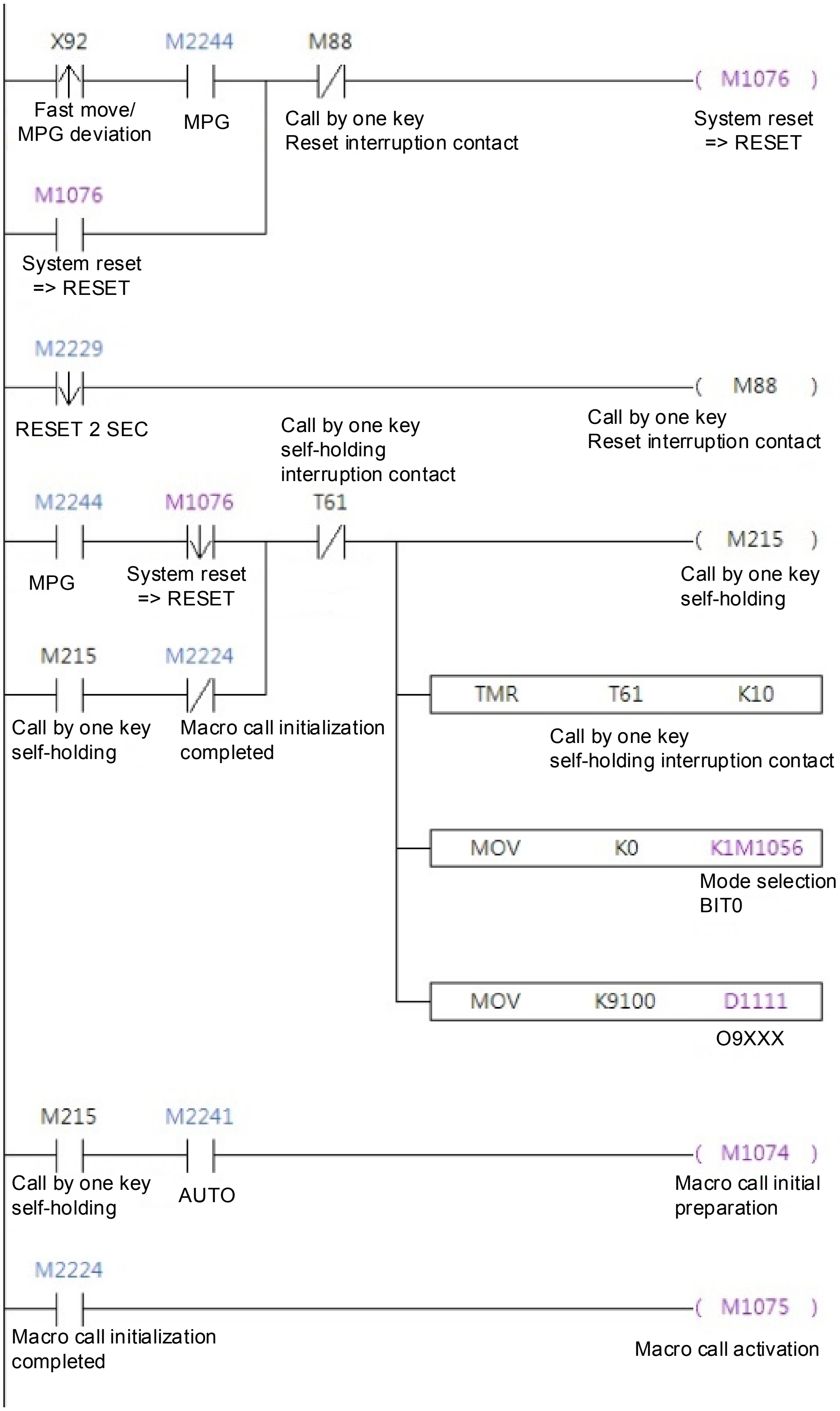
Call macro by one key means to trigger a DI from external device to call a Macro file and conduct execution in accordance with the main file specifications. Depending on different conditions, uses can edit MLC diagram to specify the macro that is requested. If there are remaining values of the system variables or # variables, errors will occur in the program. In order to ensure the program operates correctly, it is suggested using the one key function that resets the system before calling Marco files.

The special M and D for calling macro by one key are listed in the table below.

Function name	Special M code	Function name	Special M code
Initialize for calling the macro	M1074	Initialization completed	M2224
Activate the macro	M1075	Activating flag	M2225
System reset	M1076	Error flag	M2226

Function name	Special D code
Calling macro file name	D1111

Reset the system before calling macro by one key. The diagram is shown as below. In this case, the X92 (quick key) is set as the DI trigger.



6

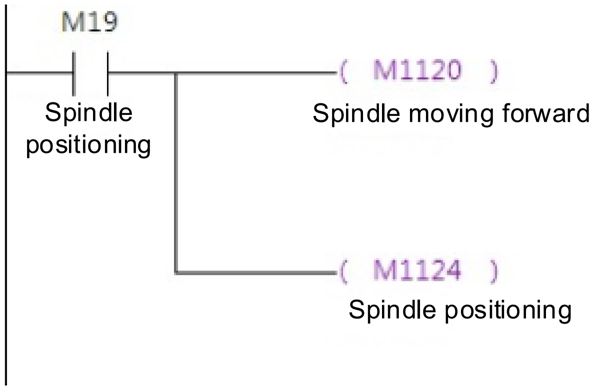
6.5 MLC for spindle positioning and spindle moving forward

Due to the processing methods and tool shape, the exact current angle of the spindle or the same tool starting point and ending point are required in some processing operations. Spindle positioning function can be used to meet these requirements. This function needs to simultaneously trigger the positioning flag and the spindle moving forward flag. See below example:

The special M related to spindle positioning:

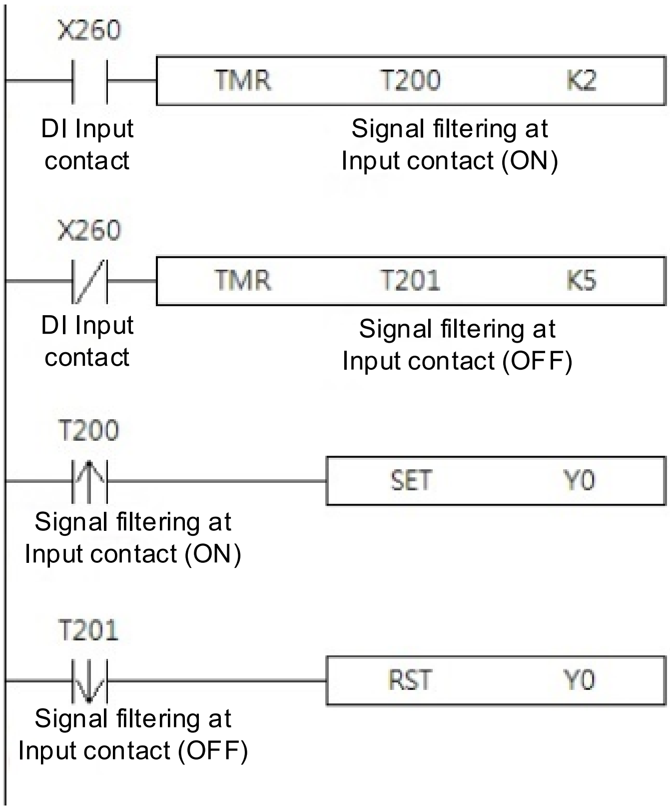
Function name	Special M code	Function name	Special M code
Spindle moves forward	M1120	Select gear ratio bit 1	M1123
Spindle moves backward	M1121	Spindle positioning control	M1124
Select gear ratio bit 1	M1122	Spindle returns from tapping	M1125

Spindle positioning function needs to trigger at least two special M, M1120 and M1124.



6.6 DI signal noise filtering function

For the DI trigger contacts that are very easy to be interfered by signal noise, users can use the counters in MLC as the noise filtering function. The filtering requirements: ON status lasts over 20 ms, OFF status lasts over 50 ms.



(This page is intentionally left blank.)

Revision History

Release Date	Version	Revised sections	Revision contents
July, 2013	V1.0	-	-
Sep., 2016	V2.0	1.2.2	Add section introduction and new notes.
		1.3.2	Add section introduction and notes for using output contacts.
		1.4	Add section introduction.
		1.4.1	Add new notes for using auxiliary relays.
		1.5	Add section introduction.
		1.6	Add section introduction.
		1.7	Add section introduction.
		1.7.1	Delete the example of 32-bit counter.
		1.8.1	Add the example of assigning D0 as the 32-bit register. The data register types are changed to 4 types.
		1.8.2	Add new table of the 32-bit indirect index registers.
		1.9	Add section introduction and amend range of indicators for external interruption.
		2.1	Add section introduction. Add new column “Execution speed (us)” to the tables in this section. Delete notes.
		2.2	Add section introduction.
		3.1	Delete the column of Model and Page in the table.
		3.2	Add section introduction.
		3.3	Add section introduction. Add descriptions to the instruction of assigning continuous numbers.
		3.4	Add section introduction. Add introduction of how to use registers.
		3.5	Delete the column of Model and Page in the table.
		4	Change the table format.

Release Date	Version	Revised sections	Revision contents
		4.1	The table contents of API-00 have been changed. The table in example 3 has been added with new column of “C78, C79”.
		4.1	Amend the table in the supplementary notes of API-05, which explains the flags that disable the insertion of interruption indicators in NC series models
		5	New chapter.
		6	New chapter.

For more information about NC Series MLC Application Manual, please refer to:

(1) NC Series Command Guidelines (to be released in September, 2016)

(2) Delta CNC Solution NC Series User Manual for Operation and Maintenance