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

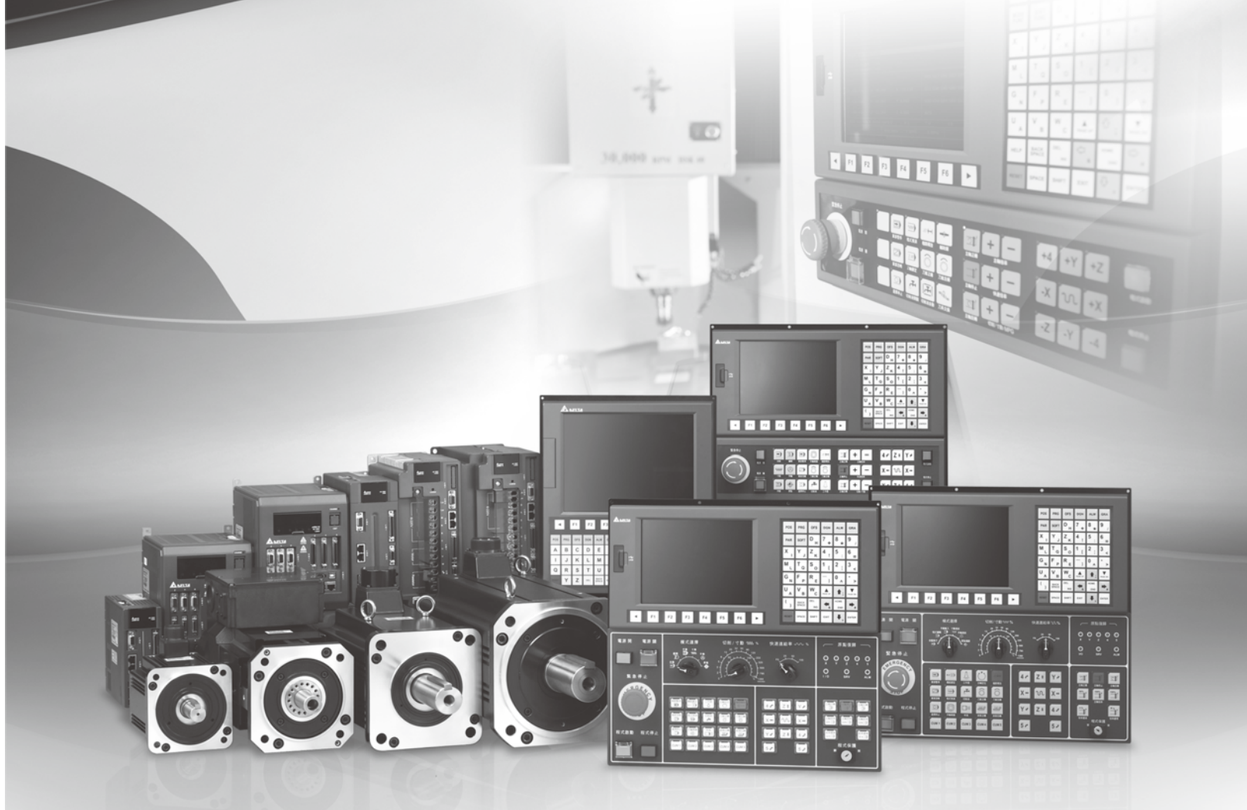
Delta Electronics (India) Pvt. Ltd.
Plot No 43 Sector 35, HSIIDC
Gurgaon, PIN 122001, Haryana, India
TEL : 91-124-4874900 / FAX : 91-124-4874945

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 NC300 MLC Application Manual



Delta CNC Solution
NC300 MLC Application Manual

www.delta.com.tw/ia

Smarter. Greener. Together.

Table of Content

Chapter 1: Device Functions Offered by NC300-MLC

1.1 List of devices offered by NC300-MLC

Range and number of devices available in MLC

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

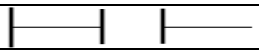
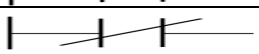
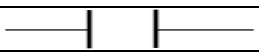
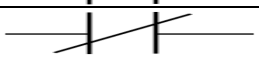
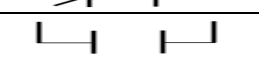
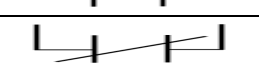
List of settings of MLC's devices

<table>
<tr><th colspan="2">Device name</th><th colspan="6">Purposes</th><th>Outage retaining</th><th>Function</th><th>Points</th></tr>
<tr><td colspan="2" rowspan="2">X mechanic input signal (bit)</td><td>On Board</td><td>MPG</td><td>Undefined</td><td>2nd. panel</td><td colspan="2">Remote</td><td rowspan="2"></td><td rowspan="2">Corresponds to external input points</td><td rowspan="2">296</td></tr>
<tr><td>X0~X27</td><td>X28~X33</td><td>X34~X63</td><td>X64~X255</td><td colspan="2">X256~X511</td></tr>
<tr><td colspan="2">Y mechanic input signal (bit)</td><td colspan="2">Y0~Y27</td><td>Y28~Y63</td><td>Y64~Y255</td><td colspan="2">Y256~Y511</td><td>None</td><td>Corresponds to external output points</td><td>296</td></tr>
<tr><td colspan="2" rowspan="4">M auxiliary relay (bit)</td><td colspan="2">General use</td><td colspan="2">Special M for system</td><td colspan="2">Special M for MLC</td><td rowspan="4">M512~ M1023</td><td>[General function]</td><td rowspan="4">3072</td></tr>
<tr><td colspan="7">M0~M3071</td><td rowspan="2">Contacts that can be switched On/Off in program</td></tr>
<tr><td colspan="3" rowspan="2">M0~M511</td><td>MLC->NC</td><td>NC->MLC</td><td>MLC</td></tr>
<tr><td>M1024~ M1215</td><td>M1696~ M1983</td><td>M2816~ M3071`</td><td>[Special M function]
Communicate between system and MLC</td></tr>
<tr><td colspan="2">A Alarm (bit)</td><td colspan="7">A0~A511</td><td>None</td><td>Custom MLC alarm, format for NC display: A0 + alarm message</td><td>512</td></tr>
<tr><td rowspan="2">T</td><td>Timer (bit)</td><td colspan="4">T0~T199 (in unit of 100ms)</td><td colspan="3">T200~T255 (in unit of 10ms)</td><td rowspan="2">None</td><td rowspan="2">Timer assigned by TMR command. T contact of the same code turns ON when timing is reached.</td><td rowspan="2">256</td></tr>
<tr><td>Timer (Word)</td><td colspan="7">T0~T255 (16-bit, range 0~65535)</td></tr>
<tr><td rowspan="5">C Counter</td><td>(Bit)</td><td colspan="7">C0~C79</td><td>None</td><td rowspan="4">Counter assigned by CNT (DCNT) command. C contact of the same code turns ON when count value is reached. C78,C79 hardware counting
32-bit count down can be enabled only when corresponding special M is On. E.g. C64's countdown corresponds to M1200 while C65 to M1201</td><td rowspan="4">80</td></tr>
<tr><td rowspan="4">Word or D Word</td><td colspan="2">16-bit (count up)</td><td colspan="2">32-bit (count up and down)</td><td colspan="2">32-bit high speed (count up and down)</td><td rowspan="3">None</td></tr>
<tr><td>Range</td><td>0~ 65,536</td><td colspan="2">-2,147,483,648~ +2,147,483,647</td><td colspan="2">-2,147,483,648~ +2,147,483,647</td></tr>
<tr><td colspan="2">C0~C63</td><td colspan="2">C64~C77</td><td colspan="2">C78 C79</td></tr>
<tr><td colspan="2">None</td><td colspan="2">M2832~M2845 Opening countdown afterwards</td><td colspan="2">Opening countdown with parameter MLC(#312)</td><td></td></tr>
<tr><td colspan="2" rowspan="3">D data register (word)</td><td colspan="2">General use</td><td colspan="2">Special D for system</td><td colspan="2">Special D for MLC</td><td rowspan="3">D512~ D1023</td><td>Serve as the memory area for data storage. C and T can function as register as well.
[Special D function]
Communicate between system and MLC</td><td rowspan="3">1536</td></tr>
<tr><td colspan="2" rowspan="2">D0~D511 (-32768~+32767)</td><td>MLC-->NC</td><td>NC-->MLC</td><td colspan="2">For MLC</td></tr>
<tr><td>D1024~ D1118</td><td>D1336~ D1384</td><td colspan="2">D1456~ D1535</td></tr>
<tr><td colspan="2">V register (word)</td><td colspan="7">V0~V7 (-32768~+32768)</td><td>None</td><td rowspan="2">V, Z can function as indirect assignment</td><td>8</td></tr>
<tr><td colspan="2">Z register (word)</td><td colspan="7">Z0~Z7 (-32768~+32768)</td><td>None</td><td>8</td></tr>
<tr><td colspan="2">Indicators</td><td colspan="5">Function</td><td colspan="2">Range</td><td></td><td>[Function]</td><td></td></tr>
<tr><td colspan="2">N (Loop indicator)</td><td colspan="5">For primary control loop</td><td colspan="2">N0~N7</td><td>None</td><td>Primary control loop's control points</td><td>8</td></tr>
<tr><td colspan="2">P (Jump indicator)</td><td colspan="5">For CJ, CALL</td><td colspan="2">P0~P255</td><td>None</td><td>Location flag of CJ, CALL</td><td>256</td></tr>
<tr><td colspan="2" rowspan="3">I (Interrupt indicator)</td><td rowspan="3">For interruption</td><td colspan="4">On Board hardware</td><td colspan="2">IX00~IX07</td><td rowspan="3">None</td><td>8 external hardware interrupts available on the main board</td><td rowspan="3">34</td></tr>
<tr><td colspan="4">Hardware counting</td><td colspan="2">IC00~IC01</td><td>2 high speed count interrupts available on the main board</td></tr>
<tr><td colspan="4">Remote hardware</td><td colspan="2">IR00~IR23</td><td>3 external hardware interrupts available on each remote card</td></tr>
<tr><td colspan="2" rowspan="2">K CONSTANT</td><td colspan="2" rowspan="2">Decimal constant</td><td colspan="5">K-32,768~K+32,767 (16-bit computing)</td><td>None</td><td></td><td></td></tr>
<tr><td colspan="5">K-2,147,483,648~ K+2,147,483,647 (32-bit computing)</td><td>None</td><td></td><td></td></tr>
<tr><td colspan="2">F Floating point number</td><td colspan="2">3 decimal places</td><td colspan="5">-3.4+10^38 ~ 3.4+10^38</td><td>None</td><td></td><td></td></tr>
</table>

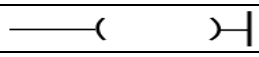
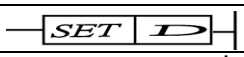
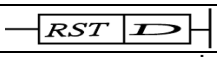
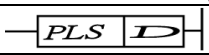
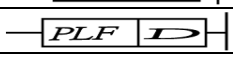
1.2 List of NC300-MLC Edit commands

Sequence command:

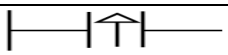
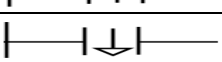
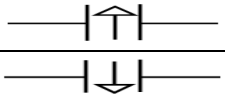
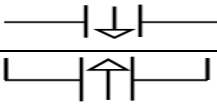
Contact commands

Type	Code	Symbol
Contact	LD	
	LDI	
	AND	
	ANI	
	OR	
	ORI	

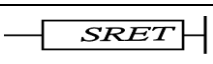
Output commands

Type	Code	Symbol
Output	OUT	
	SET	
	RST	
	PLS	
	PLF	

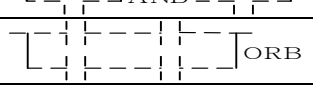
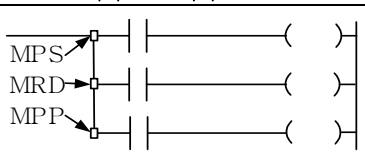
Rising and falling edge detection

Type	Code	Symbol
Rising and falling edge detection	LDP	
	LDF	
	ANDP	
	ANDF	
	ORP	
	ORF	

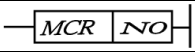
Exit command

Type	Code	Symbol
Program end	END	
	FEND	
Interrupt end	IRET	
Subroutine end	SRET	

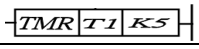
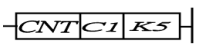
Combining commands

Type	Code	Symbol
Combine	ANB	
	ORB	
	MPS	
	MRD	
	MPP	

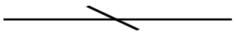
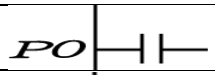
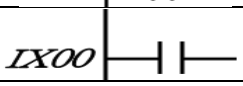
Main control commands

Type	Code	Symbol
Main control	MC	
	MCR	

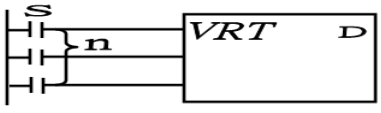
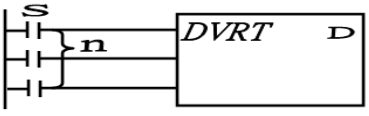
Timer, Counter

Type	Code	Symbol
Timing	TMR	
16-bit count	CNT	
32-bit count	DCNT	

Other commands

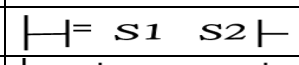
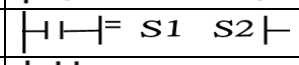
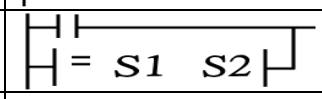
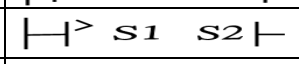
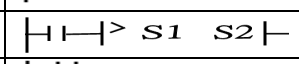
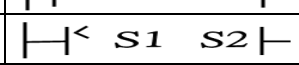
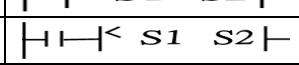
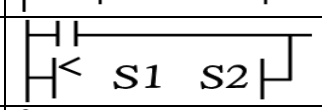
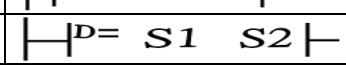
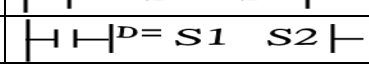
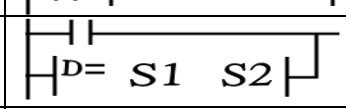
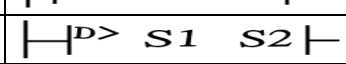
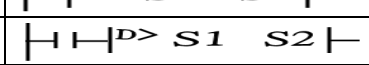
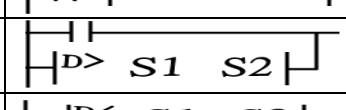
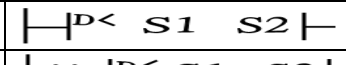
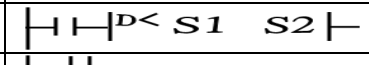
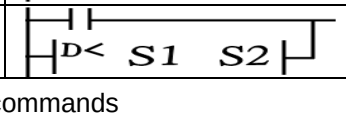
Type	Code	Symbol
Invert phase	INV	
No processing	NOP	
Indicator	P	
Interrupt	I	

Logic switch command

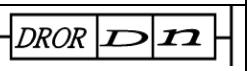
Type	Code	Symbol
16-bit	VRT	
32-bit	DVRT	

Compare and application commands

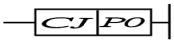
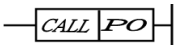
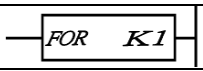
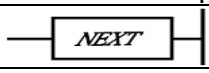
Compare commands (including <>, <=, >= without description)

Type	Code	Symbol
16-bit data compare	LD=	
	AND=	
	OR=	
	LD>	
	AND>	
	OR>	
	LD<	
	AND<	
	OR<	
32-bit data compare	DLD=	
	DAND=	
	DOR=	
	DLD>	
	DAND>	
	DOR>	
	DLD<	
	DAND<	
	DOR<	

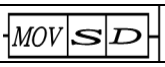
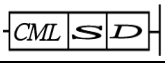
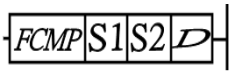
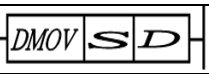
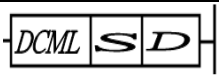
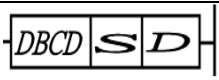
Rotate and shift commands

Type		Code	Symbol
16-bit data	Right rotate	ROR	
	Left rotate	ROL	
32-bit	Right rotate	DROR	
	Left rotate	DROL	

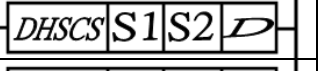
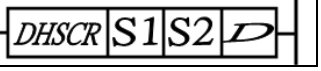
Loop control commands

Type	Code	Symbol
Conditional jump	CJ	
Call subroutines	CALL	
Enable interruption	EI	
Disable interruption	DI	
Nest loops start	FOR	
Nest loops end	NEXT	

Send compare commands

Type		Code	Symbol
16-bit data	Data move	MOV	
	Invert sending	CML	
	BIN→BCD convert	BCD	
	BCD→BIN convert	BIN	
Floating point number	Compare setup output	FCMP	
	Data move	DMOV	
32-bit data	Invert sending	DCML	
	BIN→BCD convert	DBCD	
	BCD→BIN convert	DBIN	

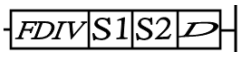
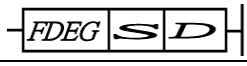
High speed processing commands

Type		Code	Symbol
16-bit	I/O update	REF	
32-bit	Compare settings	DHSCS	
	Compare clear	DHSCR	

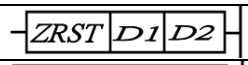
Compare Computing and Application

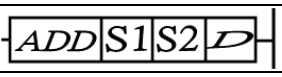
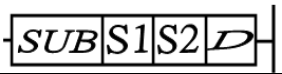
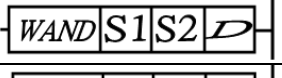
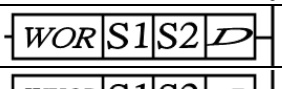
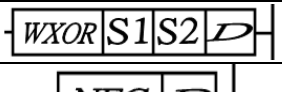
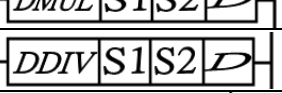
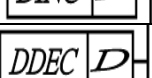
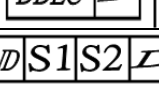
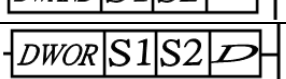
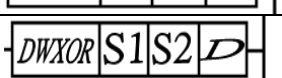
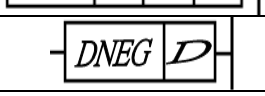
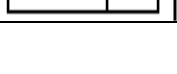
Compare computing and application commands 2

Arithmetic logic computing command and floating point command

Type		Code	Symbol
Floating points computing	Floating point number addition	FADD	
	Floating point number subtraction	FSUB	
	Floating point number multiplication	FMUL	
	Floating point number division	FDIV	
	Floating point →Integer	FINT	
	Integer→ Floating point number	FDOT	
	Degree→ Radian	FRAD	
	Radian→ Degree	FDEG	

Data processing command

Type		Code	Symbol
16-bit	Zone reset	ZRST	
	Decoder	DECO	
	Encoder	ENCO	
	Bit ON detect	BON	
	Alarm point output	ANS	
	Alarm point reset	ANR	
32-bit	Bit ON detect	DBON	

Type		Code	Symbol
16-bit data	BIN addition	ADD	
	BIN subtraction	SUB	
	BIN multiplication	MUL	
	BIN division	DIV	
	BIN minus one	INC	
	BIN add one	DEC	
	AND operation	WAND	
	OR operation	WOR	
	XOR operation	WXOR	
	Get negative value (Two's complement)	NEG	
32-bit data	BIN addition	DADD	
	BIN subtraction	DSUB	
	BIN multiplication	DMUL	
	BIN division	DDIV	
	BIN minus one	DINC	
	BIN add one	DDEC	
	AND operation	DWAND	
	OR operation	DWOR	
	XOR operation	DWXOR	
	Get negative value (Two's complement)	DNEG	

Convenience commands

Type	Code	Symbol
On/Off alternate	ALT	

1.3 Values and constants

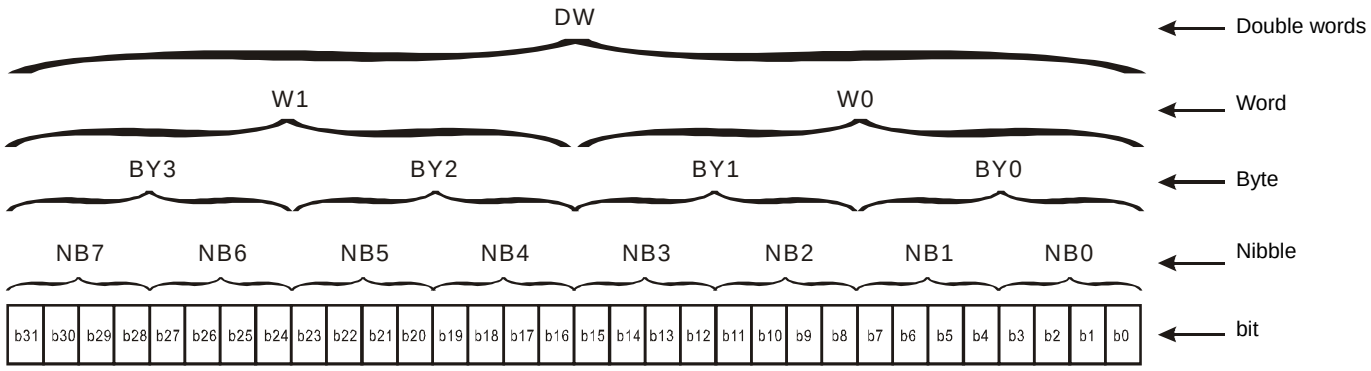
NC300-MLC employs five types of numeric values for various kind of control computation. Tasks and functions of each type of numeric values are summarized below.

1. Binary Number (BIN)

   Digital computing and storage within MLC are executed in binary format. The binary system and terms used in this manual are highlighted below:

Bit:	Bit is the fundamental unit of a binary numeric value. It features only two states: 0 and 1.
Nibble:	Composed of four consecutive bits (e.g. bit0 ~ bit3) to express decimal numbers 0 ~ 15 or once place, hexadecimal numbers 0 ~ F.
Byte:	Composed of two consecutive nibbles (i.e. 8 bits bit0 ~ bit7) to express two places, hexadecimal numbers 00 ~ FF.
Word:	Composed of two consecutive bytes (i.e. 16-bit, bit0 ~ bit15) to express four places, hexadecimal numbers 0000 ~ FFFF
Double Word:	Composed two consecutive words (i.e. 32-bit, bit0 ~ bit31) to express eight places, hexadecimal numbers 00000000~FFFFFFFF

See diagram below for relations among bit, nibble, byte, word, and double words in the binary system:



2. Decimal Number (DEC)

   Decimal numbers are used by the NC300-MLC system in the cases described below:

- External input and output terminal ID are expressed in decimal numbers:
  External input: X0 ~ X39, X64 ~ X511…(device ID)
  External output: Y0 ~ Y39, Y64 ~ Y511…(device ID)
- ID of M, A, T, C, D, V, Z, K, P, I, and N devices, e.g. M10 and T30.(device ID)
- Setup values for Timer T and Counter C, e.g. TMR T0 K50…(K constant)
- Used as an operand in application commands, e.g. MOV K123 D0…(K constant)

K constant:

Decimal numbers in MLC system are prefixed with the letter K in most cases. E.g. K100 is a decimal constant of value 100.

Exceptions: Constant K can be combined with bit devices X, Y, M, and A to express data in a nibble, byte, word or double word format. Take K2Y10 and K4M100 as the example. Here K1~K4 indicates a combination of 4, 8, 12, and 16 bits respectively.

Constant F:
Floating point numbers are used by the MLC for cases listed below:

- Operand in application command. E.g. FADD F12.3 F0 D0 (here F indicates a Floating point constant).

1.4 ID and function of External input /output contacts [X] / [Y]

ID of input/output contacts (in decimal numbers):

ID of I/O terminals begins at X0 and Y0 for both on board and remote I/O devices respectively:

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	Station 5	Station 6	Station 7	Station 8
			X256~X287	X288~X319	X320~X351	X352~X383	X384~X415	X416~X447	X448~X479	X480~X511
Output Y	Y0~Y27	Y64~Y255	Y256~Y287	Y288~Y319	Y320~Y351	Y352~Y383	Y384~Y415	Y416~Y447	Y448~Y479	Y480~Y511

Note 1: Starting ID of expansion I/O corresponds to its connection station. There are 8 stations with up to 256 points

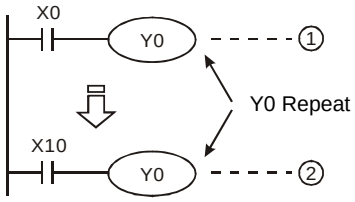
Output relay (or output terminal) is coded in decimal numbers in range of Y256 ~ Y511.

- Function of input/output contacts:

  Function of input contact X: input contact X connects to the input device to read the input signal in MLC. The On/Off status of input contact X varies only with the input device.

- Function of output contact Y:

  The output contact Y is used to send an On/Off signal to drive 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. However, the Code of output coil Y should be used only once in a program. Otherwise its output status will be determined by the last Y output circuit in the program.



Output of Y0 is ultimately determined by circuit ②. That is, by On/Off of X10.

1.5 ID of auxiliary relays: (in decimal numbers)

NC300 models:

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	
	Special purpose (MLC->NC)	M1024~M1215, 192 points, all non-outage for outage retaining	
	Special purpose (NC->MLC)	M1696~M1983, 288 points, all non-outage for outage retaining	
	MLC special purpose	M2816~M3071, 256 points, all non-outage for outage retaining	

Function of auxiliary relay:

Auxiliary relay M features the same output coil and A/B contacts as that of output relay Y and can be use unlimited number of times in a program. The auxiliary relay M can be used in combining control loop but not driving external load. There are three types of them:

1. General purpose auxiliary relay: When a power failure occurs, status of general purpose auxiliary relay is reset to Off during MLC running. It remains in Off status when power is resumed.
2. Outage retaining auxiliary relay: Status of the outage retaining auxiliary relay remains intact when a power failure occurs during MLC operation. It maintains its status before the next power on when power resumes.
3. Special purpose auxiliary relay: Special purpose auxiliary relays are used for NC and MLC status or signal transmission. They are for an individual device's special function, e.g. M2832 is for C64's counting down. Do not use undefined special purpose auxiliary relays. See Section 2.10 Special Relay and Register for special purpose auxiliary relays available to individual models.

1.6 ID and function of custom alarm relay [A]

ID (in decimal numbers) of custom alarm relay: (with range A0~A511)

NC300 model:

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

Custom alarm relay:

The custom alarm relay is used to display alarm signals in the HMI screen. For example, if "A0 = cooling water for motor overload" then warning message "cooling water for motor overload" displays in the HMI screen when A0 is set ON. Custom alarm relay A features the same output coil and A/B contacts as that of output relay Y and can be used with unlimited number of times in a program. The custom alarm relay A can be used in combining control loops but not driving external loads.

When a power failure occurs, the status of the general purpose custom alarm relay is reset to Off during MLC running. It remains in Off status when power is resumed.

1.7 Timer ID and function [T]

NC300 series' timers are coded by decimal numbers T0~T255.

NC300 models:

Timer T	100ms, general purpose	T0~T199, 200 points	Total 256 points
	10ms, general purpose	T200~T255, 55 points (10ms when M1028=On and 100ms when M1028=Off)	

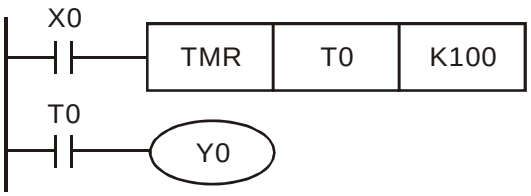
Timer:

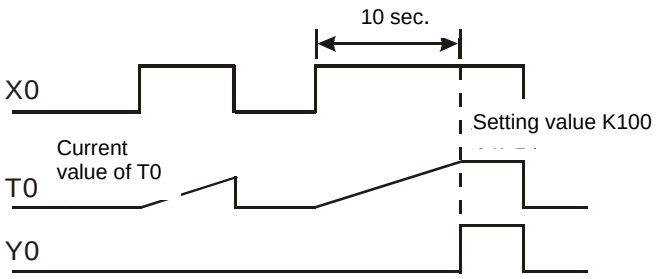
NC300 timers time in unit of 10ms and 100ms and count upward. Its output coil is set On when the current value of the timer equals its setting value. The setting value is a decimal constant (K). It can use a data register D as its setting value.

A timer's actual setup time = unit of timing * settings

1. General purpose Timer:

NC300 models: The general purpose timer times once after each TMR command execution. When the current value of the timer equals its setting value, its output coil turns On.





- In case X0=On, the current value of the timer T0 counts up in units of 100ms. In case the current value of T0 equals the setting value K100, the operand coil turns On.
- In case X0=Off or power outage, the current value of timer T0 resets to 0 and its output coil T0 sets to Off.

A timer's actual setup time = unit of timing * settings

1. Set up by constant K: Set up with constant K.
2. Indirect set up by register D: Set up with data register D indirectly.

1.8 Counter ID and function [C]

NC300 series' counters are coded by decimal numbers

NC300 models:

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 settings M2832~M2845	
	32-bit count up/down, high speed	C78~C79, 2 points, can be changed to count down with parameter (MLC parameter 312)	

Features of NC300's counters:

Item	16-bit counter	32-bit counter
Type	General purpose	High speed
Direction	Up	Up and down
Setting value	0~65,536	-2,147,483,648~+2,147,483,647
Type of setting value	Constant K or data register D	Constant K or data register D (assign both)
Change of the current value	Stop counting when setting value reached	Stop counting when setting value reached
Output contacts	Contact sets and retains On when setting value reached	Contact sets and retains On when setting value reached during counting up Contact sets and retains On when setting value reached during counting down
Reset	The RST command reset current value to 0 and contact to Off	
Contact action	Act in common when scanning ended	

Function of counter:

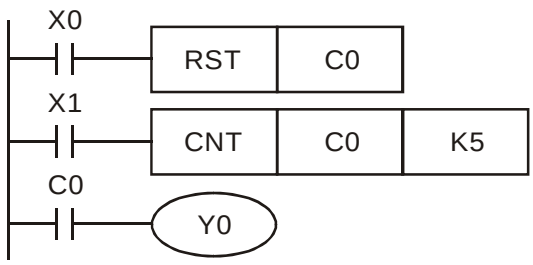
When the counter's counting pulse input signal changes from Off to On with a current value that matches setup one, it then turns On the output coil. The setting value is either a decimal constant K or a data register D.

16-bit counter C0~C63:

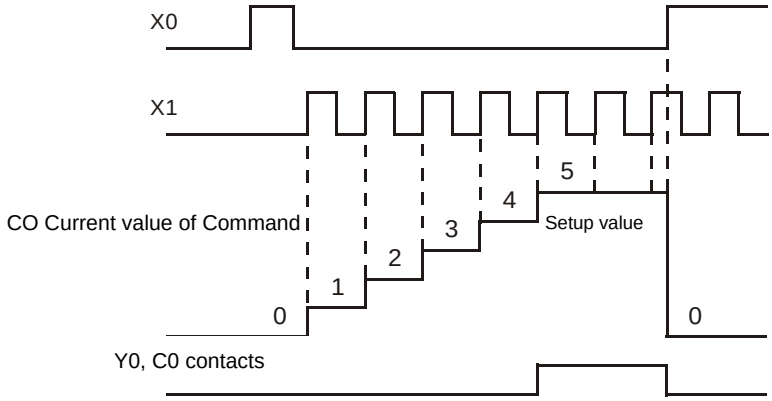
1. Setting value of the 16-bit counter is in range of K0~K65,536. (K0 and K1 functions in the same way and the output contact set On at the first counting.)
2. Setting value of the counter can be either a direct constant K or indirect data register D (excluding data register D1024~D1536).

Example:

```
LD    X0
RST   C0
LD    X1
CNT   C0 K5
LD    C0
OUT   Y0
```



1. In case X0=On, the RST command is executed to reset C0 to 0 and output contact to Off.
2. In case X1 changes from Off→On, the counter counts up by 1.
3. In case counter Command's value matches with setting value K5, the C0 contact turns On and remains so. Later, the trigger signal from X1 does not update C0 (its value remains equal to K5).

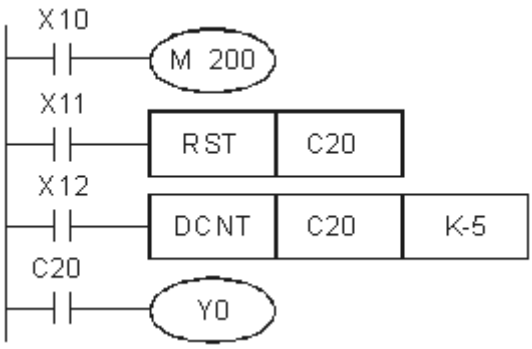


32-bit, general purpose aromatic operation counter C64~C77:

1. The 32-bit, general purpose counter's setting value is in range of K-2,147,483,648~K2,147, 483, 647.
2. 32-bit, general purpose arithmetic operation counter's counting up or down can be switched by auxiliary relay M2834~M2845. For example, M2834=Off indicates C64 is for addition and M2834=On for subtraction.
3. The setting vales can be integral constant K or data register D and the value can be positive or negative. Two consecutive data registers are required for one setup value.
4. Current value of counter changes from 2,147,483,647 to -2,147,483,648 when counting upward and -2,147,483,648 to 2,147,483,647 when counting downward.

Example:

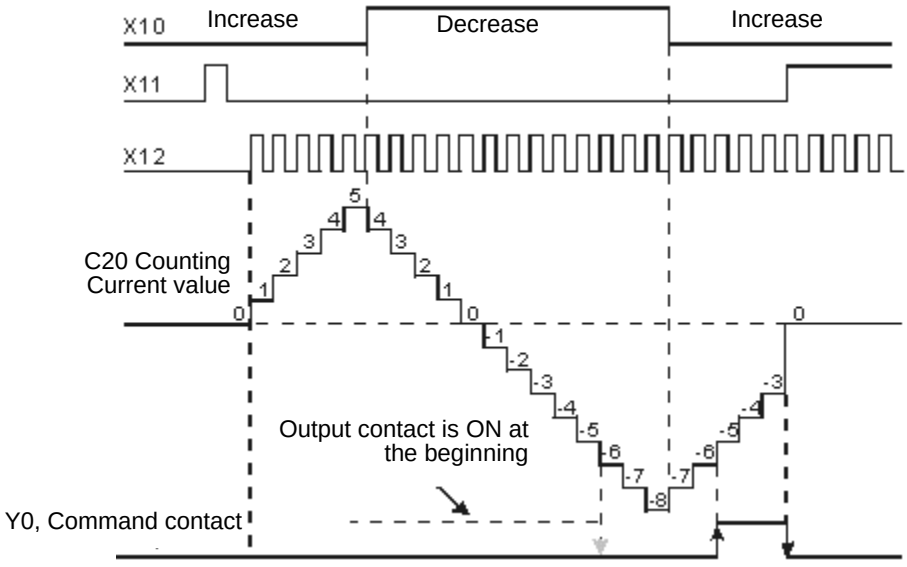
```
LD    X10
OUT   M200
LD    X11
RST   C20
LD    X12
CNT   C20 K-5
LD    C20
OUT   Y0
```



1. The X10 driven M200 determines C20 is either addition or subtraction.
2. In case X11 changes from Off to On, the RST command executes to reset C20 to 0 and

contact to Off.

3. In case X12 changes from Off to On, the counter counts up (increase current value by 1) or counts down (decrease current value by 1).
4. In case current value of counter C20 changes from K-6 to K-5, the C20 contact changes from Off to On and from On to Off when current value changes from K-5 to K-6.



32-bit high speed addition/subtraction counter C78~C79:

1. Setup values of 32-bit high speed addition/subtraction counter are in range of K-2,147,483,648~K2, 147, 483, 647.
2. The 32-bit high speed addition/subtraction counter C78~C79 can do either addition or subtraction operation as determined by parameters.
3. The setting vales can be integral constant K or data register D and the value can be positive or negative. Two consecutive data registers are required for one setup value.
4. Current value of counter changes from 2,147,483,647 to -2,147,483,648 when counting upward and -2,147,483,648 to 2,147,483,647 when counting downward.

Counter's relevant flag signals and special registers:

Flag signal	Function
M2824	Zero flag
M2825	Borrow flag
M2826	Carry flag
M2827	Reset all non-outage retaining zone
M2828	Reset all outage retaining zone
M2829	Changes Y output memory retaining from non-operating to either operating or non-operating.
M2830	All Y outputs banned (Set to LOW)

Flag signal	Function
M2831	Reserved
M2832 ~ M2845	C64 ~ C77 counter's counting direction When M28□□=Off, counter C□□ counts up and down when M28=On.
M2848	HHSC0 reset internal control signal and input contact
M2849	HHSC0 start internal control signal and input contact
M2850	HHSC1 reset internal control signal and input contact
M2851	HHSC1 start internal control signal and input contact
M2856	HHSC0 executes DHSCS or DHSCR (0:DHSCS, 1:DHSCR) command
M2857	HHSC1 executes DHSCS or DHSCR (0:DHSCS, 1:DHSCR)
M2864	I00 interrupt input to on board X0 (1: enable and 0: disable) can be either a programming expression or purely for setup
M2865	I01 interrupt input to on board X1 (Mask external interrupts)
M2866	I02 interrupt input to on board X2

1.9 Registers ID and function [D], [V], [Z]

1.9.1 Data register [D]

The data register is used for keeping 16-bit numeric data in the range of -32,768 ~ +32,767. The left most bit is a sign bit. Two 16-bit registers can be combined into one 32-bit register (D+1, D where the smaller ID represents the lower bits (16-bit)) with the left most bit serving as the sign bit. It can store numeric data in range of -2,147,483,648 ~ +2,147,483,647.

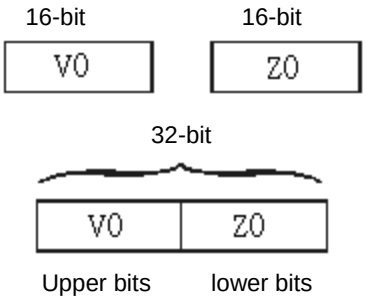
NC300 models:

Data register D	General purpose	D0~D511, 512 points	Total 1536 points
	For outage retaining*	D512~D1023, 512 points, exclusive for outage retaining zone	
	Special purpose (MLC->NC)	D1024~D1118, 95 points.	
	Special purpose (NC->MLC)	D1336~D1384, 49 points.	
	Exclusive for MLC	D1456~D1535, 80 points.	

There are four kinds of registers:

1. General purpose register: When the MLC changes from RUN to STOP data remains intact and is reset to 0 when there is an outage.
2. Register for outage retaining: D512~D1023: When MLC is powered off, data from this zone's registers remains the same as before the outage.
   Use RST or ZRST commands to reset values contained in outage retaining registers.
3. Special purpose register: Each special purpose registers bears its own meaning and usage for system status storage, error message, and monitoring status. Please refer to Section 2.10 Special Relay and Register and Section 2.11 Special Auxiliary Relay and Register for detailed description.
4. Indirect assignment register [V], [Z]: Indirect reference registers are 16-bit ones. The NC300 offers 16 points of V0~V7 and Z0~Z7.
   Register V can be assigned as 32-bit ones. This stops register Z from being used any more.

1.9.2 Indirect reference register [V], [Z]:

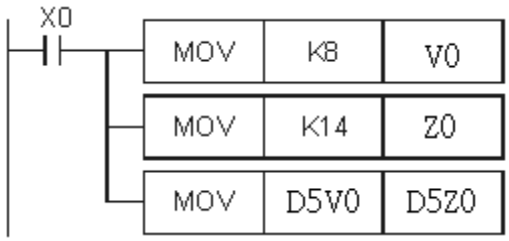


Both V and Z are general purpose 16-bit data registers for unlimited access.

Users can assign the V register to get a 32-bit register. This leads to F being covered by V and Z cannot be used any more. (It is suggested to clear V (including Z) contents to 0 with command DMOV K0 V0 when system power on).

Combination of V and Z when being used as a 32-bit indirect reference register are:

(V0, Z0) , (V1, Z1) (V2, Z2) …(V7, Z7)



In case X0=On, V0=8, Z0=14, D5V0 = D(5+8) =D13, D10Z0 = D(10+14) = D24, and contents in D13 will be moved to D24.

Same as general operands, the indirect reference register can be used for data movement and comparing for bit devices KnX, KnY, KnM, KnA, T, C, and D. It does not support bit device (X, Y, M, A) and constant (K, F) indirect reference function.

The NC300 series offer 16 points in range of V0~Z7.

※ Certain commands do not support this. See Chapter 4 Section 4.4 for modifying operand with indirect reference register V, Z.

1.10 Indicator [N] and [P] and interrupt indicator [I]

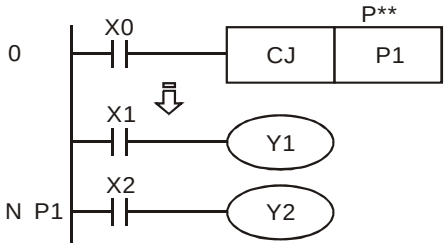
NC300 series:

Indicator	N	For main control loop		N0~N7, 8 points	Main control loop's control points
	P	For CJ, CALL command		P0~P255, 256 points	CJ, position indicator of CALL command
	I	For interruption	On Board hardware interrupt	IX00~IX07, 8 points	Position indicator of interrupt subroutine
			Hardware counting interrupt	IC00~IC01, 2 points	
			Remote I/O hardware interrupt	IR00~IR23, 24 points	

Indicator N: Work together with command MC (the main control initial command) and MCR. After the MC command is active, commands between MC and MCR run in a normal manner. See Chapter 3 for details on the use of MC/MCR commands.

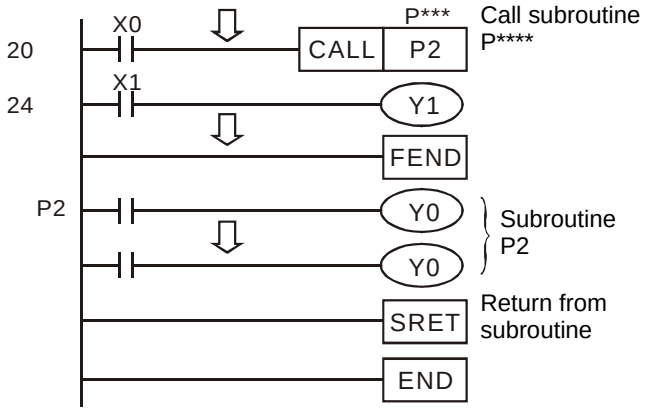
Indicator P: Work together with application commands API 00 CJ, API 01 CALL, and API 02 SRET. See Chapter 4 for description on the use of CJ, CALL, and SRET commands.

CJ conditional jump:



- In case X0=On, the program jumps from address 0 to N (the assigned label P1), keeps executing and ignores statements in between.
- In case X0=Off, the program executes from address 0 downward and ignores command CJ.

CALL for calling subroutine and SRET for ending subroutine:



- In case X0=On, executes CALL command and jumps to subroutine starting at P2. After SRET was executed the program returns to address 24 for execution downward.

Interrupt indicator I:

Interrupt indicator I is used together with application command API 04 EI, API 05 DI, and API 03 IRET. See Chapter 4 for details. The interrupt function needs accompanying command EI for its enabling, DI for disabling, and IRET for returning back to the calling statement.

1. External interrupt: The X0~X5 input ends' input signal is triggered at positive or negative edge. If triggered in the middle, the MLC host's special software jumps to subroutine indicator IX00(X0), IX01(X1), IX02(X2), IX03(X3), IX04(X4), IX05(X5), IX06(X6), IX07(X7) specified by the interrupt execution program after the running command was completed. Interrupts IR00~23 (remote X256~258) correspond to the first three INPUT of the 8 remote I/O modules respectively. The program continues from the interrupting position for sequential downward execution after the IRET command was executed.

2. Interrupt by counting reached: The high speed counter's compare command API 53 DHSCS can be set to break off from the running program and jumps to interrupt subroutine specified by interrupt indicator IC00 and IC01 when the setting value is matched.

Chapter 2: Basic Commands

2.1 Summary of basic commands

General purpose commands

Command code	Function	OPERAND	STEP	Page
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	None	1	
ORB	Parallel connect loop block	None	1	
MPS	Saves it in stack	None	1	
MRD	Stack retrieval (indicator remain intact)	None	1	
MPP	Read stack	None	1	

Output commands

Command code	Function	OPERAND	STEP	Page
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

API	Command code	Function	OPERAND	Execution speed (us)	STEP	Page
				NC300		
36	TMR	16-bit timer	T-K or T-D	9.6	3	
37	CNT	16-bit counter	C-K or C-D (16-bit)	12.8	3	
37	DCNT	32-bit counter	C-K or C-D (32-bit)	14.3	3~4	

Primary control commands

Command code	Function	OPERAND	Execution speed (us)	STEP	Page
			NC300		
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 commands

Command code	Function	OPERAND	Execution speed (us)	STEP	Page
			NC300		
LDP	Start of positive edge detection	X, Y, M, A, T, C	-	2	
LDF	Start of negative edge detection	X, Y, M, A, T, C	-	2	
ANDP	Serial connection of positive edge detection	X, Y, M, A, T, C	-	2	
ANDF	Serial connection of negative edge detection	X, Y, M, A, T, C	-	2	
ORP	Parallel connection of positive edge detection	X, Y, M, A, T, C	-	2	
ORF	Parallel connection of negative edge detection	X, Y, M, A, T, C	-	2	

Upper and lower differential output commands

Command code	Function	OPERAND	Execution speed (us)	STEP	Page
			NC300		
PLS	Upper differential output	Y, M, A	-	1~2	
PLF	Lower differential output	Y, M, A	-	1~2	

Command end

Command code	Function	OPERAND	Execution speed (us)	STEP	Page
			NC300		
END	Program ends	None	—	1	

Other commands

Command code	Function	OPERAND	Execution speed (us)	STEP	Page
			NC300		
NOP	Null action	None	-	1	
INV	Invert computing outcome	None	-	1	
P	Indicator	P0~P255	-	1	
I	Interrupt indicator	IX□□, IC□□, IR□□	-	1	

Note 1: Commands listed in the above table apply to the NC300 series.

Note: Value indicated by NC300 series' execution speed ( ) defines the execution speed of operand M1536~M4095.

2.2 Basic command description

Command	Function	Model
LD	Load A contact	NC300
		✓

Operand	X0~X33 X64~X511	Y0~Y27 Y64~Y511	M0~M3,071	A0~A511	T0~T255	C0~C77	D0~D1,535
	✓	✓	✓	✓	✓	✓	-

Command description

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

Example

Ladder diagram:

Command code:		Description:
LD	**X0**	Load X0's A contact
AND	X1	Serial connect X1's A contact
OUT	Y1	Drives coil Y1

Command	Function	Model
LDI	Load B contact	NC300
		✓

Operand	X0~X33 X64~X511	Y0~Y27 Y64~Y511	M0~M3,071	A0~A511	T0~T255	C0~C77	D0~D1,535
	✓	✓	✓	✓	✓	✓	-

Command description

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

Example

Ladder diagram:

Command code:		Description
LDI	**X0**	Load X0's B contact
AND	X1	Serial connect X1's A contact
OUT	Y1	Drives coil Y1

Command	Function	Model
AND	Serial connect A contact	NC300
		✓

Operand	X0~X33 X64~X511	Y0~Y27 Y64~Y511	M0~M3,071	A0~A511	T0~T255	C0~C77	D0~D1,535
	✓	✓	✓	✓	✓	✓	-

Command description

The AND command serial connects A contacts. It reads the current status of the given serial contacts and executes the AND operation on the acquired data together with the outcomes from previous logic operations and saves the outcome in a cumulative register.

Example

Ladder diagram:

Command code:		Description
LDI	X0	Load X0's B contact
AND	**X1**	Serial connect X0's A contact
OUT	Y1	Drives coil Y1

Command	Function	Model
ANI	Serial connect B contact	NC300
		✓

Operand	X0~X33 X64~X511	Y0~Y27 Y64~Y511	M0~M3,071	A0~A511	T0~T255	C0~C77	D0~D1,535
	✓	✓	✓	✓	✓	✓	-

Command description

The ANI command serial connects B contacts. It reads the current status of the given serial contacts and executes the AND operation on the acquired data together with the outcomes from previous logic operations and saves the outcome in a cumulative register.

Example

Ladder diagram:

Command code:		Description
LDI	X1	Load X1's A contact
AND	**X0**	Serial connect X0's B contact
OUT	Y1	Drives coil Y1

Command	Function	Model
OR	Parallel connect A contact	NC300 ✓

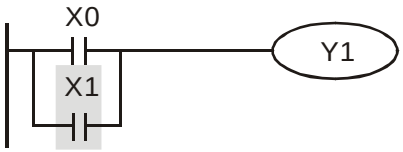
Operand	X0~X33 X64~X511	Y0~Y27 Y64~Y511	M0~M3,071	A0~A511	T0~T255	C0~C77	D0~D1,535
	✓	✓	✓	✓	✓	✓	-

Command description

The OR command parallel connects A contacts. It reads the current status of the given serial contacts and executes the OR operation on the acquired data together with the outcomes from previous logic operations and saves the outcome in a cumulative register.

Example

Ladder diagram:



Command code:		Description
LDI	X0	Load X0's A contact
OR	**X1**	Parallel connect X1's A contact
OUT	Y1	Drives coil Y1

Command	Function	Model
ORI	Parallel connect B contact	NC300 ✓

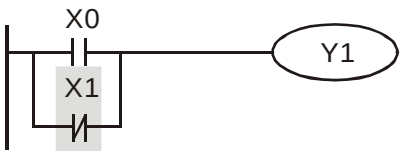
Operand	X0~X33 X64~X511	Y0~Y27 Y64~Y511	M0~M3,071	A0~A511	T0~T255	C0~C77	D0~D1,535
	✓	✓	✓	✓	✓	✓	-

Command description

The ORI command parallel connects B contacts. It reads the current status of the given serial contacts and executes OR operation on the acquired data together with the outcomes from previous logic operations and saves the outcome in a cumulative register.

Example

Ladder diagram:



Command code:		Description
LDI	X0	Load X0's A contact
ORI	**X1**	Parallel connect X1's B contact
OUT	Y1	

Command	Function	Model
ANB	Serial connect loop block	NC300 ✓

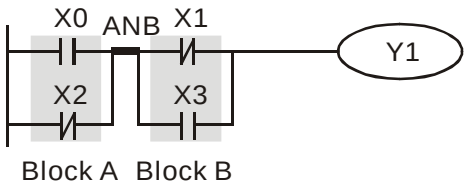
Operand	None

Command description

The ANB command executes the AND operation on previously saved logic outcome and current value in a cumulative register.

Example

Ladder diagram:



Command code:		Description
LDI	X0	Load X0's A contact
ORI	X2	Parallel connect X2's B contact
LDI	X1	Load X1's B contact
OR	X3	Parallel connect X3's A contact
ANB		Serial connect loop block
OUT	Y1	Drives coil Y1

Command	Function	Model
ORB	Parallel connect loop block	NC300 ✓

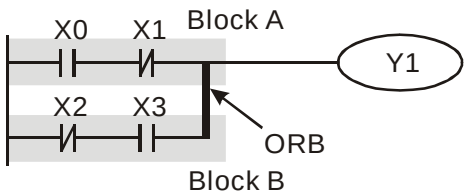
Operand	None

Command description

The ORB command executes the OR operation on previously saved logic outcomes and the current value in a cumulative register.

Example

Ladder diagram:



Command code:		Description
LD	X0	Load X0's A contact
ANI	X1	Parallel connect X1's B contact
LDI	X2	Load X2's B contact
AND	X3	Serial connect X3's A contact
ORB		Parallel connect loop block
OUT	Y1	Drives coil Y1

Command	Function	Model
MPS	Saves it in stack	NC300 ✓
Operand	None	

Command description

Saves the current value contained in the cumulative register in a stack. (Stack index increase by 1)

Command	Function	Model
MRD	Read stack (Stack index remain intact)	NC300 ✓
Operand	None	

Command description

Reads the current value contained in the stack and saves it in a cumulative register. (Stack index remain intact)

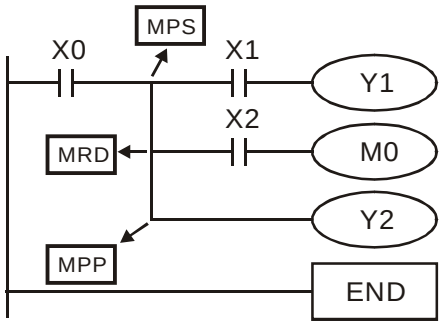
Command	Function	Model
MPP	Read stack	NC300 ✓
Operand	None	

Command description

Retrieves the last saved logic computing outcome and saves it in a cumulative register. (Stack index decrease by 1)

Example

Ladder diagram:



Command code:		Description
LD	X0	Load X0's A contact
MPS		Saves it in stack
AND	X1	Serial connect X1's A contact
OUT	Y1	Drives coil Y1
MRD		Read stack (Stack index remain intact)
AND	X2	Serial connect X2's A contact
OUT	M0	Drives coil Y2
MPP		Read stack
OUT	Y2	Drives coil Y2

END	Program ends

Command	Function	Model
OUT	Drives coil	NC300 ✓

Operand	X0~X33 X64~X511	Y0~Y27 Y64~Y511	M0~M1,215 M2,816~M3,071	A0~A511	T0~T255	C0~C77	D0~D1,535
	-	✓	✓	✓	-	-	-

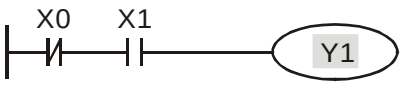
Command description

- Outputs the logic computing outcome before the OUT command to the given components.
- Coil contact action:

Computing outcome	OUT command		
	Coil	Contact	
		A contact (frequently open)	B contact (frequently close)
FALSE	OFF	Turns off	Turns on
TRUE	ON	Turns on	Turns off

Example

Ladder diagram:



Command code:		Description
LDI	X0	Load X0's B contact
AND	X1	Serial connect X1's A contact
OUT	**Y1**	Drives coil Y1

Command	Function	Model
SET	Fix actions (ON)	NC300 ✓

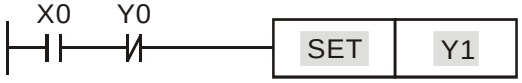
Operand	X0~X33 X64~X511	Y0~Y27 Y64~Y511	M0~M1,215 M2,816~M3,071	A0~A511	T0~T255	C0~C77	D0~D1,535
	-	✓	✓	✓	-	-	-

Command description

The SET command sets components assigned by it to ON and remains ON until being SET OFF by RST command.

Example

Ladder diagram:



Command code:		Description
LD	X0	Load X0's A contact
ANI	Y0	Serial connect Y0's B contact
SET	**Y1**	Fix Y1's action (ON)

Command	Function	Model
RST	Clear contacts or registers	NC300 ✓

Operand	X0~X33 X64~X511	Y0~Y27 Y64~Y511	M0~M1,215 M2,816~M3,071	A0~A511	T0~T255	C0~C77	D0~D1,118 D1,456~D1,535	V, Z
	-	✓	✓	✓	✓	✓	✓	✓

Command description

◆ See the table below for actions of components driven by RST command:

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

◆ Status of the components assigned by RST command remains intact if it was not executed.

Example

Ladder diagram:

Command code:		Description
LD	X0	Load X0's A contact
RST	**Y5**	Clear contact Y5

Command	Function	Model
TMR	16-bit timer	NC300 ✓

Operand	T-K	T0~T255, K0~K32,767
	T-D	T0~T255, D0~D1,536

Command description

◆ After a TMR command is executed, the timer assigned by it turns On and starts timing. The timer's contacts function as shown in table below when setup time is reached (timing value >= setup value):

NO (Normally Open) contact	Open
NC (Normally Close) contact	Close

Example

Ladder diagram:

Command code:		Description
LD	X0	Load X0's A contact
TMR	**T5K1000**	Timer T5 is set to K1000

Supplementary description

See the specification of all series model for the use of timer operand T.

Command	Function	Model
CNT	16-bit counter	NC300 ✓

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

Command description

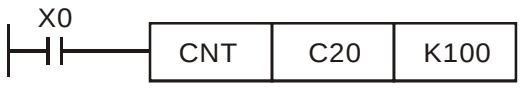
- When the CNT command changes from Off to On, the coil of the counter assigned by it switches from Off to On, leading to its counting value increasing by 1. The counter's contacts function as shown in table below when setup counts is reached (counting value >= setup value):

NO (Normally Open) contact	Open
NC (Normally Close) contact	close

- After the count settings is reached, the counter's contacts and counting values remain intact even when more counting pulse inputs are received. An RST command is required to restart counting or clear the value.

Example

Ladder diagram:



Command code:		Description
LD	X0	Load X0's A contact
CNT	**C20 K100**	Counter C20 is set to K100.

Command	Function	Model
DCNT	32-bit counter	NC300 ✓

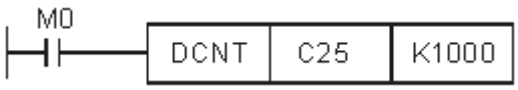
Operand	C-K	C64~C79, K-2,147,483,648~K2,147,483,647
	C-D	C64~C77, D0~D1,535

Command description

- The DCNT is a 32-bit counter for counters C64 ~ C79 initiation.
- General arithmetic counter C64~C77: When the DCNT command changes from Off to On, the counter's current value increases or decreases by 1 in setup mode to that of special M2832~2845.
- This counter counts when its specific high speed counting pulse input changes from Off to On. (C78, C79) are the high speed counter's counting pulse input end and (count up, increase count value by 1 and count down, decrease count value by 1) are counting action.
- When the DCNT command is OFF, its counters stop counting and the existing values remain. An RST C2XX command is required to clear the counting values and its contacts.

Example

Ladder diagram:



Command code:		Description
LD	M0	Load M0's A contact
DCNT	**C25**	Counter C25 is set to
	K1000	K1000.

Command	Function	Model
MC / MCR	Connection/disconnection of common serial contacts	NC300 ✓
Operand	N0~N7	

Command description

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

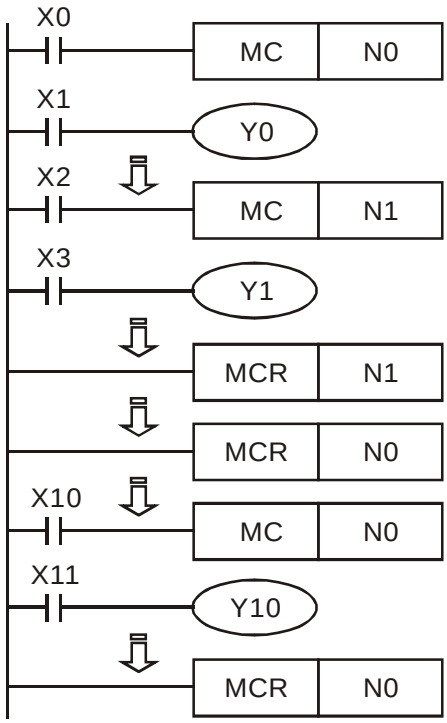
Types of commands	Description
Common timers	Reset timing value, coil OFF, contacts remain inactive
Counter	Coil OFF, counting values and contacts remain as the current status.
Coils driven by OUT command	All turned OFF
Components driven by SET and RST commands	Remain the current status
Application commands	Action remains intact. The FOR-NEXT nest loop keeps running for N times. Commands in the FOR-NEXT loop run in the same manner as that of commands between MC and MCR.

- The MCR command is the primary control end command and is placed at the bottom of the latter. No contact command is allowed before the MCR one.
- The MC-MCR primary control commands support nest structure up to 8 layers from N0 to N7. See example program shown below for details:

Example

Ladder diagram:

Command code:		Description
LD	X0	Load X0's A contact
MC	**N0**	N0 common serial contacts' connection in existence.
LD	X1	Load X1's A contact
OUT	Y0	Drive Y0 coil
:		



LD	X2	Load X2's A contact
MC	**N1**	Connection of N1 common serial contacts
LD	X3	Load X3's A contact
OUT	Y1	Drives coil Y1
:		
MCR	**N1**	Disconnection of N1 common serial contacts
:		
MCR	**N0**	Disconnection of N0 common serial contacts
:		
LD	X10	Load X10's A contact
MC	**N0**	Connection of N0 common serial contacts
LD	X11	Load X11's A contact
OUT	Y10	Drives coil Y10
MCR	**N0**	Disconnection of N0 common serial contacts

Command	Function	Model
LDP	Start of positive edge detection	NC300
		✓

Operand	X0~X33 X64~X511	Y0~Y27 Y64~Y511	M0~M3,071	A0~A511	T0~T255	C0~C77	D0~D1,535
	✓	✓	✓	✓	✓	✓	-

Command description

- The LDP command is used as the LD command but with a different function. It saves the current contents and saves the acquired contact's rising edge detection status in a cumulative register.

Example

Ladder diagram:

Command code:		Description
LDP	**X0**	X0: the positive edge detection

		operation starts
AND	X1	Serial connect X1's A contact
OUT	**Y1**	Drives coil Y1

Supplementary description

- See the function specifications of all series model for use range of individual operands.
- Before the PLC power is turned On, set the status of the rising edge contact On, then this rising edge contact is TRUE after power On.

Command	Function	Model
LDF	Start of negative edge detection	NC300 ✓

Operand	X0~X33 X64~X511	Y0~Y27 Y64~Y511	M0~M3,071	A0~A511	T0~T255	C0~C77	D0~D1,535
	✓	✓	✓	✓	✓	✓	-

Command description

- The LDF command is used as the LD command but with a different function. It saves the current contents and saves the acquired contact's falling edge detection status in a cumulative register.

Example

Ladder diagram:

Command code:		Description
LDF	**X0**	X0: the negative edge detection operation starts
AND	X1	Serial connect X1's A contact
OUT	Y1	Drives coil Y1

Command	Function	Model
ANDP	Positive edge detection serial connection	NC300 ✓

Operand	X0~X33 X64~X511	Y0~Y27 Y64~Y511	M0~M3,071	A0~A511	T0~T255	C0~C77	D0~D1,535
	✓	✓	✓	✓	✓	✓	-

Command description

◆ The ANDP command serial connects the contact's rising edge detection.

Example

Ladder diagram:

Command code:		Description
LD	X0	Load X0's A contact
ANDP	**X1**	X1 positive edge detection serial connection
OUT	Y1	Drives coil Y1

Command	Function	Model
ANDF	Negative edge detection serial connection	NC300 ✓

Operand	X0~X33 X64~X511	Y0~Y27 Y64~Y511	M0~M3,071	A0~A511	T0~T255	C0~C77	D0~D1,535
	✓	✓	✓	✓	✓	✓	-

Command description

◆ The ANDF command serial connects the contact's falling edge detection.

Example

Ladder diagram:

Command code:		Description
LD	X0	Load X0's A contact
ANDF	**X1**	X1: Negative edge detection serial connection
OUT	Y1	Drives coil Y1

Command	Function	Model
ORP	Positive edge detection parallel connection	NC300 ✓

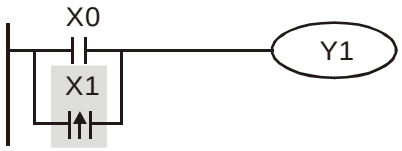
Operand	X0~X33 X64~X511	Y0~Y27 Y64~Y511	M0~M3,071	A0~A511	T0~T255	C0~C77	D0~D1,535
	✓	✓	✓	✓	✓	✓	-

Command description

◆ The ORP command parallel connects the contact's rising edge detection.

Example

Ladder diagram:



Command code:		Description
LD	X0	Load X0's A contact
ORP	**X1**	X1: Positive edge detection parallel connection
OUT	Y1	Drives coil Y1

Command	Function	Model
ORF	Negative edge detection parallel connection	NC300 ✓

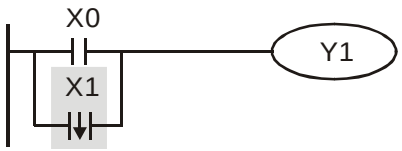
Operand	X0~X33 X64~X511	Y0~Y27 Y64~Y511	M0~M3,071	A0~A511	T0~T255	C0~C77	D0~D1,535
	✓	✓	✓	✓	✓	✓	-

Command description

◆ The ORF command parallel connects the contact's falling edge detection.

Example

Ladder diagram:



Command code:		Description
LD	X0	Load X0's A contact
ORF	**X1**	X1: Negative edge detection parallel connection
OUT	Y1	Drives coil Y1

Command	Function	Model
PLS	Upper differential output	NC300 ✓

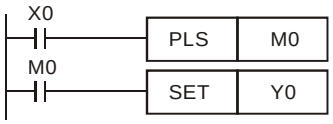
Operand	X0~X33 X64~X511	Y0~Y27 Y64~Y511	M0~M1,215 M2,816~M3,071	A0~A511	T0~T255	C0~C77	D0~D1,535
	-	✓	✓	-	-	-	-

Command description

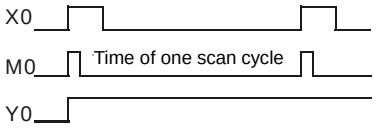
◆ Upper differential output command. In case X0=OFF→ON (positive edge triggering), the PLS command executes, M0 sends one pulse with a length of one cycle time.

Example

Ladder diagram:



Timing diagram:



Command code:		Description
LD	X0	Load X0's A contact
PLS	**M0**	M0 upper differential output
LD	M0	Load M0's A contact
SET	Y0	Y0 action retaining (ON)

Command	Function	Model
PLF	Lower differential output	NC300 ✓

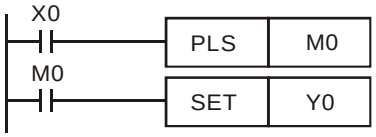
Operand	X0~X33 X64~X511	Y0~Y27 Y64~Y511	M0~M1,215 M2,816~M3,071	A0~A511	T0~T255	C0~C79	D0~D1,535
	-	✓	✓	-	-	-	-

Command description

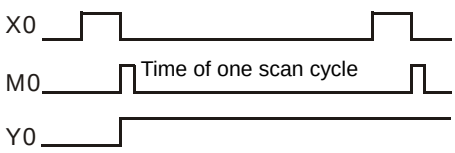
◆ Lower differential output command. In case X0= ON→OFF (negative edge triggering), the PLF command executes, M0 sends one pulse with a length of one cycle time.

Example

Ladder diagram:



Timing diagram:



Command code:		Description
LD	X0	Load X0's A contact
PLF	**M0**	M0 lower differential output
LD	M0	Load M0's A contact
SET	Y0	Y0 action retaining (ON)

Command	Function	Model
END	Program ends	NC300 ✓
Operand	None	

Command description

◆ A ladder or command program must end with an END command. The PLC scans and runs from address 0 to END command and then back to address 0 to repeat.

Command	Function	Model
NOP	No action	NC300 ✓
Operand	None	

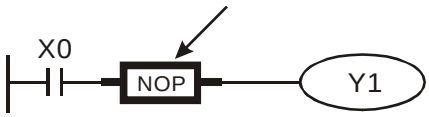
Command description

◆ The NOP command does not compute at all. After its execution, the logic computing outcome remains. If users desire to delete a statement in a program and keep the program size intact, then it can be replaced with a NOP command.

Example

Ladder diagram:

The NOP command is omitted from the ladder diagram.



Command code:		Description
LD	X0	Load X0's B contact
NOP		No action
OUT	Y1	Drives coil Y1

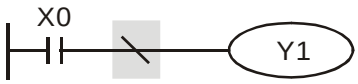
Command	Function	Model
INV	Invert the computing outcome	NC300 ✓
Operand	None	

Command description

◆ Invert the logic outcome before the INV command and saves it in a cumulative register.

Example

Ladder diagram:



Command code:		Description
LD	X0	Load X0's A contact
INV		Computing outcome invert
OUT	Y1	Drives coil Y1

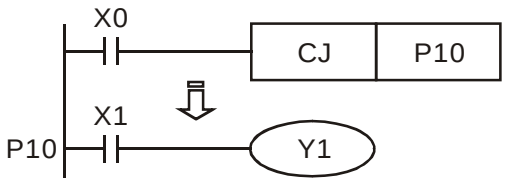
Command	Function	Model
P	Indicator	NC300 ✓
Operand	P0~P255	

Command description

◆ Indicator P is used to jump command API 00 CJ and subroutine calling command API 01 CALL. Indicator P can be used in random sequence. It cannot be repeated or an unexpected error may result.

Example

Ladder diagram:



Command code:		Description
LD	X0	Load X0's A contact
CJ	P10	Jump command CJ to command P10
:		
P10		Indicator P10
LD	X1	Load X1's A contact
OUT	Y1	Drives coil Y1

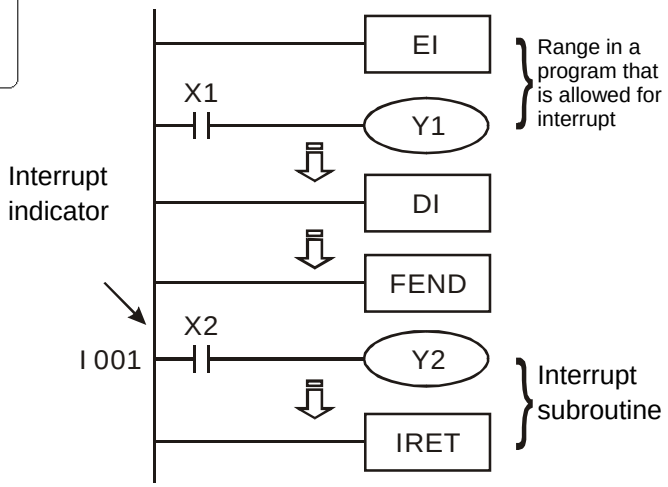
Command	Function	Model
I	Interrupt indicator	NC300 ✓
Operand	IX□□(IX00~IX07), IC□□(IC00~IC01), IR□□(IR00~IR23)	

Command description

◆ To interrupt a service program, use an interrupt indicator (I□□□) to tag its starting point and end and return with application command API 03 IRET. It must be used together with application command API 03 IRET, API 04 EI, and API 05 DI.

Example

Ladder diagram:



Command code:		Description
EI		Interrupt enabling
LD	**X1**	Load X1's A contact
OUT	Y1	Drives coil Y1
:		
DI		Interrupt disabling
:		
FEND		Main program

		stops running
I001		Interrupt indicator
LD	X2	Load X2's A contact
OUT	Y2	Drives coil Y2
:		
IRET		Interrupt return

Chapter 3: Types and Basic Uses of Application Commands

3.1 List of application commands

Type	API	Command code		Number of operand	Function	Model	STEPS		Page
		16-bit	32-bit			NC300	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	–	–	Main program end	✓	1	–	
	07	FOR	–	1	Nest loops start	✓	3	–	
	08	NEXT	–	–	Nest loops end	✓	1	–	
Transmission and compare	**09**	MOV	DMOV	2	Move data	✓	4	6	
	10	CML	DCML	2	Invert transmission	✓	4	5	
	11	BCD	DBCD	2	BIN→BCD conversion	✓	4	4	
	12	BIN	DBIN	2	BCD→BIN conversion	✓	4	4	
Arithmetic and logic computing	**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	BIN add one	✓	3	3	
	18	DEC	DDEC	1	BIN minus one	✓	3	3	
	19	WAND	DWAND	3	Logic AND operation	✓	6	8	
	20	WOR	DWOR	3	Logic OR operation	✓	6	8	
	21	WXOR	DWXOR	3	Logic XOR operation	✓	6	8	
	22	NEG	DNEG	1	Acquire negative value (Two's complement)	✓	3	3	
Rotate & shift	**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	Bit ON detect	✓	6	7	
	29	ANS	–	3	Alarm point output	✓	5	–	
	30	ANR	–	–	Alarm point reset	✓	1	–	

Type	API	Command code		Number of operand	Function	Model	STEPS		Page
		16-bit	32-bit			NC300	16-bit	32-bit	
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 commands	**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	–	
Contact type compare command	**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	Logic switch form		70	134	
Floating point computing	**58**		FADD	3	Binary floating point number addition	✓	–	7	
	59		FSUB	3	Binary floating point number subtraction	✓	–	7	
	60		FMUL	3	Binary floating point number multiplication	✓	–	7	
	61		FDIV	3	Binary floating point number division	✓	–	7	
	62		FCMP	3	Binary floating point number compare	✓	–	7	
	63		FINT	2	Binary floating point number 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: Valid for NC300 series

3.2 Syntax of application command

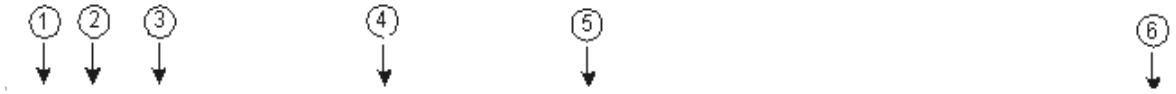
◆ An application command is composed of a command name and its operands.

Command name: The execution function

Operand: Target device the command is applied

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

◆ Illustration of application command format



API		MOV		(S) (D)	Move data	Model
09	D					NC300 ✓

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

Notes on the use of operands: For S and D operands running on Z devices only 16-bit commands are valid.
See specifications of each series model for valid device use rage.

16-bit command (4 STEP)

MOV	Continuous running type

32-bit command (6 STEP)

DMOV	Continuous running type

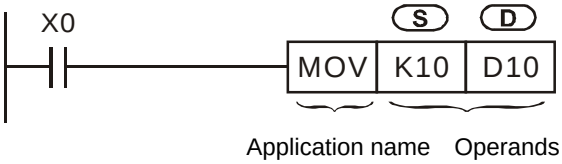
- Flag: None

1. Application command API code
2. The upper box indicates it's a 16-bit command. A box with a dotted border indicates that there is no 16-bit version that exists for this command.
   The lower box indicates it's a 32-bit command. A box with a dotted border indicates there is no 32-bit version that exists for this command. A command with the 32-bit version 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's function
6. The NC300 series MLC models that can use this command.
7. Steps taken by the 16-bit version and name of the continuous running version.
8. Steps taken by the 32-bit version and name of the continuous running version.
9. Flag signals relevant with this application command.
10. A box shadowed in gray and filled with the symbol '*' indicates that the device can use register V and Z indirectly
11. Notes on its uses
12. Devices marked with the symbol '*' are the ones that can be used by the operand.
13. Device name
14. Device type

◆ Input of application command

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

Each NC300 MLC application command 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 input the command API09, just type its name "MOV" in the MLCEditor. The application command has its own operand(s). Take MOV as the example.

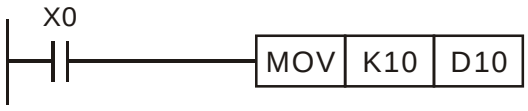
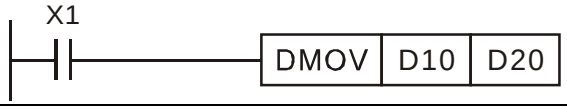


This command moves values in the (S) assigned operand to the target operand assigned by (D) where:

(S)	The source operand. Use (S1), (S2) and so on to represent multiple operands.
(D)	The target operand. Use (D1) , (D2) and so on to represent multiple operands.
In case the operand can assign the constant K/F or registers only, they are represented by (m), (m1), (m2), (n), (n1), and (n2).	

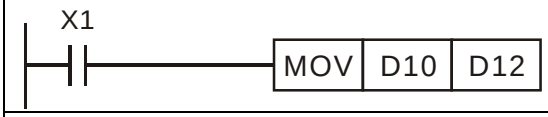
◆ Operand length (16-bit or 32-bit command)

Length of the operand's digital contents can be 16-bit or 32-bit. The 32-bit version of the command that can have either 16-bit or 32-bit operands is identified by prefixing letter "D" to the command name.

16-bit MOV command 	☞In case X0=On, operand K10 is sent to operand D10.
32-bit DMOV department 	☞In case X1=On, contents in (D11,D10) is sent to operand (D21,D20).

◆ Continuous running type

Some commands are executed continuously. They are the so called [continuous type] command.

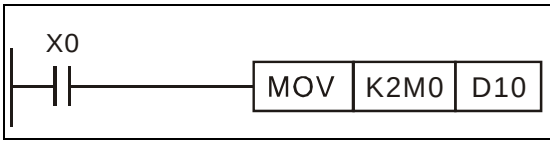
Continuous running type 	☞In case X1=On, the MOV command is executed every time for the scanning cycle and so is a continuous type command.
☞In case both condition points X0 and X1=Off, the command is ignored and contents contained in the target operand remains intact.	

◆ Target assigned by operand

1. Bit devices X, Y, M, and A may be combined into Word devices KnX, KnY, KnM, and KnA to keep data for computing in application command.
2. Data register D, timer T, counter C, and indirect assignment register V and Z are all targets assigned by general operands.
3. Data registers are 16-bit, one D register, in general. Users can assign two D registers in consecutive to form a 32-bit one.
4. If a 32-bit command's operand assigns D0 only, the 32-bit data register (D1, D0), where D0 for the lower bits and D1 for the upper, is still fully occupied. Timer, 16-bit counter, and C0~C63 follow the same rule.
5. When 32-bit counters C64~C77 are used as data register, they are valid for 32-bit command's operand only.

◆ Format of operand format

1. Devices X, Y, M and A can be used as single point On/Off only. They are the so called bit device.
2. Operands of 16-bit or 32-bit devices T, C, and D and registers V and Z are word device.
3. Users can define X, Y, M and A as word devices by prefixing them with Kn (where 'n' represents 4 bits and so a combination of K1~K4 is 16-bit and K1~K8 is 32 bit) for word device computation. Take K2M0 as the example. It represent 8-bit, M0~M7.



☞ In case X0=On, move values contained in M0~M7 to bit 0~7 of D10 and set remaining bits (8~15) to 0.

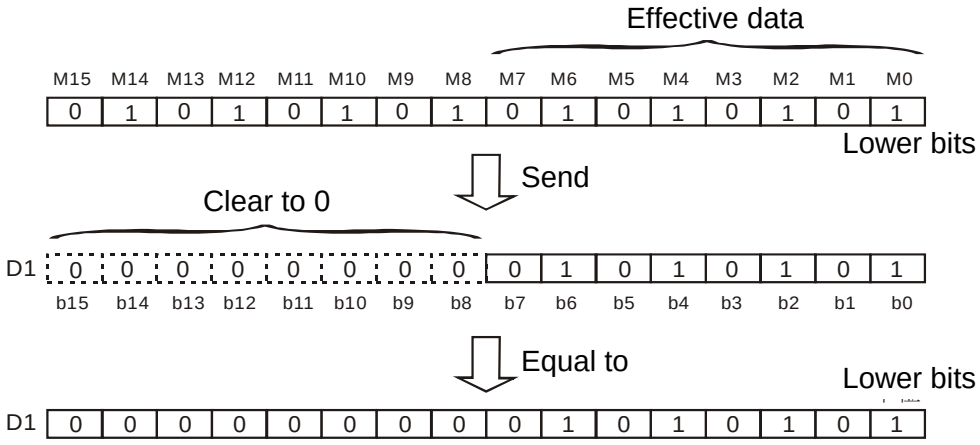
◆ How to deal with digital data contained in word devices combined by bit ones

16-bit command	
Values assigned by 16-bit command are in 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

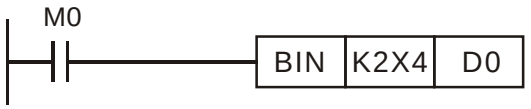
32-bit command	
Values assigned by 32-bit command are in range of K-2,147,483,648~K2,147,483,647	
Values contained in bit groups K1~K8 are:	
K1 (4 bits)	0~15
K2 (8 bits)	0~255
K3 (12 bits)	0~4,095
K4 (16 bits)	0~65,535
K5 (20 bits)	0~1,048,575
K6 (24 bits)	0~167,772,165
K7 (28 bits)	0~268,435,455
K8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

3.3 How numeric values are dealt with by application commands

- Devices X, Y, M, and A are called bit device as they can hold On/Off values only. Devices T, C, D, V, and Z are called word devices as they can contain numeric values. Bit device can be used by application command's operand in numeric value format by special declartion. The declaration is exercised by prefixing the bit device with a place value in form of 'Kn'.
- A 16-bit number can expressed by device K1~K4 while a 32-bit one by K1~K8. For 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 fill the upper bits with 0.
- For a 16-bit or 32-bit computation, if the command's register assigns to K1~K3 (or K4~K7) bit devices then all the un-used upper bits are filled with zero. This leads to the fact that they are treated as positive number computation in most cases.



☞Convert BCD binary value in X4~X13 to BIN format and save in D0.

- How to assign consecutive numbers

  Take data register D as the example. Its consecutive numbers are D0, D1, D2, D3, D4 and so on.

  For bit devices with assigned place number that consecutive numbers are shown in table below.

K1X0	K1X4	K1X10	K1X14……
K2Y0	K2Y10	K2Y20	Y2X30……
K3M0	K3M12	K3M24	K3M36…….
K4A0	K4A16	K4A32	K4A48…….

  The bit device numbers are shown in the table above. Do not skip or jump numbers to avoid confusion. When using K4Y0 in 32-bit computation, the upper 16 bits are treated as 0. For 32 bits data, please use K8Y0 instead.

  The NC300 series' MLC internal numeric computation is conducted in BIN integers. In case of division operation, e.g. 40÷3=13, the remainder is 1 in case of integer computing and decimal

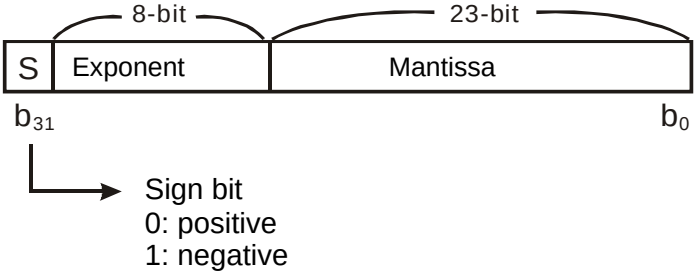
number for decimal computation.

Decimal number relavant application commands are shown in table below:

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)	

The expression of a binary floating point number

The NC300 series MLC expresses floating point number with 32-bit digit according to the IEEE754 standards. See below for its format:



Valid range of values are:

$$(-1)^S \times 2^{E-B} \times 1.M \quad \text{where B=127}$$

Range of values that can be expressed by a 32-bit floating point number is $\pm 2^{-126} \sim \pm 2^{+128}$ or $\pm 1.1755 \times 10^{-38} \sim \pm 3.4028 \times 10^{+38}$.

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

Step 1: Convert 23 to binary: 23.0=10111

Step 2: Normalize the binary 23 to $10111=1.0111 \times 2^4$, where 0111 is the mantissa and 4 is the exponent.

Step 3: Get the storage value of the exponent

$\because E-B=4 \rightarrow E-127=4 \therefore E=131=10000011_2$

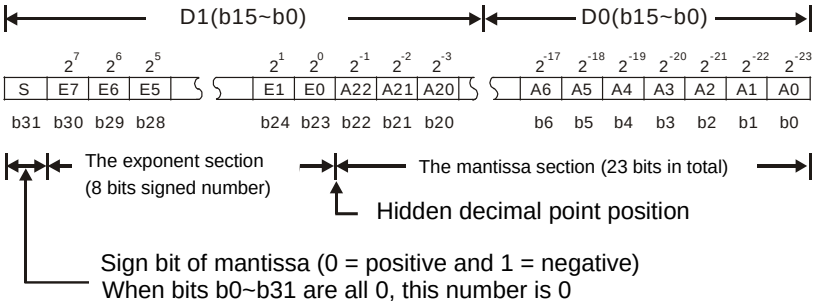
Step 4: Combine sign bit, exponent and mantissa into a floating point number.

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

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

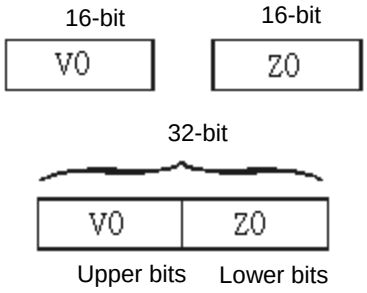
The floating point format -23.0 can be converted exactly the same as +23.0 except that the sign bit is 1.

The DVP-PLC employes two consecutive registers to form a 32-bit floating point number. Here register (D1, D0) is used to keep a binary floating point number as described below:


D1(b15~b0)
D0(b15~b0)
2^{7} 2^{6} 2^{5} 2^{1} 2^{0} 2^{-1} 2^{-2} 2^{-3} 2^{-17} 2^{-18} 2^{-19} 2^{-20} 2^{-21} 2^{-22} 2^{-23}
S E7 E6 E5 E1 E0 A22 A21 A20 A6 A5 A4 A3 A2 A1 A0
b31 b30 b29 b28 b24 b23 b22 b21 b20 b6 b5 b4 b3 b2 b1 b0
The exponent section
(8 bits signed number)
The mantissa section (23 bits in total)
Hidden decimal point position
Sign bit of mantissa (0 = positive and 1 = negative)
When bits b0~b31 are all 0, this number is 0

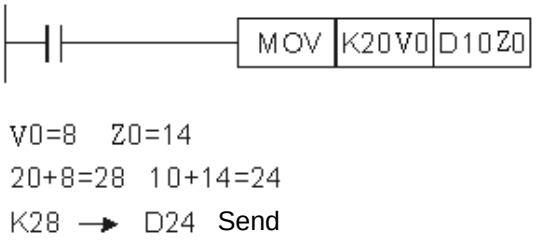
3.4 Use an indirect specified register V and Z to modify operand

All indirect specified registers are 16-bit ones. The NC300 series offer total 16 Z and V points.



The V and Z registers are all 16-bit and can be accessed for reading and writing in as desired.

If 32-bit length is required, then register V must be specified with register Z contained in V and cannot be used any more. (Please set value in Z to 0 with command MOVP when the machine is started.)
The V and Z combination used for 32-bit indirect specified register are: (V0, Z0), (V1, Z1), (V2, Z2) …and (V7, Z7).



As shown in figure to the left, contents of operands vary with those contained in V and Z. That is, they are modified by V and Z and thus indirectly specified.

In case of constants, e.g. V0=8, then K20V0 indicate constant K28 (20+8). With valid condition, constant K28 is sent to register D24.

Devices of the NC300 series that can be modified are: P, KnX, KnY, KnM, KnA, T, C, and D.

Devices that can be modified by V, Z are described above. V and Z cannot modify themselves and Kn (K4M0V0 is effective but not K0V0M0) in individual application commands. Gray shadowed operands in the subroutine forms are valid for modifying by V and Z.

Users can modify devices P, KnX, KnY, KnM, KnA, T, C, and D with 16-bit V and Z. To modify with 32-bit register, only V can be used.

3.5 Command index

◆ List of commands in alphabetic order

Type	API	Command code		Function	Model	Page
		16-bit	**32-bit**		NC300	
A	**13**	ADD	DADD	BIN addition	✓	
	29	ANS	–	Alarm point output	–	
	30	ANR	–	Alarm point reset	–	
	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→BCD conversion	✓	
	12	BIN	DBIN	BCD→BIN conversion	✓	
	28	BON	DBON	Bit ON detect	✓	
C	**00**	CJ	–	Conditional jump	✓	
	01	CALL	–	Call subroutines	✓	
	10	CML	DCML	Invert transmission	✓	
D	**05**	DI	–	Disable interruption	✓	
	16	DIV	DDIV	BIN division	✓	
	18	DEC	DDEC	BIN minus one	✓	
	26	DECO	–	Decoder	✓	
E	**04**	EI	–	Enable interruption	✓	
	27	ENCO	–	Encoder	✓	
F	**06**	FEND	–	Main program end	✓	
	07	FOR	–	Nest loops start	✓	
	58	–	FADD	Binary floating point number addition		
	59	–	FSUB	Binary floating point number subtraction		
	60	–	FMUL	Binary floating point number multiplication		
	61	–	FDIV	Binary floating point number division		
	62	–	FCMP	Binary floating point number compare		
	63	–	FINT	Binary floating point number convert to integer (truncated)		
	64	–	FDOT	Integer convert to binary decimal		
	62	–	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	Command code		Function	Model	Type
		16-bit	32-bit		NC300	
I	**03**	IRET	–	Return from interruption	✓	
	17	INC	DINC	BIN add one	✓	
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	Logic switch form	–	
W	**19**	WAND	DAND	Logic AND operation	✓	
	20	WOR	DOR	Logic OR operation	✓	
	21	WXOR	DXOR	Logic XOR operation	✓	
Z	**25**	ZRST	–	Zone clear	✓	

Chapter 4: Application Command API00~66

API		CJ		(S)	Conditional jump	Model
00						NC300
						✓

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

Notes on the use of operands:
The S operand can assign index P
The P ID can be modified by parameter V and Z
The S operand of the NC300 series model can assign parameter P0~P255

16-bit command (2 STEP)
CJ Continuous running type

32-bit command
- - - -

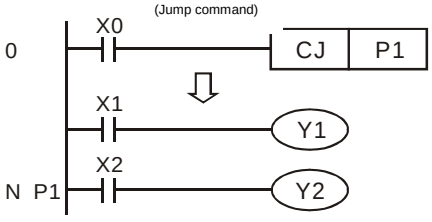
- Flag: None

Command description

- (S): Command indicator of a conditional jump.
- Use the CJ command to skip a section of statements in an MLC program to reduce scan time.
- Multiple CJ commands can point to one subject P. DO NOT point CJ and CALL commands to the same subject P as this may lead to a program error.
- Device actions when executing jump command: All commands given by users shall be executed when running jump commands.
  1. Status of device Y, M, and A remains intact before jump command execution.
  2. The 10ms and 100ms timer keeps on timing.
  3. The high speed counter C78 and C79 keeps on counting and the output contact functions normally.
  4. General application commands are ignored.
  5. Running application command API 53 DHSCS and API 54 DHSCR keeps on executing.

Example I

- In case X0=On, the program jumps from address 0 to N (the assigned label P1) for execution and ignore all statements in between.
- In case X0=Off, the program executes from address 0 downward in sequence as common ones and ignores the CJ command.

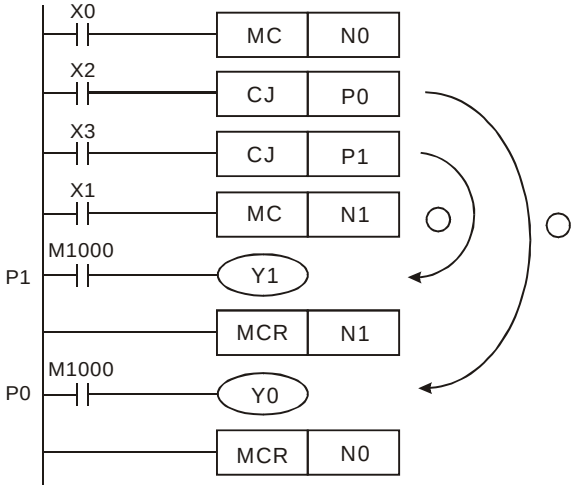


Example 2

◆ CJ command can be used between MC and MCR commands for 5 cases described below:

1. Outside of range MC~MCR.
2. From outside of the MC to inside of the MC as the loop below the Position shown in Figure P1 below.
3. Within an MC in the same layer N.
4. From inside an MC to outside of the MCR.
5. From within one range of MC~MCR to another range of MC~MCR.

◆ For an NC300 series model with an edition earlier than the MLC one (inclusive): When used between MC and MCR commands, the CJ command can be used only outside of MC~MR or within a pair of MC and MCR commands in the same N layer. The jumping from within one range of MC~MCR to another range of MC~MCR leads to a program error. That is, only Item 1 and 3 described above are valid while the remaining ones are invalid.



Example 3

◆ See the table below for status changes of each device.

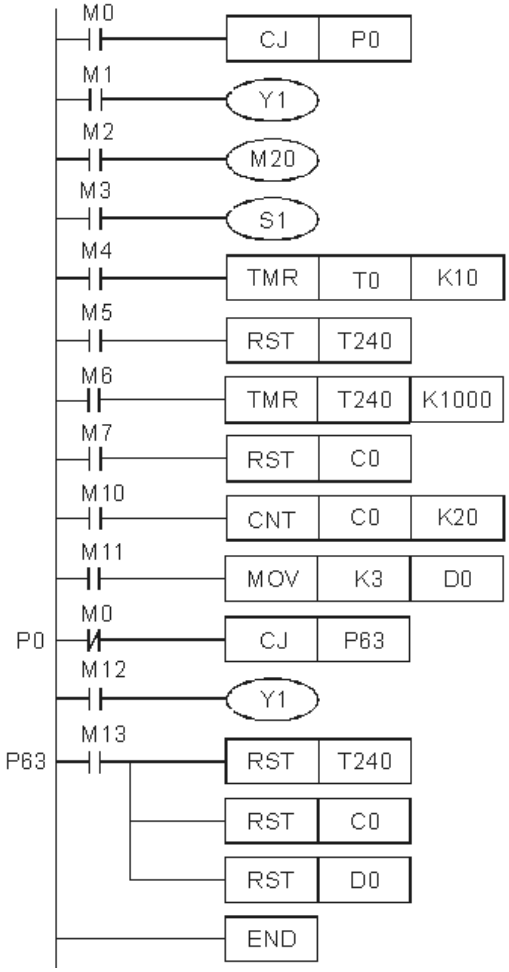
Device	Contact status before CJ command execution	Contact status during CJ command execution	Output coil status during CJ command execution
Y, M, A	M1, M2, M3 Off	M1, M2, M3 Off→On	Y1[Note1], M20, S1 Off
	M1, M2, M3 On	M1, M2, M3 On→Off	Y1[Note1], M20, S1 On
10, 100ms Timer	M4 Off	M4: Off→On	Timer T0 stops timing
	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	Timer C0 stops counting

	M7 Off, M10 On/Off trigger	M10 On/Off trigger	Timer C0 stops counting and remains. C0 keeps on counting after M0 is Off.
Application command	M11 Off	M11: Off→On	Application command stops executing.
	M11 On	M11: On→Off	Skipped application commands do not execute yet API 53 DHSCS and API 54 DHSCR remain executing.

Note1: Y1 is a dual output. It is controlled by M1 when M0 is OFF and by M12 when M0 is ON.

Nore2: When the activated high speed counter (C78 and C79) encounters a CJ command, both continue counting with the contact points remaining functioning.

◆ Y1 is a dual output. It is controlled by M1 when M0 is OFF and by M12 when M0 is ON.



API		CALL		(S)	Call subroutines	Model
01						NC300
						✓

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

Notes on the use of operands:
The S operand can assign index P
The P ID can be modified by parameter V and Z
The S operand of NC300 series model can assign parameter P0~P255

16-bit command (2 STEP)
CALL Continuous running type

32-bit command
- - - -

• Flag: None

Command description

◆ (S): Command indicator of calling subroutine

◆ The subroutine called by the CALL command follows command FEND.

◆ The number of index P called by command CALL cannot be the same as assigned by command CJ.

◆ In case only a CALL command is used, it can call a subroutine of the same index number as many times as desired.

◆ The CALL command can nest five calling layers inclduing the original one. (Subroutine called in the sixth layer does not run.)

API		SRET			END subroutines	Model
02						NC300
						✓

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

Notes on the use of operands:
No operand is required.
Connection point driven command does not follow.

16-bit command (1 STEP)
SRET Continuous running type

32-bit command
- - - -

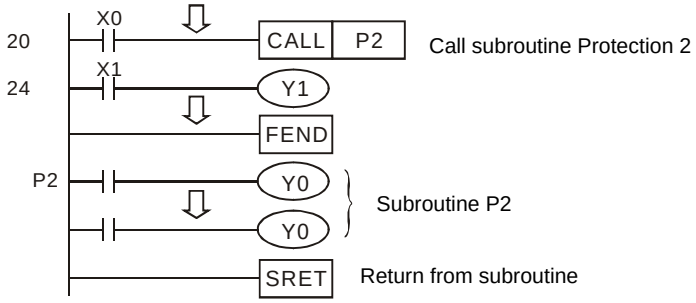
• Flag: None

Command description

◆ This command ends a subroutine. The SRET command ends a subroutine and goes back to the next statement after the CALL command in the calling program.

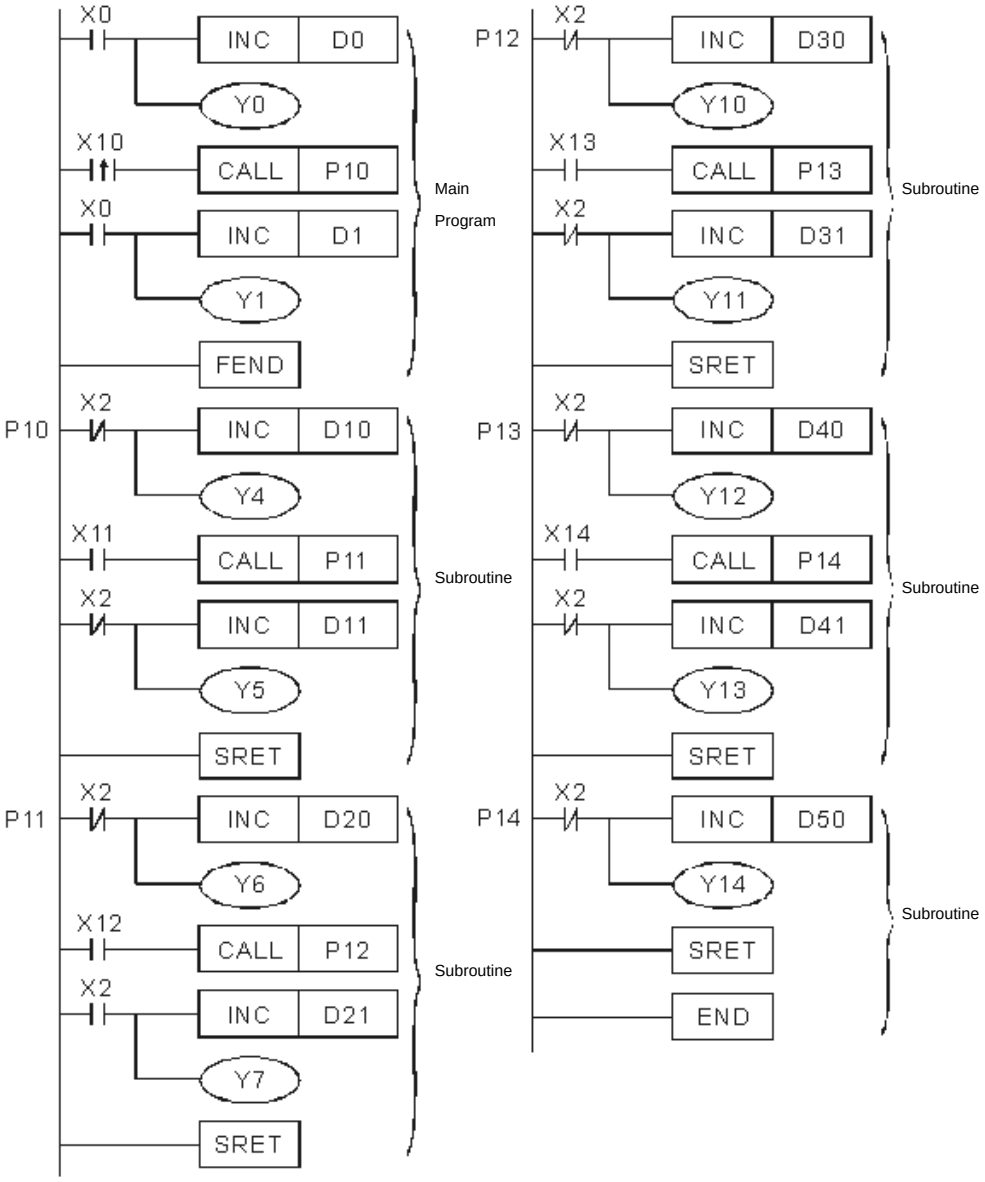
Example I

◆ In case X0 is ON, the CALL command is executed and the program jumps to the subroutine specified by parameter P2. Once the SRET command is executed the program returns to address 24 for execution.



Example 2

- When X10 switches positive from OFF to ON, the CALL P10 command is executed and the program jumps to the subroutine specified by parameter P10.
- In case X11 is ON, do command CALL P11 and the program jumps to the subroutine specified by parameter P11.
- In case X12 is ON, do command CALL P12 and the program jumps to the subroutine specified by parameter P12.
- In case X13 is ON, do command CALL P13 and the program jumps to the subroutine specified by parameter P13.
- In case X14 is ON, do command CALL P14 and the program jumps to the subroutine specified by parameter P14. Once the SRET command is executed, the program returns to previous subroutine P※ for execution.
- In subroutine P10, the SRET command returns the execution back to the main program.


X0
INC D0
Y0
X10
CALL P10
X0
INC D1
Y1
FEND
Main Program
P10
X2
INC D10
Y4
X11
CALL P11
X2
INC D11
Y5
SRET
Subroutine
P11
X2
INC D20
Y6
X12
CALL P12
X2
INC D21
Y7
SRET
Subroutine
P12
X2
INC D30
Y10
X13
CALL P13
X2
INC D31
Y11
SRET
Subroutine
P13
X2
INC D40
Y12
X14
CALL P14
X2
INC D41
Y13
SRET
Subroutine
P14
X2
INC D50
Y14
SRET
END
Subroutine

API		IRET			Return from interruption	Model
03						NC300
						✓

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

Notes on the use of operands:
No operand is required.
Connection point driven command does not follow.

16-bit command (1 STEP)
IRET Continuous running type

32-bit command
- - - -

- Flag: None

Command description

- Indicate to interrupt the subroutine.
- Interrupt the execution of service program and return to the main program by command IRET for the execution of the next statement after the interruption initiating statement.

API		EI			Enable interruption	Model
04						NC300
						✓

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

Notes on the use of operands:
No operand is required.
Connection point driven command does not follow.
Pulse width of interruption signal must be greater than 200 us.
See description below for ranges of I coding of each model.

16-bit command (1 STEP)
EI Continuous running type

32-bit command
- - - -

- Flag: See description below for M2864~M2873 and M2880~M2891. And refer to the following supplement for M2896~M2907.

API		DI			Disable interruption	Model
05						NC300
						✓

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

Notes on the use of operands:
No operand is required.
Connection point driven command does not follow.

16-bit command (1 STEP)
DI Continuous running type

32-bit command
- - - -

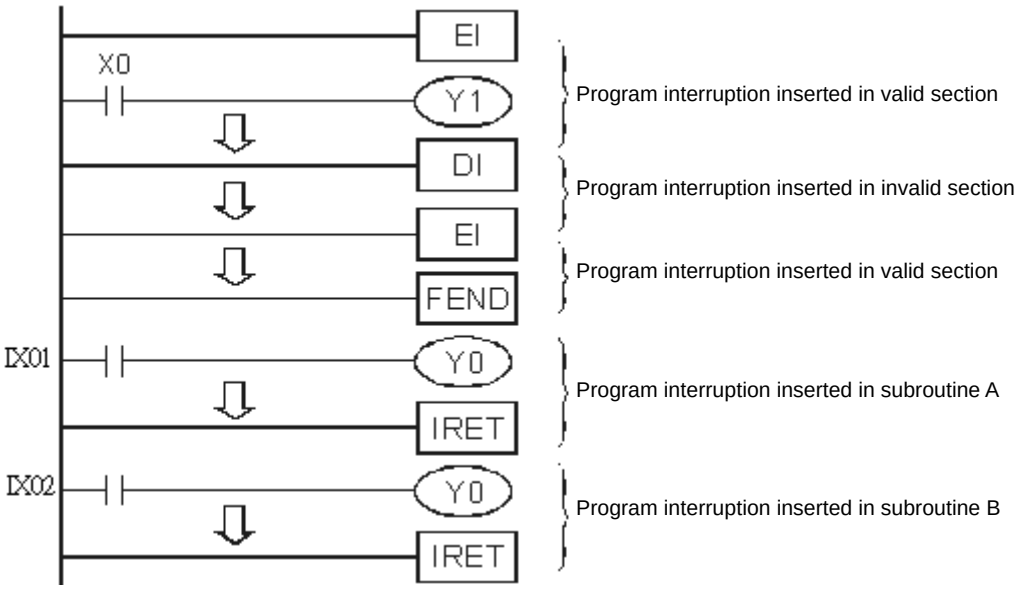
• Flag: None

Command description

- The EI command enables using interruption subroutines, including external and high speed counter interruptions.
- An interruption subroutine is valid between subroutine EI and DI. The DI command is not required if there is no interruption disabled section in the program.
- For NC300 series models that do not drive interruption disabled special relays M2864~M2873, M2880~M2891, and M2896~M2907, corresponding interruption requests are ignored even when placed in interruption enabled sections.
- The indicator I used by the interruption must appear after the FEND command.
- No other interruption command can be executed while another interruption command is running.
- When multiple interruption commands occur, the currently running one prevails and the one with a smaller index prioritizes when they occur at the same time.
- An interruption request in between DI~EI commands is kept in memory and can only be exectuted when it is within a permitted interruption range.
- In case an interruption index is used, do not use high speed counter driven by the same X input connection point again.
- If immediate I/O action is required during interruption processing, please insert a REF command in the program to update I/O status.

Example

- During the execution of MLC, when the program scans a section from EI to DI and X1=On or X2=On, then interruption subroutine A or B is executed. The subroutine returns to the main program for running downward after an IRET command was executed.



Supplementary description

◆ Coding of indicator I for NC300 series models:

1. OnBoard interruptions: including point (IX0□, X0), (IX0□, X1), (IX0□, X2), (IX0□, X3), (IX0□, X4), (IX0□, X5), (IX0□, X6), and (IX0□, X7).
2. The high speed counter counts to interruption IC00 and IC01. (Requires an API 32 DHSCS command for the generation of interruption signal.)
3. A total of 24 Remote I/O interruptions from IR00 to IR23, where IR00~IR02 match Input X to Remote X256~258. Every three IR interruptions match the first three INPUT X in each Remote I/O. That is, 24 interruptions (IR00~23) match 8 Remote I/O modules respectively.
4. The interruption indicator I services on the basis of first-in-first-out and is without any priorities.

◆ NC300 series models interruption indicator insertion banned flag:

Flag	Function description
M2864	I00 interruption input On Board X0 (1 = enabled and 0 = disabled) can be used for program coding or setup.
M2865	I01 interruption input On Board X1 (Mask external interrupts)
M2866	I02 interruption input On Board X2
M2867	I03 interruption input On Board X3
M2868	I04 interruption input On Board X4
M2869	I05 interruption input On Board X5
M2870	I06 interruption input On Board X6
M2871	I07 interruption input On Board X7
M2872	I08 interruption input Hardware Counter 0 (command only)
M2873	I09 interruption input Hardware Counter 1

M2880	I10 interruption input Remote Card 1 X0
M2881	I11 interruption input Remote Card 1 X1
M2882	I12 interruption input Remote Card 1 X2
M2883	I13 interruption input Remote Card 2 X0
M2884	I14 interruption input Remote Card 2 X1
M2885	I15 interruption input Remote Card 2 X2
M2886	I16 interruption input Remote Card 3 X0
M2887	I17 interruption input Remote Card 3 X1
M2888	I18 interruption input Remote Card 3 X2
M2889	I19 interruption input Remote Card 4 X0
M2890	I20 interruption input Remote Card 4 X1
M2891	I21 interruption input Remote Card 4 X2
M2896	I22 interruption input Remote Card 5 X0
M2897	I23 interruption input Remote Card 5 X1
M2898	I24 interruption input Remote Card 5 X2
M2899	I25 interruption input Remote Card 6 X0
M2900	I26 interruption input Remote Card 6 X1
M2901	I27 interruption input Remote Card 6 X2
M2902	I28 interruption input Remote Card 7 X0
M2903	I29 interruption input Remote Card 7 X1
M2904	I30 interruption input Remote Card 7 X2
M2905	I31 interruption input Remote Card 8 X0
M2906	I32 interruption input Remote Card 8 X1
M2907	I33 interruption input Remote Card 8 X2

API		FEND			Main program end	Model
06						NC300
						✓

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

Notes on the use of operands:
No operand is required.
Connection point driven command does not follow.

16-bit command (1 STEP)
FEND Continuous running type

32-bit command
- - - -

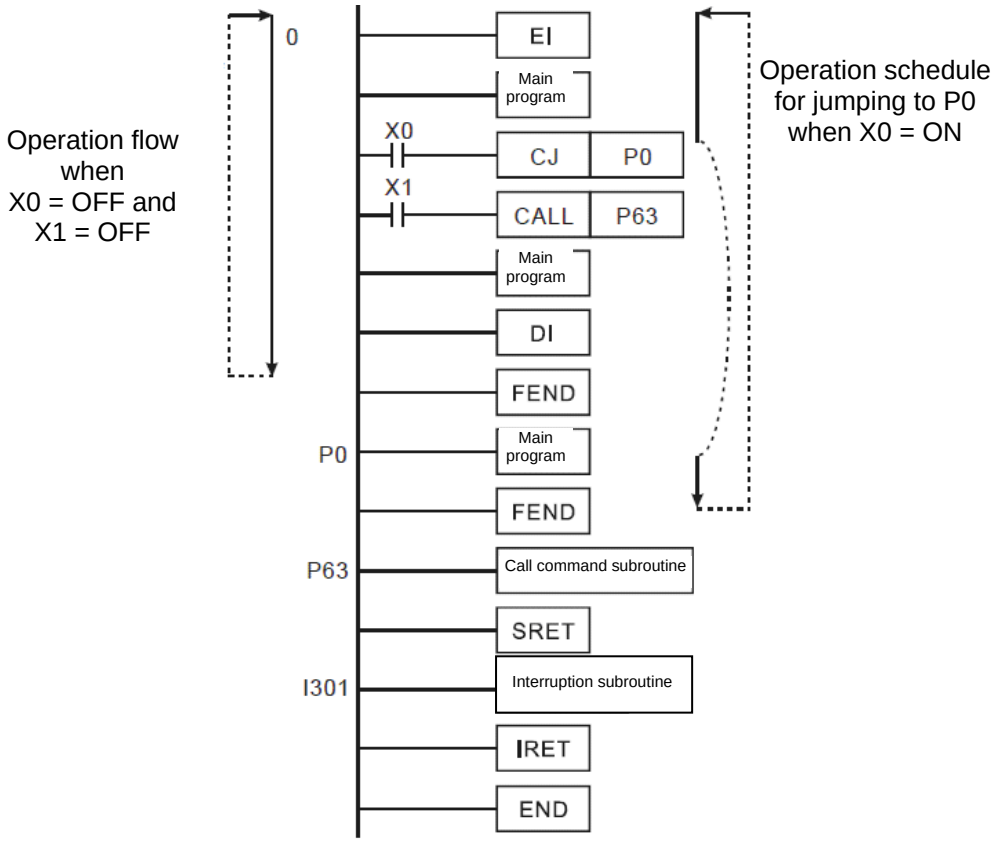
• Flag: None

Command description

- This command indicates the end of the main program. It functions the same as END does during MLC execution.
- The subroutine called by the CALL command must be coded after FEND command and ended by a SRET command. The interruption program must be coded after FEND command and ended by an IRET command.
- In case multiple FEND commands are used, please insert the subroutine and interruption service program between the last pair of FEND and END commands.

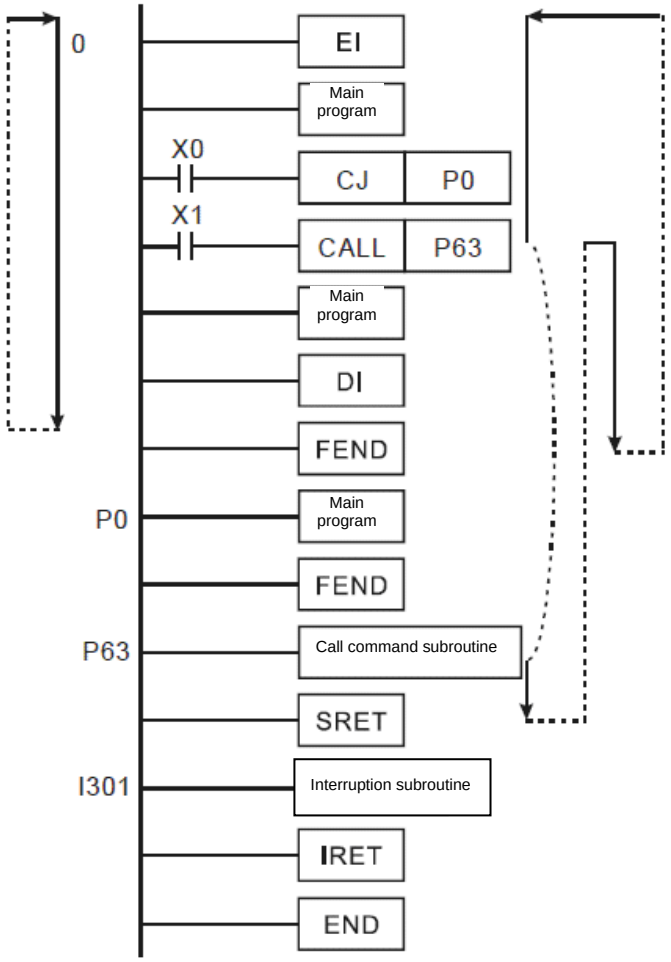
- After the execution of a CALL command, running a SRET command before its counterpart FEND will lead to a program error.
- After the execution of a FOR command, running a NEXT command before its counterpart FEND will lead to a program error.

Operation flow of the CJ command



Operation flow of the CJ command

Operation flow when X0 = OFF and X1 = OFF



Operation schedule when X0 = OFF and X1 = ON

API		FOR		(S)	Nest loops start	Model
07						NC300 ✓

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

Notes on the use of operands:
Connection point driven command does not follow.
See specification of each model for valid device use rage.

16-bit command (3 STEP)
FOR Continuous running type

32-bit command
- - - -

- Flag: None

Command description

◆ (S): Number of times the loop is to be executed.

API		NEXT			Nest loops end	Model
08						NC300
						✓

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

Notes on the use of operands:
No operand is required.
Connection point driven command does not follow.

16-bit command (1 STEP)
NEXT Continuous running type

32-bit command
- - - -

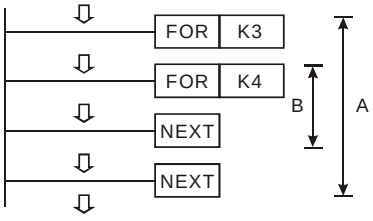
• Flag: None

Command description

- The FOR command specifies the number of times a FOR~NEXT loop is to be executed. After the loop is ended, the program continues running from the statement next to the NEXT command.
- The valid range of repetition times is indicated by N=K1~K32,767. Any value of N less than K1 will be rounded to K1.
- Users can use a CJ command to exit the FOR~NEXT loop.
- Possible errors are:
  1. The NEXT command precedes the FOR one.
  2. The FOR command lacks an accompanying NEXT one.
  3. The FEND or END command follows by a NEXT one.
  4. The FOR~NEXT commands do not pair.
- The FOR~NEXT loops can nest for up to 5 layers. Please note the more nestings there are, the more time is required for MLC scanning.

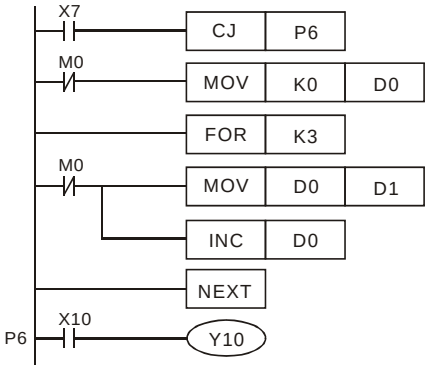
Example I

- Program A continues running the subroutine next to the last NEXT command after being repeated 3 times. During each execution of program A, program B is executed for 4 times. That is, program B runs for 12 times in total.



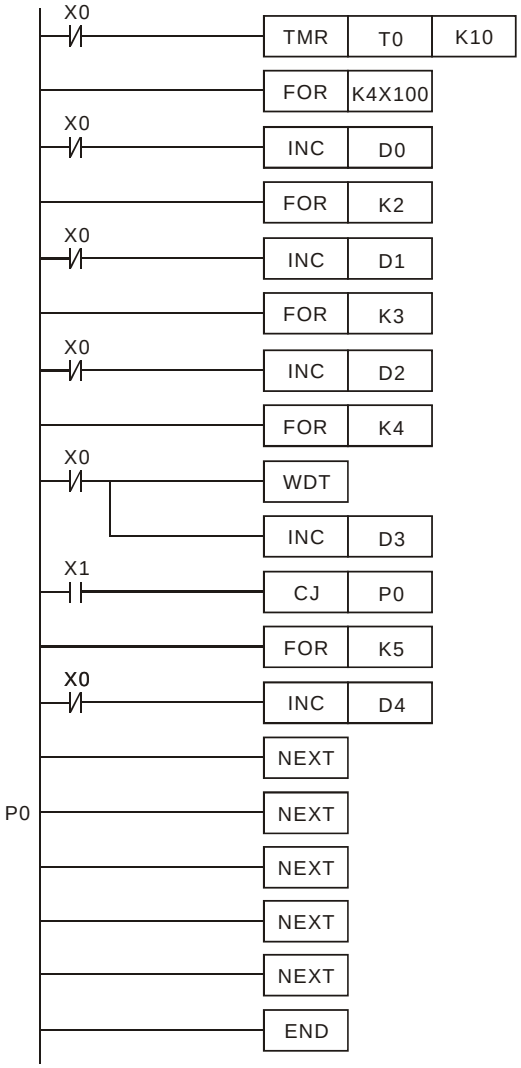
Example 2

- In case X7=Off, the MLC executes program in between commands FOR-NEXT pair. In case X7=On, the CJ command jumps to statement P6 and ignores program between FOR and NEXT commands.



Example 3

◆ Users can use a CJ command to skip a FOR~NEXT program loop. The most inner layer of the FOR~NEXT loop is ignored by jumping to statement P0 with command CJ when X1 = On.



API		MOV		(S) (D)	Move data	Model
09	**D**					NC300 ✓

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

Notes on the use of operands:

For S and D operands running on Z devices only 16-bit commands are valid.

See specification of each NC300 model for valid device use rage.

16-bit command (4 STEP)
MOV Continuous running type

32-bit command (6 STEP)
DMOV Continuous running type

- Flag: None

Command description

- (S): Source of data. (D): Destination of data to be moved to.
- This command moves data contained in (S) to (D). Contents contained in (D) remain intact.
- For 32-bit outoput from computing outcomes (e.g. application command MUL) and current values of the 32-bit device's high speed counter, it moves them with the DMOV command.

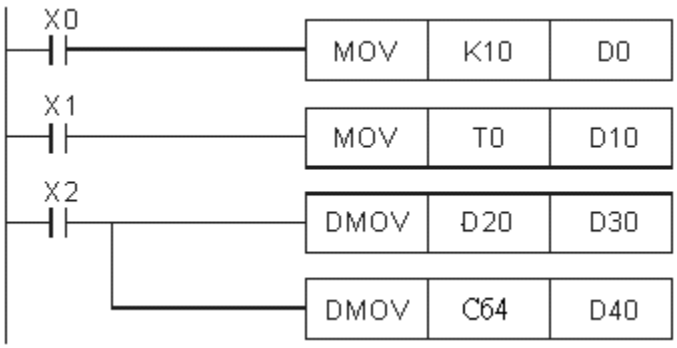
Example I

- Move 16-bit data with the MOV command.

  1. In case X0=off, contents of D10 remain intact. In case X0=On, it moves data contained in K10 to register D10.
  2. In case X1=off, contents of D10 remain intact. In case X1=On, it moves the current value of T0 to register D10.

- Move 32-bit data with the DMOV command.

  In case X2=off, contents of (D31, D30) and (D41, D40) remain intact. In case X2=On, it moves the current values of (D21, D20) to register (D31, D30) and that of C64 to register (D41, D40).



API		CML		(S) (D)	Invert transmission	Model
10	**D**					NC300 ✓

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

Notes on the use of operands:
For S and D operands running on Z devices only 16-bit commands are valid.
See specification of each model for valid device use rage.

16-bit command (4 STEP)
CML Continuous running type

32-bit command (5 STEP)
DCML Continuous running type

- Flag: None

Command description

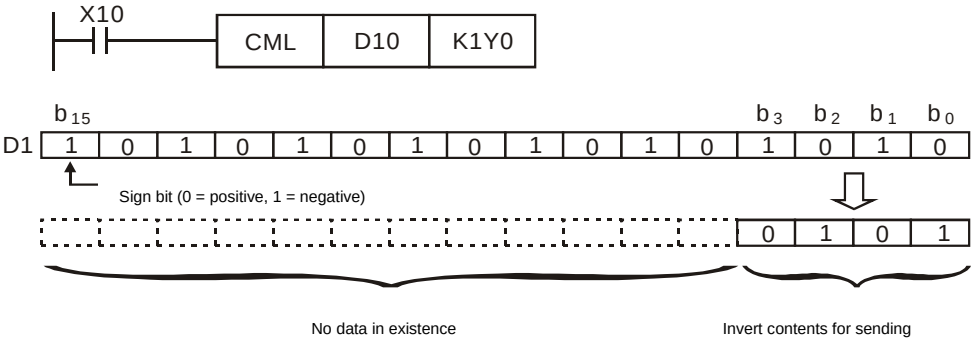
◆ (S): Source of data to be transmitted. (D): Target device of transmission.

◆ Invert (0→1, 1→0) data contained in (S) and send to (D). Automatically invert constant K to BIN value.

Example I

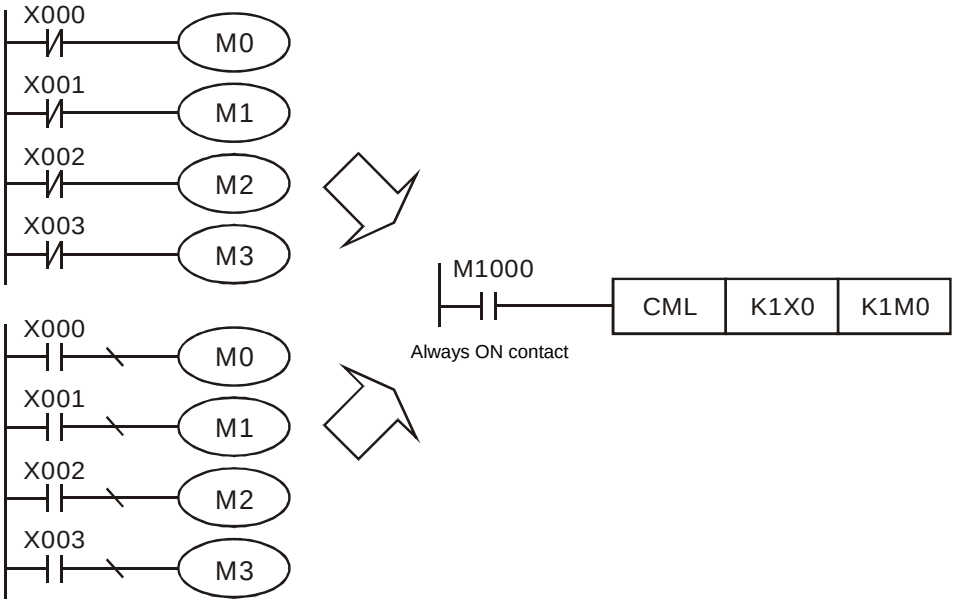
◆ Use this command for inverted output.

◆ In case X10=On, invert D1's b0~b3 contents and send to Y0~Y3.



Example 2

◆ The circuit shown to the left in the figure below can be presented with a CML command as presented in the circuit to the right.



API		BCD		S D	BIN → BCD conversion	Model
11	**D**					NC300 ✓

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

Notes on the use of operands:
For S and D operands running on Z devices only 16-bit commands are valid.
See specification of each model for valid device use rage.
BCD: Binary-Coded Decimal

16-bit command (4 STEP)
BCD Continuous running type

32-bit command (4 STEP)
DBCD Continuous running type

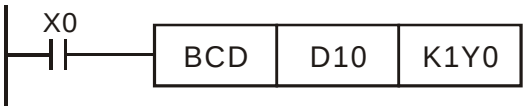
- Flag: M2930 (computing error)
  D1467 (error code)

Command description

- Do BCD conversion for BIN data contained in (S) and save in (D).
- In case the BCD conversion output exceeds K0~K9,999 and M2930=On, the D1467 register records error code 0001.
- In case the DBCD conversion output exceeds K0~K99,999,999 and M2930=On, the D1467 register records error code 0001.
- The INC and DEC commands used by MLC's arithmetic operations are executed with values in BIN format. To see values displayed in decimal format, Users need to convert values in BIN format to BCD one with the BCD conversion.

Example

- In case X0=On, values in D10 are converted from BIN to BCD format and the digit in ones of the outcome is stored in bit elements K4Y0 (Y0~Y3).



If D10=001E (Hex)=0030 (decimal), then the outcome of execution is Y0~Y3=0000(BIN).

API		BIN		(S) (D)	BCD → BIN conversion	Model
12	D					NC300
						✓

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

Notes on the use of operands:
For S and D operands running on Z devices only 16-bit commands are valid.
See specification of each model for valid device use rage.

16-bit command (4 STEP)
BIN Continuous running type

32-bit command (4 STEP)
DBIN Continuous running type

- Flag: No display error for now

Command description

- (S): Source of data. (D): Outcome of coversion.
- Do BIN conversion for source data in (S) (BCD: 0~9,999) and save in (D).
- Valid range of source data in (S) is BCD (0~9,999) and DBCD (0~99,999,999).
- This command is not required for constant K and H as they are converted into BIN format automatically.

Example

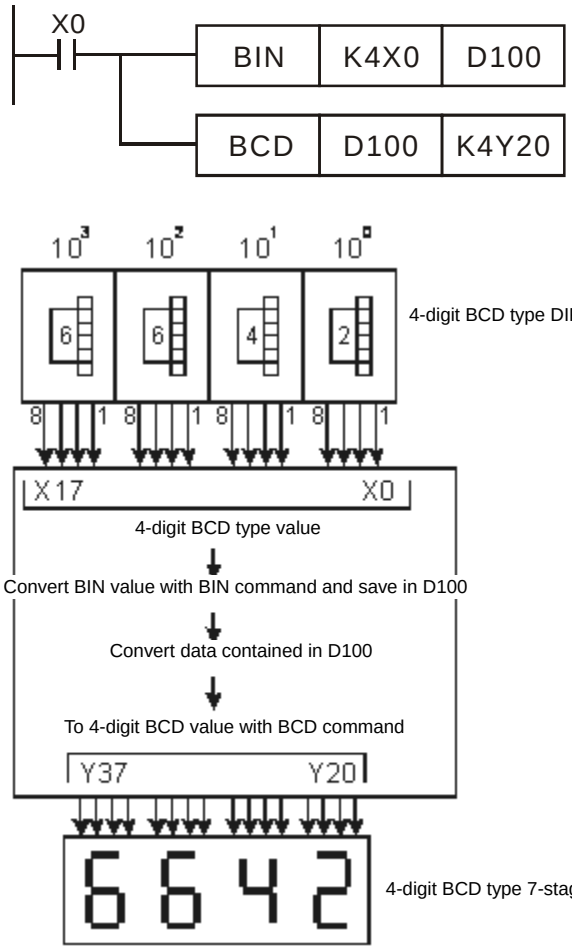
- In case X0=On, BCD format value in K1X0 is converted to a BIN format one and save in D10.

Supplementary description

- How to use BCD and BIN commands:

1. In case the MLC is to read the value of a BCD type DIP switch, a BIN command is required to convert the readings into BIN format and store in MLC.
2. To convert data stored in MLC for displaying in an external BCD type 7-stage display, a BCD command is required to convert internal data in to BCD format and send to the 7-stage display.

3. In case X0=On, convert BCD value in K4X0 BCD to BIN format and send to D100, the convert BIN value in D100 to BCD format and send to K4Y20.



API		ADD		(S1) (S2) (D)	BIN addition	Model
13	**D**					NC300 ✓

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

Notes on the use of operands:
For S_1, S_2 and D operands running on Z devices only 16-bit commands are valid.
See specification of each model for valid device use rage.

16-bit command (6 STEP)
ADD Continuous running type

32-bit command (8 STEP)
DADD Continuous running type

- Flag: M2824 Zero flag
  M2825 Borrow flag
  M2826 Carry flag
  Please refer to supplementary description shown below.

Command description

- ◆ (S1): Summand. (S2): Addend. (D): Sum.
- ◆ Add values contained in data sources (S1) and (S2) in BIN format and

save the sum in (D).

- The very first bit of each data represnts it's positive (0) or negative (1). This enables algebraic addition operations like 3+(-9)=-6.
- Addition relevant flag changes.

  16-bit BIN addition:

  1. In case the addition outcome is 0, the zero flag M2824 sets On.
  2. In case the addition outcome is less than –32,768, the borrow flag M2825 sets On.
  3. In case the addition outcome is greater than 32,767 the carry flag M2826 sets On.

  32-bit BIN addition:

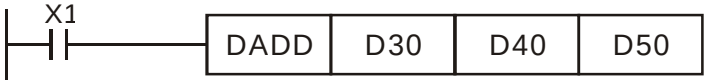
  1. In case the addition outcome is 0, the zero flag M2824 sets On.
  2. In case the addition outcome is less than –2,147,483,648, the borrow flag M2825 sets On.
  3. In case the addition outcome is greater than 2,147,483,647 the carry flag M2826 sets On.

Example I

- 16-bit BIN addition: In case X0=On, the sum of summand D0 and addend D10 is kept in D20.

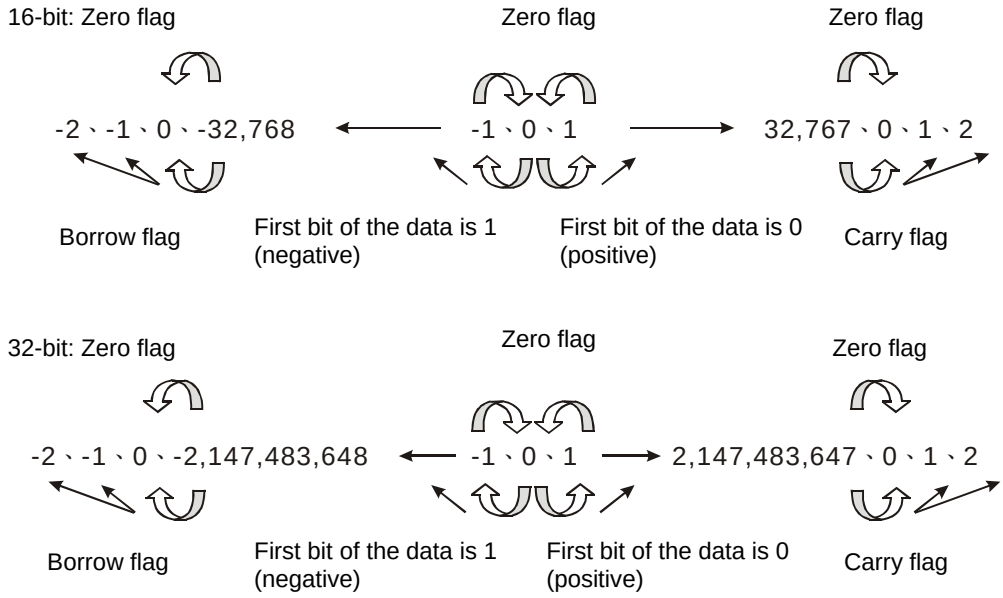
Example 2

- 32-bit BIN addition: in case X1=On, the sum of summand (D31, D30) and addend (D41, D40) is kept in (D51, D50) where D30, D40, and D50 are the lower 16-bit data while D31, D41, and D51 are the upper one.



Supplementary description

- Relations between flag changes and the positive/negative property of a number:



API		SUB		(S1) (S2) (D)	BIN subtraction	Model
14	**D**					NC300 ✓

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

Notes on the use of operands:
For S_1, S_2 and D operands running on Z devices only 16-bit commands are valid.
See specification of each model for valid device use rage.

16-bit command (6 STEP)
SUB Continuous running type

32-bit command (8 STEP)
DSUB Continuous running type

- Flag: M2824 Zero flag
  M2825 Borrow flag
  M2826 Carry flag
  Please refer to supplementary description shown in ADD operation.

Command description

- (S1): Minuend. (S2): Subtrahend. (D): Difference.
- Subtract values contained in data sources (S1) and (S2) in BIN format and save the sum in (D).
- The very first bit of each data represnts it's positive (0) or negative (1). This enables algebraic subtraction operation.
- Subtraction relevant flag changes.

  16-bit BIN subtraction:

  1. In case the addition outcome is 0, the zero flag M2824 sets On.
  2. In case the addition outcome is less than –32,768, the borrow flag M2825 sets On.

3. In case the addition outcome is greater than 32,767 the carry flag M2826 sets On.

32-bit BIN subtraction:

1. In case the addition outcome is 0, the zero flag M2824 sets On.
2. In case the addition outcome is less than –2,147,483,648, the borrow flag M2825 sets On.
3. In case the addition outcome is greater than 2,147,483,647 the carry flag M2826 sets On.

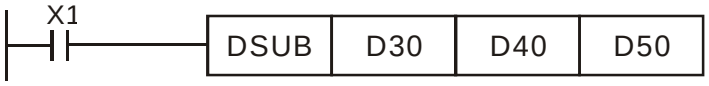
◆ Please refer to the supplementary description of ADD operation shown in pages above for relations between flag changes and the positive/negative property of a number.

Example I

◆ 16-bit BIN subtraction: In case X0=On, the remnant of D0 less D10 is kept in D20.

Example 2

◆ 32-bit BIN subtraction: in case X1=On, the remnant of (D31, D30) less (D41, D40) is kept in (D51, D50) where D30, D40, and D50 are the lower 16-bit data while D31, D41, and D51 are the upper one.



API		MUL		(S1) (S2) (D)	BIN multiplication	Model
15	**D**					NC300 ✓

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

Notes on the use of operands:
For S_1 and S_2 operands running on Z devices only 16-bit commands are valid.
For D operands running on V devices only 16-bit commands is valid.
16-bit command D operand takes consecutive 2 points.
32-bit command D operand takes consecutive 4 points.

16-bit command (6 STEP)
MUL Continuous running type

32-bit command (8 STEP)
DMUL Continuous running type

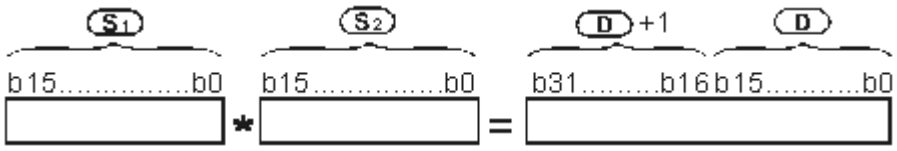
- Flag: None

Command description

◆ (S1): Multiplicand. (S2): Multiplier. (D): Product.

◆ Multiply values contained in data source (S1) and (S2) in binary integer multiplication and save its product in (D). Please pay special attention to the sign bit of data contained in (S1), (S2) and (D) during 16-bit and 32-bit operation.

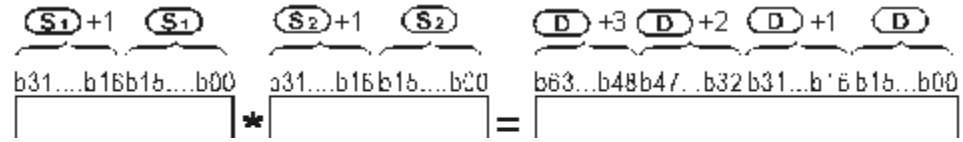
◆ 16-bit BIN multiplication operation:



b15 is a sign bit	b15 is a sign bit	b31 is a sign bit (the b15 of D+1)
b15=0, S1 is positive b15=1, S1 is negative	b15=0, S_2 is positive b15=1, S_2 is negative	b31=0, D(D+1) is positive b31=1, D(D+1) is negative

In case (D) is a bit device, users may assign K1~K4 into a 16-bit number taking consecutive 2 groups.

◆ 32-bit BIN multiplication operation:

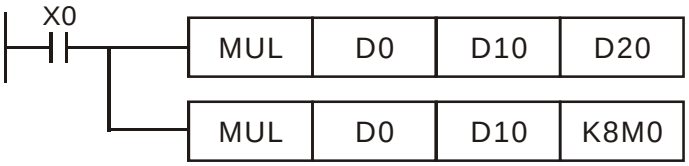


b31 is a sign bit	b31 is a sign bit	b63 is a sign bit (the b15 of D+1)
b31=0, S_1(S_1+1) is positive b31=1, S_1(S_1+1) is negative	b31=0, S_2(S_2+1) is positive b31=1, S_2(S_2+1) is negative	b63=0, D(D+1)(D+2)(D+3) is positive b31=1, D(D+1)(D+2)(D+3) is negative

In case (D) is a bit device, users may assign K1~K8 into a 32-bit number for keeping 32-bit data.

Example

◆ The product of 16-bit D0 and 16-bit D10 is a 32-bit value with the upper 16-bit kept in D21 and lower 16-bit in D20. The number's positive or negative property is determined by Off/On status of its first bit.



API		DIV		(S1) (S2) (D)	BIN division	Model
16	D					NC300 ✓

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

Notes on the use of operands:
For S_1 and S_2 operands running on Z devices only 16-bit commands are valid.
For D operands running on V devices only 16-bit commands is valid.
16-bit command D operand takes consecutive 2 points.
32-bit command D operand takes consecutive 4 points.

16-bit command (7 STEP)
DIV Continuous running type

32-bit command (13 STEP)
DDIV Continuous running type

- Flag: None

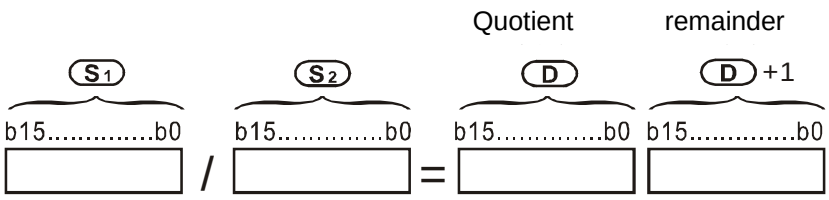
Command description

◆ (S1): Dividend. (S2): Divisor. (D): Quotient and remainder.

◆ Values contained in (S1) divided by that of (S2) in binary integer division and saves its quotient and remainder in (D). Please pay special attention to the sign bit of data contained in (S1), (S2) and (D) during 16-bit and 32-bit operation.

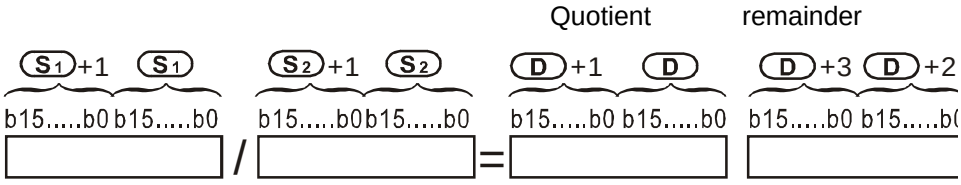
◆ This command is ignored in case the divisor is 0. In case M2930=On, an error code 0002 (Hex) will be kept in D1467.

16-bit BIN division operation:



In case (D) is a bit device, users may assign K1~K4 into a 16-bit number taking consecutive 2 groups for quotient and remainder.

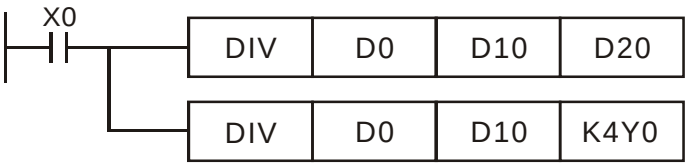
32-bit BIN division operation:



In case (D) is a bit device, users may assign K1~K8 into a 32-bit number for quotient but not remainder.

Example

◆ In case X0=On, the quotient and remainder of D0 divided by divisor D10 is kept in D20 and D21 respectively. Both numbers' positive or negative property is determined by Off/On status of their first bit respectively.



API		INC		(D)	BIN add one	Model
17	D					NC300 ✓

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

Notes on the use of operands:

For D operands running on Z devices only 16-bit commands are valid.

16-bit command (3 STEP)
INC Continuous running type

32-bit command (3 STEP)
DINC Continuous running type

• Flag: None

Command description

◆ (D): The target device.

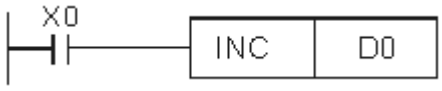
◆ This command increases the value contained in specified device (D) by 1 every time it is scanned by the program.

◆ For 16-bit operation the sum of 32,767 and 1 is -32,768 and the sum of 2,147,483,647 and 1 is -2,147,483,648 for 32-bit operation.

◆ The outcome of this command does not change flag M2824~M2826.

Example

◆ In case X0=Off→On, value of D0 increase by 1 automatically.



API		DEC		(D)	BIN less one	Model
18	D					NC300 ✓

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

Notes on the use of operands:

For D operands running on F devices only 16-bit commands are valid.

16-bit command (3 STEP)
DEC Continuous running type

32-bit command (3 STEP)
DDEC Continuous running type

- Flag: None

Command description

- (D): The target device.
- This command decrease value contained in specified device (D) by 1 every time it is scanned by the program.
- For 16-bit operation the remnant of -32,768 less 1 is 32,767 and the remnant of -2,147,483,648 less 1 is 2,147,483,647.
- The outcome of this command does not change flag M2824~M2826.

Example

- In case X0=Off→On, value of D0 decrease by 1 automatically.

API		WAND		(S1) (S2) (D)	Logic AND operation	Model
19	D					NC300 ✓

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

Notes on the use of operands:

For S_1, S_2 and D operands running on Z devices only 16-bit commands are valid.

16-bit command (6 STEP)
WAND Continuous running type

32-bit command (8 STEP)
DWAND Continuous running type

- Flag: None

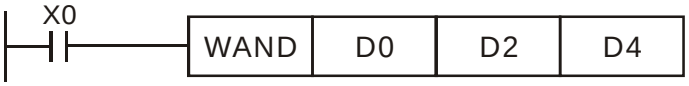
Command description

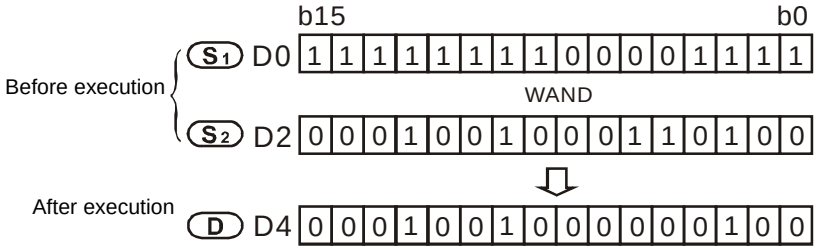
- (S1): Source data device 1.(S2): Source data device 2.(D): Operation outcome.
- Do logic AND operation on data sources (S1) and (S2) and save its outcome in (D).
- The logic AND operation turns an outcome of 0 when either of its two

values is 0.

Example I

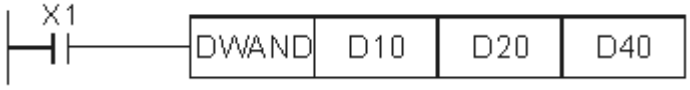
◆ In case X0=On, do WAND (logic AND) operation on 16-bit D0 and D2 and save the outcome in D4.

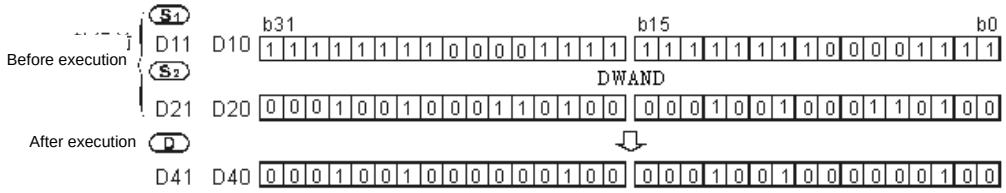




Example 2

◆ In case X1=On, do DAND (logic AND) operation on 32-bit (D11, D10) and (D21, D20) and save the outcome in (D41, D40).





API		WOR		S1 S2 D	Logic OR operation	Model
20	**D**					NC300 ✓

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

Notes on the use of operands:
For S_1, S_2 and D operands running on Z devices only 16-bit commands are valid.

16-bit command (6 STEP)
WOR Continuous running type

32-bit command (8 STEP)
DWOR Continuous running type

- Flag: None

Command description

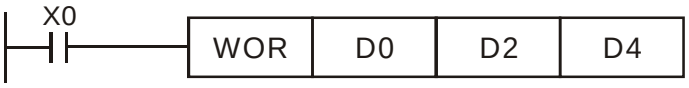
◆ S1: Source data device 1. S2: Source data device 2. D: Operation outcome.

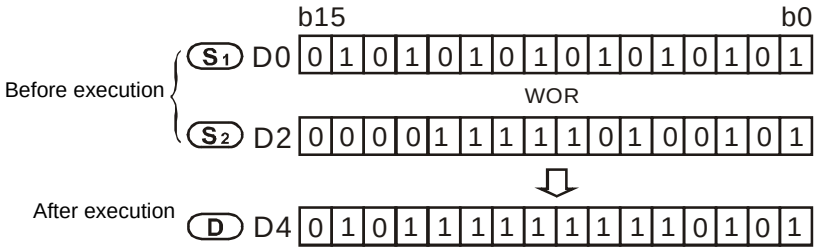
◆ Do logic OR operation on data sources S1 and S2 and save its outcome in D.

◆ The logic OR operation turns an outcome of 1 when either of its two values is 1.

Example I

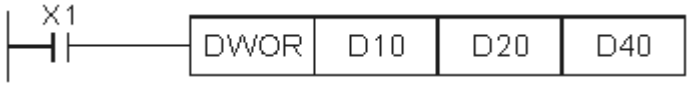
◆ In case X0=On, do WOR (logic OR) operation on 16-bit D0 and D2 and save the outcome in D4.

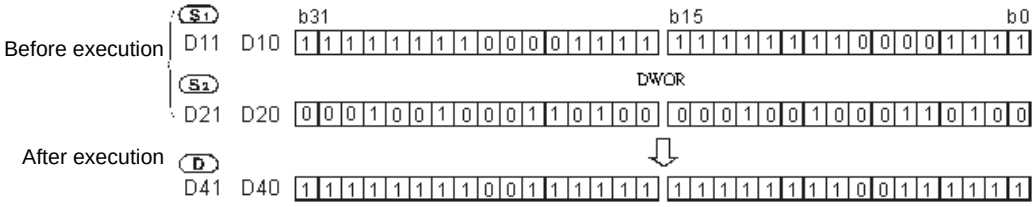




Example 2

◆ In case X1=On, do DOR (logic OR) operation on 32-bit (D11, D10) and (D21, D20) and save the outcome in (D41, D40).





API		WXOR		S_1 S_2 D	Logic exclusive (XOR) operation	Model
21	D					NC300 ✓

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

Notes on the use of operands:
For S_1, S_2 and D operands running on Z devices only 16-bit commands are valid.

16-bit command (7 STEP)
WXOR Continuous running type

32-bit command (8 STEP)
DWXOR Continuous running type

- Flag: None

Command description

◆ S_1: Source data device 1. S_2: Source data device 2. D: Operation outcome.

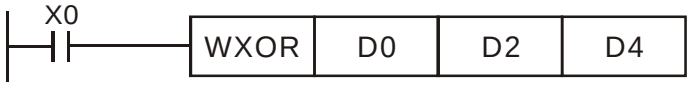
◆ Do logic XOR operation on data sources S_1 and S_2 and save its

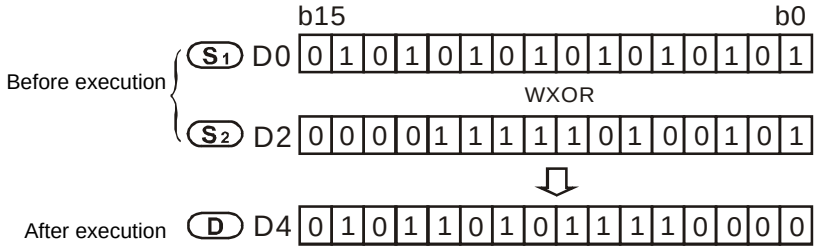
outcome in (D).

◆ The logic XOR operation turns an outcome of 0 when both of its two values are the same and 1 when its two values differ from each other.

Example I

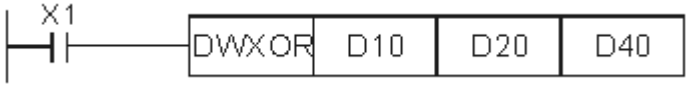
◆ In case X0=On, do WXOR (logic XOR) operation on 16-bit D0 and D2 and save the outcome in D4.

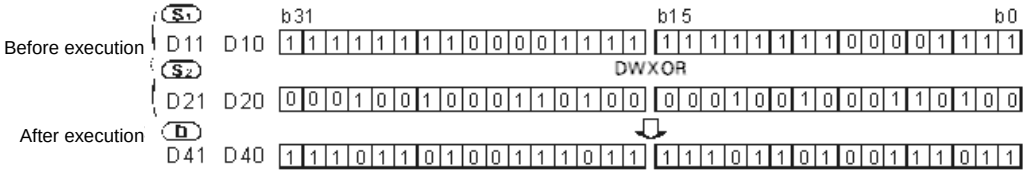




Example 2

◆ In case X1=On, do DXOR (logic XOR) operation on 32-bit (D11, D10) and (D21, D20) and save the outcome in (D41, D40).





API		NEG		(D)	Two's complement	Model
22	**D**					NC300 ✓

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

Notes on the use of operands:

For D operands running on Z devices only 16-bit commands are valid.

16-bit command (3 STEP)
NEG Continuous running type

32-bit command (3 STEP)
DNEG Continuous running type

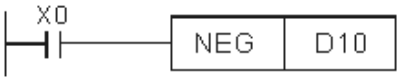
- Flag: None

Command description

◆ (D): The device where two's complement is required. This command converts negative BIN value into the absolute one.

Example I

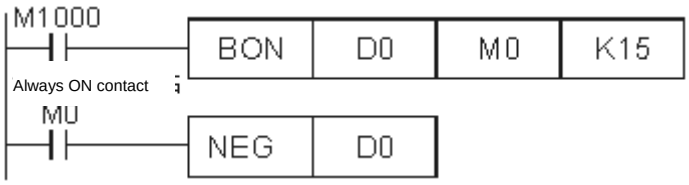
◆ In case X0=Off→On, invert (0→1, 1→0) every bit of digit contained in D10 and increase it by 1 to save in register D10.



Example 2

◆ To get the absolute value of a negative number.

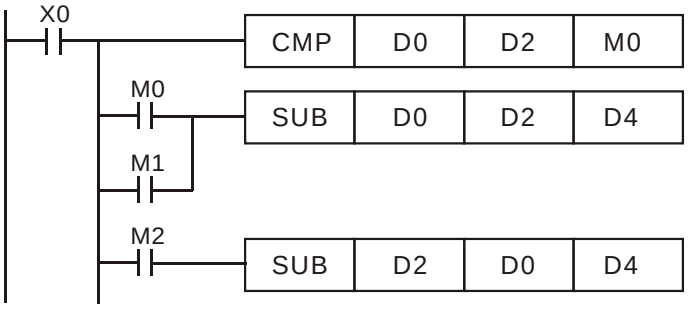
1. The 15th bit of digit in D0 is “1” and M0=On. (D0 is negative)
2. In case M0=On, use NEG command to take two’s complement and get the absolute value of D0.



Example 3

◆ Get the absolute value of the remnant of a subtraction operation when X0=On:

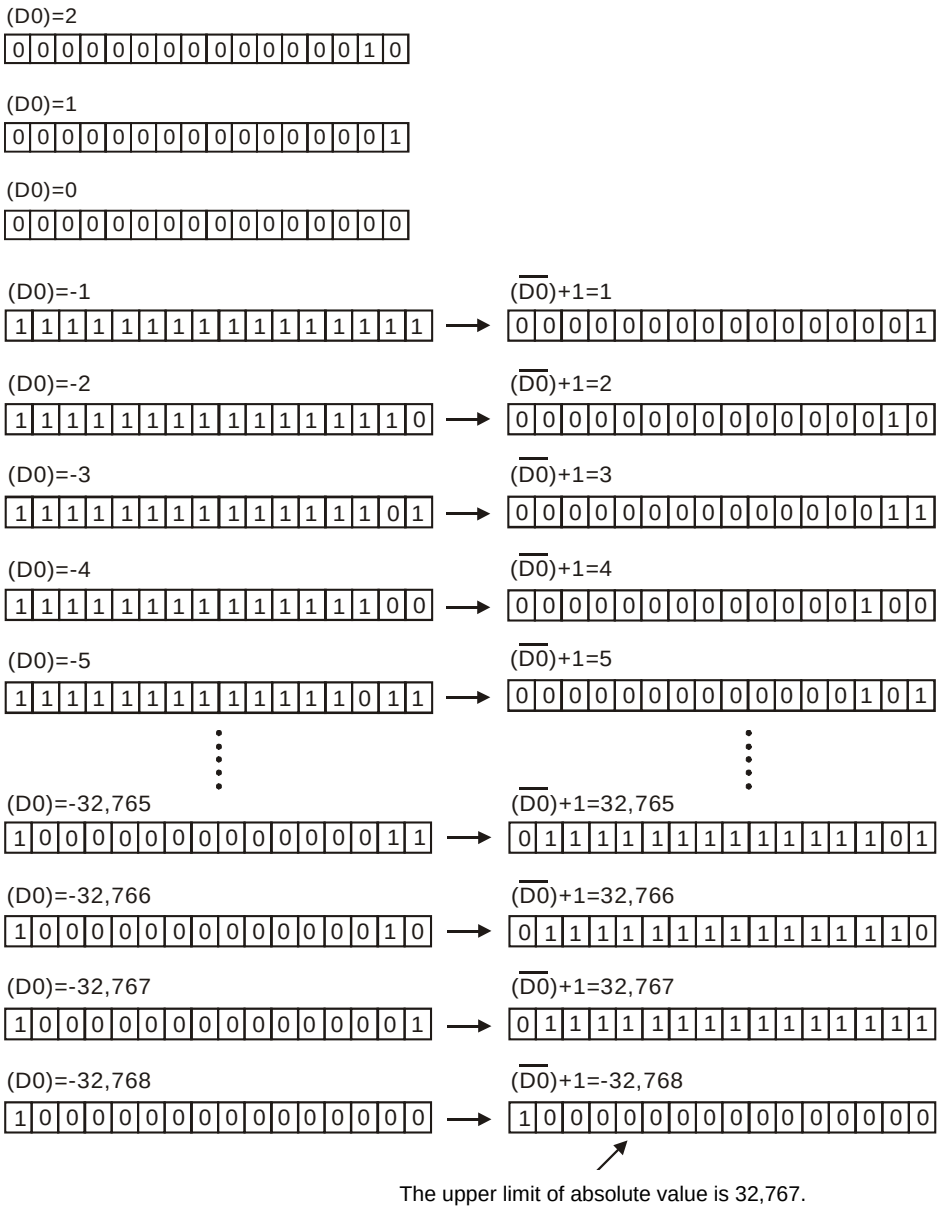
1. If D0>D2 and M0=On.
2. If D0=D2 and M1=On.
3. If D0<D2 and M2=On.
4. This ensures value in D4 to be positive.



Supplementary description

◆ Presentation of negative number and absolute value

1. Digit in a register is either positive or negative according to value of its leftest bit: “0” indicate a positive number and “1” negative.
2. Users may convert a negative number into its absolute value with NEG command (API 22).



API		ROR		D n	Rotate right	Model
23	**D**					NC300 ✓

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

Notes on the use of operands:

For D operands running on Z devices only 16-bit commands are valid.

For D operands assigned to KnY, KnM and KnA only K4 (16-bit) and K8 (32-bit) are valid.

Valid value for n operand is K1~K16 (16-bit) and K1~K32 (32-bit)

16-bit command (4 STEP)
ROR Continuous running type

32-bit command (4 STEP)
DROR Continuous running type

- Flag: M2826 carry flag

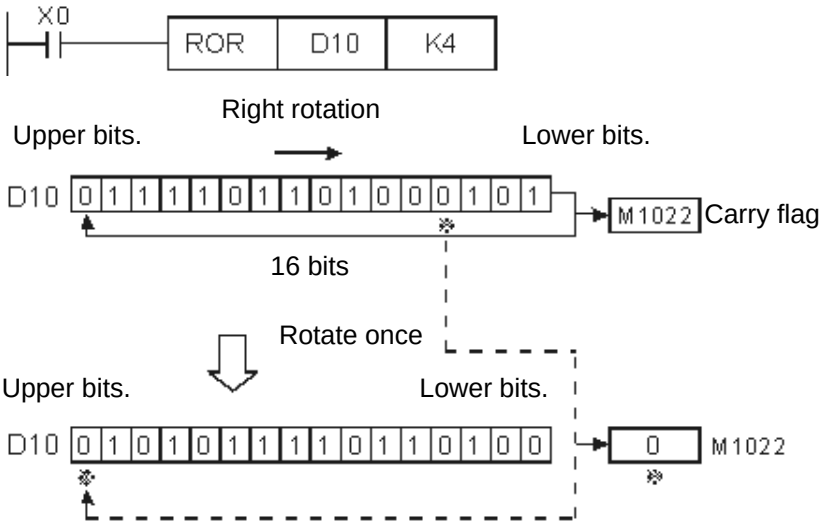
Command description

- (D): Device to be rotated. (n): Number of bits to be rotated in one operation.
- Right rotate (n) bits of digit contained in device specified by (D) for one time.

Example

- In case X0 change from Off→On, the 16 bits of number kept in D10 right rotates in unit of 4 bits as shown in figure below.

The ※ marked bit value is sent to carry flag M2826.



API		ROL		(D) (n)	Rotate left	Model
24	D					NC300 ✓

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

Notes on the use of operands:

For D operands running on Z devices only 16-bit commands are valid.

For D operands assigned to KnY, KnM and KnA only K4 (16-bit) and K8 (32-bit) are valid.

Valid value for n operand is 1~16 (16-bit) and 1~32 (32-bit)

16-bit command (5 STEP)
ROL Continuous running type

32-bit command (9 STEP)
DROL Continuous running type

- Flag: M2826 carry flag

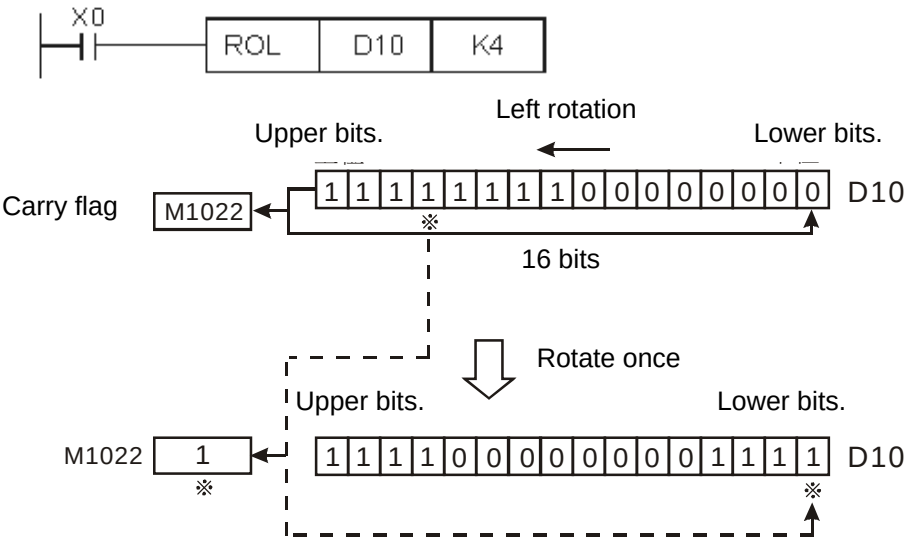
Command description

- (D): Device to be rotated. (n): Number of bits to be rotated in one operation.
- Left rotate (n) bits of digit contained in device specified by (D) for one time.

Example

◆ In case X0 change from Off→On, the 16 bits of number kept in D10 left rotates in unit of 4 bits as shown in figure below.

The ※ marked bit value is sent to carry flag M2826.



API		ZRST		D_1 D_2	Zone reset	Model
25						NC300
						✓

	Bit device				Word device										
	X	Y	M	A	K	F	KnX	KnY	KnM	KnA	T	C	D	V	Z
D1		＊	＊	＊							＊	＊	＊		
D2		＊	＊	＊							＊	＊	＊		

Notes on the use of operands:

The $D_{1\ operand}$ ID ≦ D_2 operand ID.

Both D_1 and D_2 operands must be assigned to devices of the same type.

16-bit command (4 STEP)
ZRST Continuous running type

32-bit command

- Flag: None

Command description

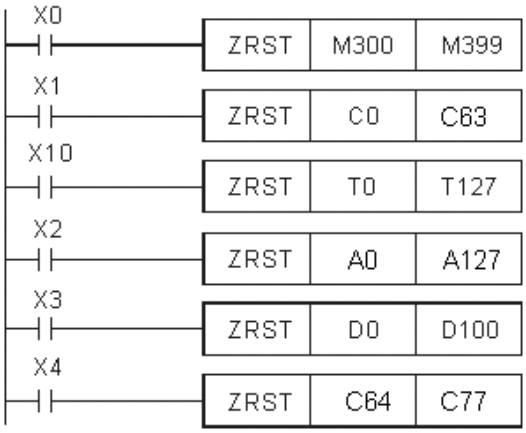
◆ D_1: Zone reset starting device. D_2: Zone reset ending device.

◆ The 16-bit and 32-bit counters of the NC300 series models cannot use the ZRST command together.

◆ In case D_1 operand ID > D_2 operand ID, only the device assigned by D_2 is reset.

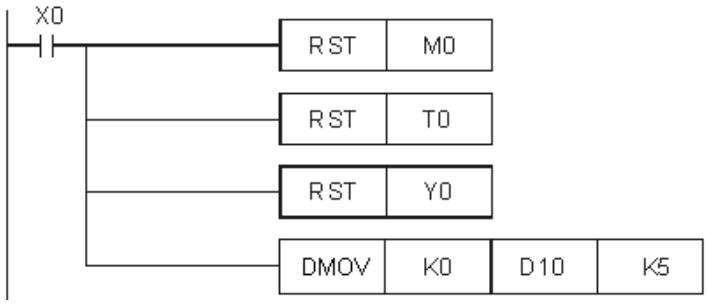
Example

- In case X0 is ON, auxiliary relays M300 ~ M399 are reset to Off.
- In case X1 is On, 16-bit counters C0 ~ C63 are all reset. (Overwrite with value 0 and reset contacts and coils to Off.)
- In case X10 is On, Timer T0 ~ T127 are all reset. (Overwrite with value 0 and reset contacts and coils to Off.)
- In case X2 is On, alarm points A0 ~ A127are all reset to Off.
- In case X3 is On, data registers D0 ~ D100 are all reset 0.
- In case X4 is On, 32-bit counters C64 ~ C77 are all reset. (Overwrite with value 0 and reset contacts and coils to Off.)



Supplementary description

- Devices including bit device Y, M, and A and Word device T, C, and D may use RST command independently.
- The same effects may be realized with command DMOV (API 09) by sending K0 to multiple points of Word device T, C, D or bit register KnY, KnM, KnA.



API		DECO		(S) (D) (n)	Decoder	Model
26						NC300
						✓

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

Notes on the use of operands:
In case the D operand is a bit device then the valid range of n operand is 1~8 or 1~4 if it is a Word device.

16-bit command (6 STEP)
DECO Continuous running type

32-bit command
- - - -

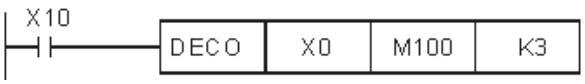
- Flag: None

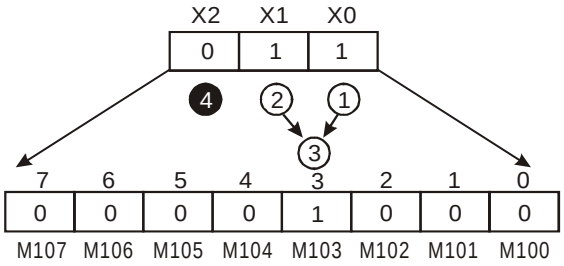
Command description

- (S): Source device for decoding. (D): Target device where decoded value is kept. (n): Decoding bit length.
- Decode the lower bits of the "n" bits in source device (S) and save its outcome of "2^n" bit length in (D).

Example I

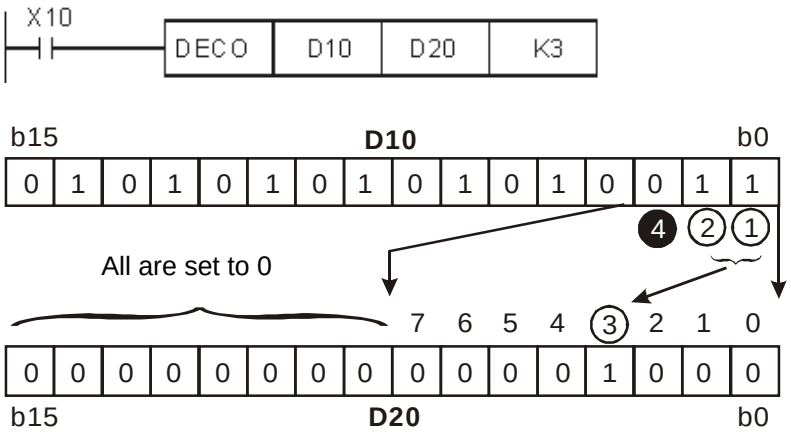
- In case (D) is a bit device, the valid values of n is in range 1~8. If n = 0 or n > 8, the DECO command returns with error.
- In case n=8, the DECO command can decode up to 256 (2^8) points. (Please ensure that the range of storage devices after decoding is not duplicated.)
- In case X10 = Off→On, the DECO command decodes values stored in X0~X2 to M100~M107.
- In case data source is 1+2 = 3, then the third bit (M103) from M100 is set to 1.
- In case the DECO command turns X10 to Off, those have been decoded continue the operation.





Example 2

- In case (D) is a Word device, the valid values of n is in range 1~4. If n = 0 or n > 4, the DECO command returns with error.
- In case n=4, the DECO command can decode up to 16 (2^4) points.
- In case X10 = Off→On, the DECO command decodes values stored in D10's b2~b0 to D20's b7~b0. Bits not used (b15~b8) in D20 are all set to 0.
- The DECO command decodes lower three bits of D10 and saves in the lower 8 bits of D20. The upper 8 bits of D20 is set to 0.
- In case the DECO command turns X10 to Off after its execution, those have been decoded continue the operation.



API		ENCO		(S) (D) (n)	Encoder	Model
27						NC300
						✓

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

Notes on the use of operands:
In case the S operand is a bit device then the valid range of n operand is 1~8 or 1~4 if it is a word device.

16-bit command (6 STEP)
ENCO Continuous running type

32-bit command
- - - -

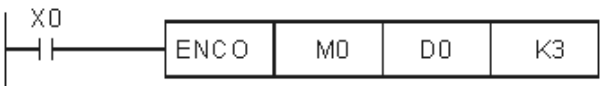
- Flag: None

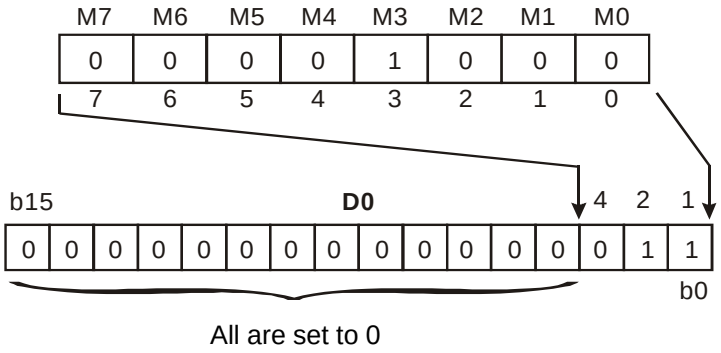
Command description

- (S): Source device for encoding. (D): Target device where encoded value is kept. (n): Encoding bit length.
- Encode the lower bits of the "n" bits in source device (S) and save their outcome of "2^n" bit length in (D).
- On case most bits of the data source (S) are 1 in value, then its lower bits are left un-attended.

Example I

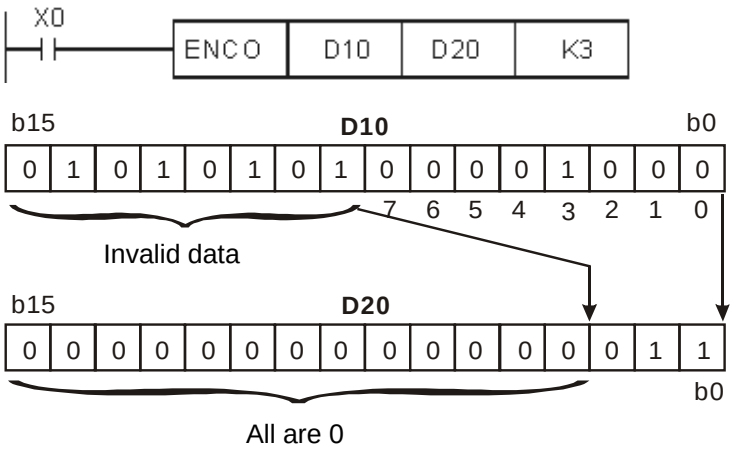
- In case (S) is a bit device, the valid values of n is in range 1~8. If n = 0 or n > 8, the ENCO command returns with error.
- In case n=8, the ENCO command can encode up to 256 (2^8) points.
- In case X0 = Off→On, the ENCO command encodes 8 (2^3) bits of data (M0~ M7) and saves in the lower bits (b2~b0) of D0. Bits not used in D0 (b15~b3) are all set to 0.
- In case the ENCO command turns X0 to Off after its execution, data in (D) remains intact.





Example 2

- In case (S) is a word device, the valid values of n is in range 1~4. If n = 0 or n > 4, the ENCO command returns with error.
- In case n=4, the ENCO command can encode up to 16 (2^4) points.
- In case X0 = Off→On, the ENCO command encodes 8 (2^3) bits of data (M0~ M7) in D10 and saves in the lower bits (b2~b0) of D20. Bits not used in D20 (b15~b3) are all set to 0. (Values in D10's b8~b15 are all invalid data.)
- In case the ENCO command turns X0 to Off after its execution, data in (D) remains intact.



API		BON		(S) (D) (n)	Bit ON detect	Model
28	**D**					NC300 ✓

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

Notes on the use of operands:
For S operands running on Z devices only 16-bit commands are valid.
n=0~15 (16-bit command).
n=0~31 (32-bit command)

16-bit command (6 STEP)
BON Continuous running type

32-bit command (7 STEP)
DBON Continuous running type

- Flag: None

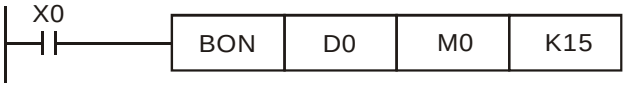
Command description

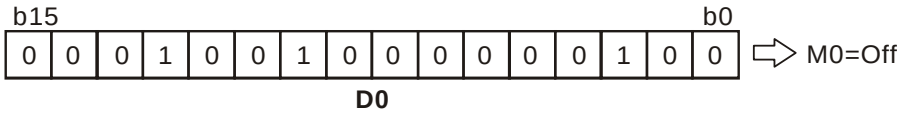
◆ (S): Source device. (D): Target device for judgment outcome. (n): Position of bit to be judged (beginning with 0).

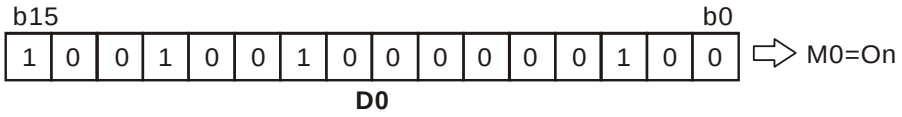
Example

◆ In case X0=On and the value of the 15th bit in D0 is “1” then M0=On and M0=Off the value is "0" instead.

◆ If X0 turns to Off, M0 remains intact.







API		ANS		(S) (m) (D)	Alarm point output	Model
29						NC300
						✓

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

Notes on the use of operands:
Valid values for S operands of NC300 series models are T0~T255.
The m operand can assign values in range of K1~K32,767 in unit of 100ms or 10ms as determined by T(n), where n=0~255.
Valid values for D operand of NC300 series models are A0~A511 with T0~T199 in unit of 100ms and T200~T255 in 10ms.

16-bit command (5 STEP)
ANS Continuous running type - -

32-bit command
- - - -

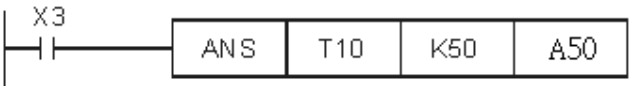
• Flag: Please refer to supplementary descriptions shown below.

Command description

- (S): Detecting alarm timer. (m): Timing span setup. (D): Alarm point device.
- The ANS command is exclusive for driving alarm point outputs.

Example

- In case X3=On for more than 5 seconds then alarm point A50=On. After 5 seconds, the X3 is set to Off while A50 remains On. (But T10 is reset to Off with current value=0.)



API		ANR			Alarm point reset	Model
30						NC300
						✓

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

Notes on the use of operands: No operand is required.

16-bit command (1 STEP)
ANR Continuous running type

32-bit command
- - - -

• Flag: None

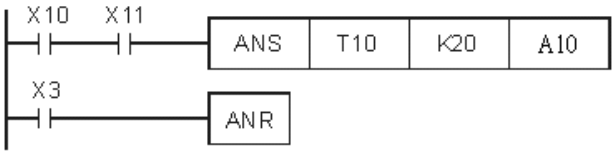
Command description

- The ANR command resets the alarm point exclusively.
- In case multiple alarm points turn On the one with the least ID resets.

Example

- In case both X10 and X11 turn On for more than 2 seconds, the alarm point A10 turns On and remains so even after both X10 and X11 turn Off. (But T10 is reset to Off with current value=0.)

- In case both X10 and X11 turn On for less than 2 seconds, the T10's current value is reset to 0.
- In case X3=Off→On, the active alarm point is reset. Valid alarm points of NC300 series models are A0~A511.
- In case X3 changes Off→On again the alarm point with the next least ID resets.

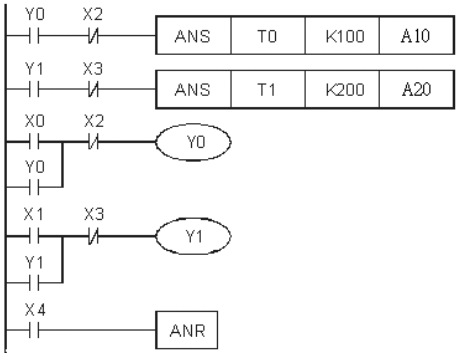


Supplementary description

- How to use alarm points:

I/O point configuration:

X0: forward switch		Y0: forward	A10: forward alarm point
X1: backward switch		Y1: backward	A20: backward alarm point
X2: front end positioning switch			
X3: back end positioning switch			
X4: alarm point reset button			



1. Y0=On for more than 10 seconds yet the object does not reach given front end position X2, A10=On.
2. Y1=On for more than 10 seconds yet the object does not reach given back end position X3, A20=On.
3. In case backward switch X1=On and backward device Y1=On, then the backward device Y1 turns Off till the workpiece reaches backward positioing switch X3.
4. One of the active alarm points resets every time the alarm point reset button is pressed. The reset starts from the point with the least ID number.

API		REF		(D) (n)	I/O refresh	Model
31						NC300
						✓

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

Notes on the use of operands:
Operand D must assign a device with the code ending in the number "0", such as X0, X10, Y0, Y10 and so on. Please refer to the supplementary description shown on the next pages.
The "n" operand ranges from 8 to 256 and is in unit of 8.

16-bit command (3 STEP)
REF Continuous running type

32-bit command
- - - -

- Flag signal: None

Command description

◆ (D): The starting device for I/O refresh.(n): Number of devices to be I/O refreshed.

◆ The MLC I/O terminals are refreshed only after all their statuses are scanned (up to the END one). The status of the input device is read from the status of the external input point and saved in the input point's memory after the program scanning is started. Contents contained in the output terminal's memory are sent to output devices only after the END command is executed. Use this command to get the latest I/O data during computing.

◆ Operand D must assign a device with the code ending in the number "0" like X0, X10, Y0, Y10 and so on. Please refer to the supplementary description shown on the next pages. The "n" operand ranges from 8 to 256 and is in units of 8. Numbers not meeting this criteria are most likely treated as errors. The use range varies with models. Please refer to the relevant supplementary descriptions.

Example I

◆ In case X0=On, the PLC reads the status of the input points X0~X15 and updates the input signals immediately (without any input delay).

Example 2

◆ In case X0=On the output signal from 8 points Y0~Y7 are sent to the output end for immediate update (without pending for output till END command).

Supplementary description

- I/O and RIO points that can be processed by NC300 series are without limits. That is, the "n" operands have a range of n=K8 or K16.

API		DHSCS		(S1) (S2) (D)	Compare setup (high speed counter)	Model
32	**D**					NC300 ✓

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

Notes on the use of operands:

The S_2 operand must refer to high speed counter C78 and C79. Please refer to the supplementary description shown on the next pages.

The D operand range may refer to IC00 and IC01 as well as be modified by indirect register V and Z.

See the specification included with each model for the use range of specific devices.

This command is valid for the 32-bit command version DHSCS only.

This command sets up a comparison value for high speed counters.

High speed counters count by hardware. They return interrupt messages when the counting criteria is validated. In case an interrupt is enabled, a high level output is sent to the M or On Board Y device. For output to Y device, the output changes in accordance with the polarity setup of Y device (contact a or b).

16-bit command

-	-	-	-

32-bit command (5STEP)

DHSCS	Continuous running type	-	-

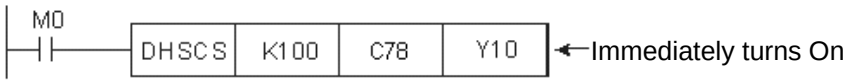
- Flag signal: M2872~M2873 are for high speed counter interruption disabling. Please refer to Example 3 described below.

Command description

- (S1): Comparison value. (S2): High speed counter ID. (D): Comparison outcome.
- The high speed counter is activated by interruption generated by input of the corresponding external high speed counter. When the high speed counter referred to by DHSCS command (S2) increases or decreases the value in the unit of one, the DHSCS command compares the current value of the high speed counter against the setup value given by (S1) immediately and sets On the device assigned by (D). The device remains On even if the comparison result becomes unequal later.
- If the devices assigned by (D) are Y0~ Y23 (for On Board Ys only), then when the comparison settings are equal to the current value of the high speed counters, an output is sent to external Y0~Y23 (for On Board Ys only) output end immediately. Remaining Y devices vary with scan cycle. Both device M and A effect immediately instead of subjected to the scan cycle.

Example I

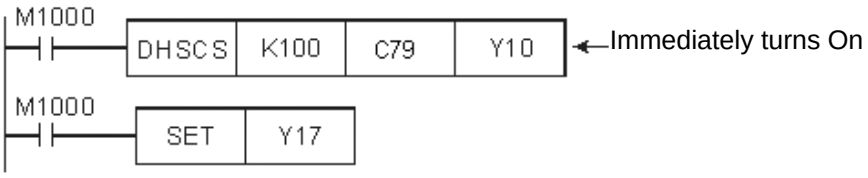
◆ After a RUN command is executed by MLC and the DHSCS command is executed when M0=On, then if the current value of C78 changes from 99 to 100 or 101 to 100, then Y10 turns On and outputs to external Y10's output end. It then remains On afterwards.



Example 2

◆ The difference between DHSCS command's and general Y outputs:

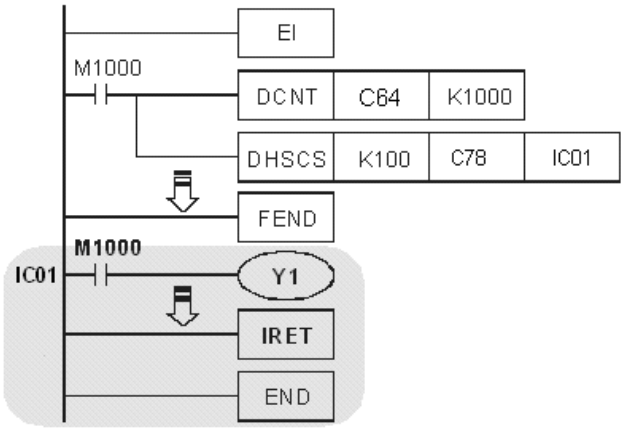
1. When C79's current value changes from 99 to 100 or 101 to 100, the DHSCS command's output Y10 is sent to the external output end by an immediate interrupt which is irrelevant to the MLC scan time yet still delays the output delay caused by the output module's relay (10ms) or transistor (10us).
2. When C79's current value changes from 99 to 100 the C79 contact turns on immediately. However, when running to SET Y17, the Y17 is still subject to scan time and outputs only efter END is executed.



Example 3

◆ High speed counter interrupts:

1. DHSCS command's D operand may assign IC00, IC01 to generate an interrupt when the counter's counting arrives and executes the relevant interrupt service program.
2. Limits imposed on the use of high speed counter interrupts by NC300 series models: When using the DHSCS command's I interrupt, the high speed counter assigned by it can be used by no other DHSCS command or errors will be detected.
3. The NC300 series' high speed counter generates the given interrupts when the counting data arrives. Here C78 is set as the first counter for high speed counting input with interrupts coded IC00 or IC01.
4. When the current value of C78 changes from 99 to 100 or 101 to 100 (together with MLC#312 parameter being opened) the program jumps to the interrupt indicator IC01 for the execution of interrupt service subroutine.



- The NC300 series' M2872 and M2873 refer to high speed counters IC00~IC01 respectively. That is, when M2872 is OFF then interrupt IC0 is disabled.

Interrupt ID	Interrupt disable flag
IC00	M2872
IC01	M2873

API		DHSCR		(S1) (S2) (D)	Compare reset (high speed counter)	Model
33	**D**					NC300 ✓

	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 command

-	-	-	-

32-bit command (5STEP)

DHSCR	Continuous running type	-	-

- Flag signal:

Notes on the use of operands:

The S_2 operand must refer to high speed counter C78 and C79. Please refer to the supplementary description of API 53 DHSCS.

The D operand range may refer to the same high speed counter of S_2 (C78~C79 only).

This command is valid for the 32-bit command version DHSCR only.

This command sets up a comparison value for high speed counters.

High speed counters count by hardware. They return interrupt messages when the counting criteria is validated. In case an interrupt is enabled, a high level output is sent to the M or On Board Y device. For output to Y device, the output changes in accordance with the polarity setup of Y device (contact a or b).

Command description

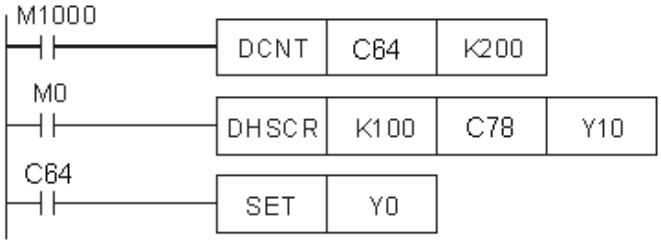
- (S1): Comparison value. (S2): High speed counter ID. (D): Comparison outcome.
- The high speed counter is activated by the interruption generated by the input of the corresponding external high speed counter. When the high

speed counter referred to by DHSCR command (S2) increases or decreases the value in the unit of one, the DHSCR command compares the current value of the high speed counter against the setup value given by (S1) immediately and sets Off the device assigned by (D). The device remains Off even if the comparison result becomes unequal later.

- If the devices assigned by (D) are Y0~ Y23 (On Board Ys), then when the comparison settings are equal to the current values of the high speed counters, the output to them is immediate rather than the other Y devices' delay by the scan cycle. Both device M and S are effected immediately instead of being subjected to the scan cycle.

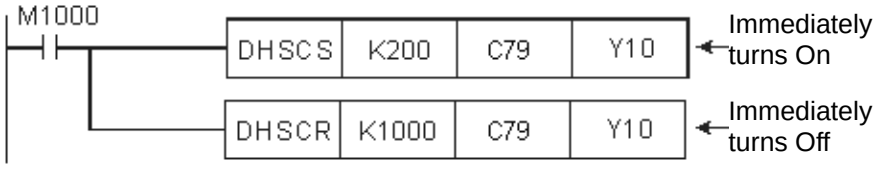
Example I

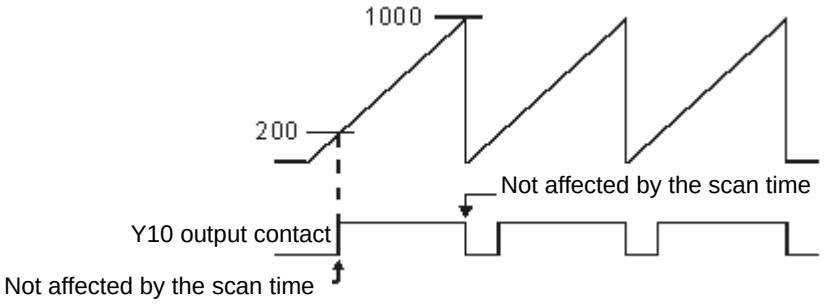
- In case M0=On and current value of C78 changes from 99 to 100 or 101 to 100, then Y10's Off status is reset.
- In case the current value of counter C64 changes from 199 to 200, the C64's contact turns on and sets Y0 On with the output delayed by the program scan time.
- Y10 is a component that features an immediate status reset when the given counting data is received. It may be assigned as a high speed counter of the same code. Please refer to Example 2 for detail.



Example 2

- When being assigned as a high speed counter of the same code, C79 contact's status is reset to off when the current value of C79 changes from 999 to 1000 or 1001 to 1000.





API		ALT		(D)	On/Off alternate	Model
34						NC300
						✓

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

Notes on the use of operands:
See the specification included with each model for the use range of specific devices.

16-bit command (3 STEP)

ALT	Continuous running type	ALTP	Pulse executed type

32-bit command

-	-	-	-

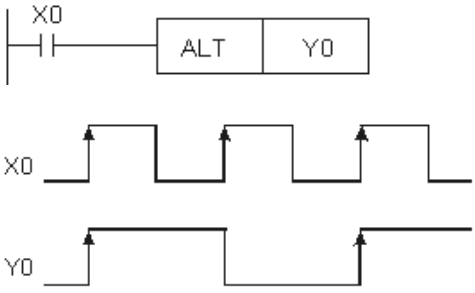
- Flag signal: None

Command description

- (D): Target device.
- This command is commonly used as a execution command (ALT).

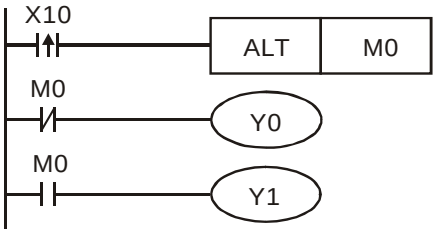
Example I

- In case Y0 turns On when X0 changes from Off to On for the first time, then Y0 turns Off when X0 changes from Off to On for the second time.



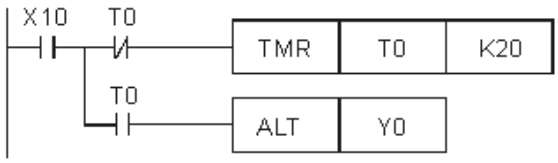
Example 2

- Use a single switch to control the start up and stop operation. In the beginning, M0=Off and so Y0=On and Y1=Off. When X10 does its first On/Off the M0 turns On which leads to Y1=On, Y0=Off. When X10 does its second On/Off the M0 turns Off and results in Y0=On and Y1=Off.



Example 3

◆ To create a flashing operation. When X10 is On, T0 generates one pulse in every 2 seconds and Y0 turns On and Off alternatively based on the T0 pulses.



API		LD※		(S1) (S2)	Contact type compare LD※	Model
39~44						NC300
						✓

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

Notes on the use of operands:

※ : = 、 > 、 < 、 <> 、 ≦ 、 ≧

See specification included with each model for use range of specific devices.

16-bit command (4 STEP)

LD※	Continuous running type	-	-

32-bit command (6 STEP)

DLD※	Continuous running type	-	-

- Flag signal: None

Command description

◆ (S1): Data source device 1. (S2): Data source device 2.

◆ This command compares values stored in (S1) and (S2). Take API 39 (LD=). If both operands are equal to each other the command turns on otherwise it does not turn on.

◆ The LD※ command may connect to a bus bar directily.

API No.	16-bit device	32-bit device	Turn-on condition	Not turn-on condition
39	LD =	**D**LD =	(S1) = (S2)	(S1) ≠ (S2)
40	LD >	**D**LD >	(S1) > (S2)	(S1) ≦ (S2)
41	LD <	**D**LD <	(S1) < (S2)	(S1) ≧ (S2)
42	LD < >	**D**LD < >	(S1) ≠ (S2)	(S1) = (S2)
43	LD < =	**D**LD < =	(S1) ≦ (S2)	(S1) > (S2)

44	LD > =	**D**LD > =	(S1) ≧ (S2)	(S1) < (S2)

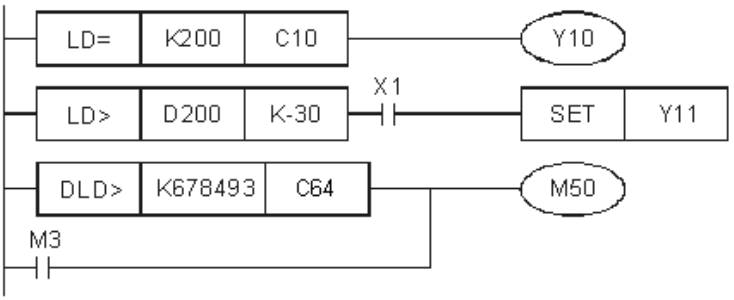
◆ Use the 32-bit command (DLD※) to compare the 32-bit counter (C64~C77) instead of the 16-bit command (LD※). Or the MLC program determines it as "Program error" and flashs the ERROR light indicator on the main board.

Example

◆ In case data contained in C10 equals to that in K200 then Y10=On.

◆ In case data contained in D200 is greater than that in K-30 and X1=On then Y11=On and remains so.

◆ In case data contained in C64 is less than K678,493 or M3=On then M50=On.



API		AND※		(S1) (S2)	Contact type compare	Model
45~50	**D**				AND※	NC300 ✓

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

Notes on the use of operands:

※ : = 、 > 、 < 、 <> 、 ≦ 、 ≧

See the specification included with each model for the use range of specific devices.

16-bit command (4 STEP)

AND※	Continuous running type	-	-

32-bit command (6 STEP)

DAND※	Continuous running type	-	-

• Flag signal: None

Command description

◆ (S1): Data source device 1. (S2): Data source device 2.

◆ This command compares values stored in (S1) and (S2). Take API 45 (AND=). If both operands are equal to each other the command turns on otherwise it does not turn on.

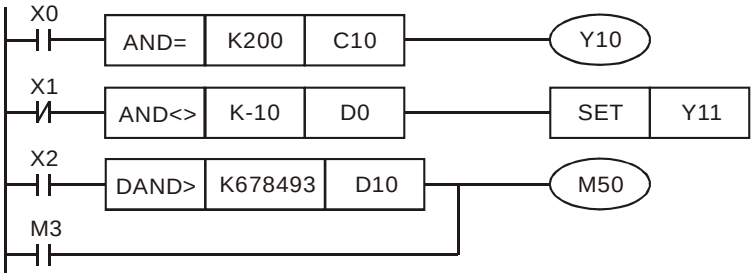
◆ The AND※ is a compare command series connects to a contact.

API No.	16-bit device	32-bit device	Turn-on condition	Not turn-on condition
45	AND =	**D**AND =	(S1) = (S2)	(S1) ≠ (S2)
46	AND >	**D**AND >	(S1) > (S2)	(S1) ≦ (S2)
47	AND <	**D**AND <	(S1) < (S2)	(S1) ≧ (S2)
48	AND < >	**D**AND < >	(S1) ≠ (S2)	(S1) = (S2)
49	AND < =	**D**AND < =	(S1) ≦ (S2)	(S1) > (S2)
50	AND > =	**D**AND > =	(S1) ≧ (S2)	(S1) < (S2)

- Use the 32-bit command (DAND※) to compare 32-bit counter (C64~C77) instead of the 16-bit command (AND※). Or the MLC program determines it as "Program error" and flashs the ERROR light indicator on the main board.

Example

- In case the data contained in C10 equals to that in K200 and X0=On then Y10=On.
- In case the data contained in D0 is not equal to that in K-10 and X1=Off then Y11=On and remains so.
- In case X2=On and the data contained in 32-bit register D10(D11) are less than 678,493 then M50=On.



API		OR※		(S1) (S2)	Contact type compare	Model
51~56	**D**				OR※	NC300 ✓

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

Notes on the use of operands:

※ : = 、 > 、 < 、 <> 、 ≦ 、 ≧

See the specification included with each model for the use range of specific devices.

16-bit command (4 STEP)

OR※	Continuous running type	-	-

32-bit command (6 STEP)

DOR※	Continuous running type	-	-

- Flag signal: None

Command description

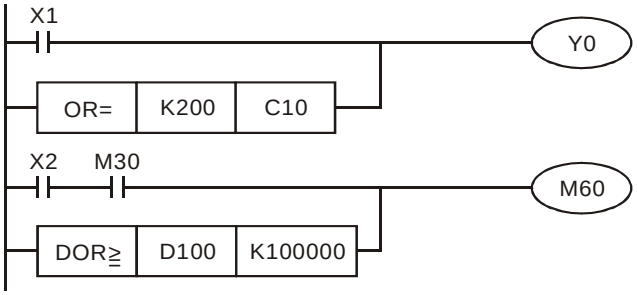
- (S1): Data source device 1. (S2): Data source device 2.
- This command compares values stored in (S1) and (S2). Take API 51 (OR=). If both operands are equal to each other the command turns on otherwise it does not turn on.
- The OR※ is a compare command parallel connects to a contact.

API No.	16-bit device	32-bit device	Turn-on condition	Not turn-on condition
51	OR =	**D**OR =	(S1) = (S2)	(S1) ≠ (S2)
52	OR >	**D**OR >	(S1) > (S2)	(S1) ≦ (S2)
53	OR <	**D**OR <	(S1) < (S2)	(S1) ≧ (S2)
54	OR < >	**D**OR < >	(S1) ≠ (S2)	(S1) = (S2)
55	OR < =	**D**OR < =	(S1) ≦ (S2)	(S1) > (S2)
56	OR > =	**D**OR > =	(S1) ≧ (S2)	(S1) < (S2)

- Use the 32-bit command (DOR※) to compare the 32-bit counter (C64~C77) instead of the 16-bit command (OR※). Or the MLC program determines it as "Program error" and flashs the ERROR light indicator on the main board.

Example

- In case the data contained in C10 equals to that in K200 and X10=On then Y0=On.
- In case both X2 and M30 is On or the data contained in 32-bit register D100(D101) is greater than or equal to K100,000 then M60=On.



API		VRT		(S) (n) (D)	Logic switch form	Model
57	D					NC300 ✓

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

Notes on the use of operands:

16-bit command(70 STEP)
VRT - Continuous running type

32-bit command (134 STEP)
DVRT Continuous running type

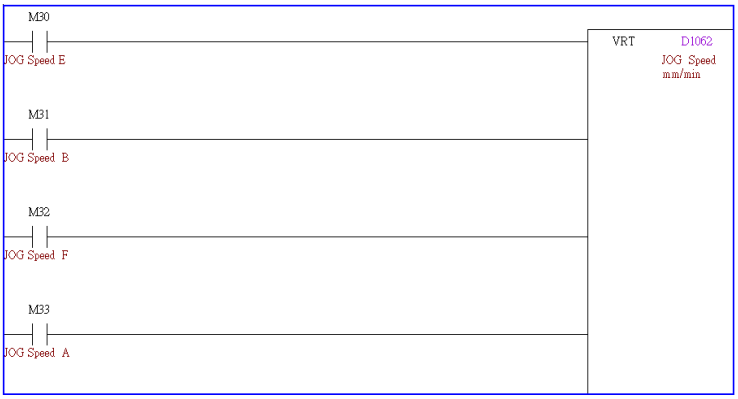
- Flag signal: None

Command description

- (S): Source device to be transformed. (n): Number of source devices. (D): Outcome of transformation.
- (S): Beginning of the source of the transforming device and assigned in sequence for the number of (n) devices. When the source device's logic switches, a value from the given logic number table is sent to the specified register (D) for type conversion.
- (S): The source operand can assign X or Y, M, T, and C to convert to the predefined value when the source device undergoes contact switching.

Example I

- In case M30=On, M31=On, M32=Off, and M33=Off, their binary number changes to 3 and its corresponding value in the table is 50, then it saves number 50 in D1062.



Edit Table

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

Example 2

API		FADD		(S1) (S2) (D)	Binary floating point number addition	Model
58						NC300 ✓

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

Notes on the use of operands:
See the specification included with each model for the use range of specific devices.
This command is valid for the 32-bit command version FADD only.

16-bit command
- - - -

32-bit command (7 STEP)
FADD Continuous running type

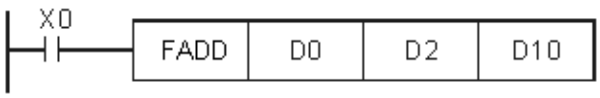
- Flag signal: M2824 Zero flag
  M2825 Borrow flag
  M2826 Carry flag

Command description

- (S1): Summand. (S2): Addend. (D): Sum.
- Add the value contained in the register assigned by (S1) and (S2) save the sum in the register assigned by (D) with all operations executed in binary floating point number format.
- The constant (K or F) contained in the source operand (S1) or (S2) will be converted into a binary floating point number before the addition operation.
- Both (S1) and (S2) can refer to the same register. For "continuous running" type commands, the addition operation is executed every time the operand is scanned during the condition that the contact point is On.
- In case the absolute value of the sum is greater than the maximum value of floating point, the carry flag M2826 turns On.
- In case the absolute value of the sum is less than the minimum value of floating point, the carry flag M2825 turns On.
- In case the sum equals 0, the zero flag M2824 turns On.

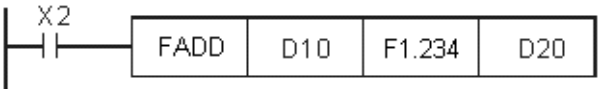
Example I

- In case X0=On places the sum of a binary floating point number (D1, D0) + binary floating point number (D3, D2) in (D11, D10).



Example 2

- In case X2=On places the sum of a binary floating point number (D11, D10) + F1.234 (after automatically converted into a binary floating point format) in (D21, D20).



API		FSUB		(S1) (S2) (D)	Binary floating point number subtraction	Model
59						NC300
						✓

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

Notes on the use of operands:
See the specification included with each model for the use range of specific devices.
This command is valid for the 32-bit command version FSUB only.

16-bit command
- - - -

32-bit command (7 STEP)
FSUB Continuous running type

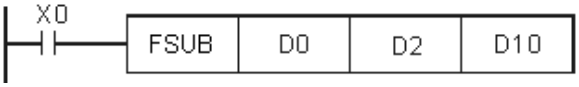
- Flag signal: M2824 Zero flag
  M2825 Borrow flag
  M2826 Carry flag

Command description

- (S1): Minuend. (S2): Subtrahend. (D): Remnant.
- Subtract value contained in the register (S1) by value contained in the register (S2) and save the remnant in the register defined by (D) with all operations executed in binary floating point number format.
- The constant (K or F) contained in the source operand (S1) or (S2) will be converted into a binary floating point number before the subtraction operation.
- Both (S1) and (S2) can refer to the same register. For "continuous running" type commands, the subtraction operation is executed every time the operand is scanned during the condition that the contact point is On.
- In case the absolute value of the remnant is greater than the maximum value of floating point, the carry flag M2826 turns On.
- In case the absolute value of the remnant is less than the minimum value of floating point, the carry flag M2825 turns On.
- In case the remnant equals 0, the zero flag M2824 turns On.

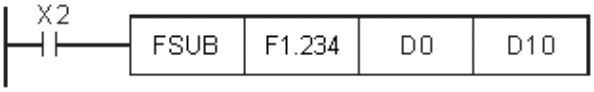
Example I

- In case X0=On places the remnant of the binary floating point number (D1, D0) - binary floating point number (D3, D2) in (D11, D10).



Example 2

◆ In case X2=On places remnant of F1.234 (after automatically being converted into a binary floating point format) - binary floating point number (D1, D0) in (D11, D10).



API		FMUL		(S1) (S2) (D)	Binary floating point number multiplication	Model
60						NC300 ✓

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

Notes on the use of operands:
See the specification included with each model for the use range of specific devices.
This command is valid for the 32-bit command version FMUL only.

16-bit command
- - - -

32-bit command (7 STEP)
FMUL Continuous running type

- Flag signal: M2824 Zero flag
  M2825 Borrow flag
  M2826 Carry flag

Command description

◆ (S1): Multiplicand. (S2): Multiplier. (D): Product.

◆ Multiply the value contained in the register assigned by (S1) and (S2) save the product in the register assigned by (D) with all operations executed in binary floating point number format.

◆ The constant (K or F) contained in the source operand (S1) or (S2) will be converted into a binary floating point number before the multiplication operation.

◆ Both (S1) and (S2) can refer to the same register. For "continuous running" type commands, the multiplication operation is executed every time the operand is scanned during the condition that the contact point is On.

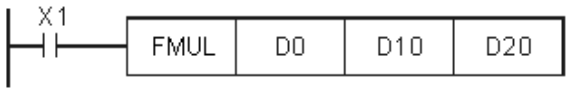
◆ In case the absolute value of the product is greater than the maximum value of floating point, the carry flag M2826 turns On.

◆ In case the absolute value of the product is less than the minimum value of floating point, the carry flag M2825 turns On.

◆ In case the product equals 0, the zero flag M2824 turns On.

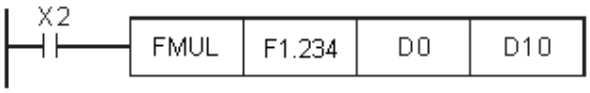
Example I

- In case X1=On places the product of the binary floating point number (D1, D0) multiply binary floating point number (D11, D10) in the register assigned by (D21, D20).



Example 2

- In case X2=On places the product of the constant F1.234 (after automatically converted into a binary floating point format) × binary floating point number (D1, D0) in (D11, D10).



API		FDIV		(S1) (S2) (D)	Binary floating point number division	Model
61						NC300 ✓

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

Notes on the use of operands:
See the specification included with each model for the use range of specific devices.
This command is valid for the 32-bit command version FDIV only.

16-bit command
- - - -

32-bit command (7 STEP)
FDIV Continuous running type

- Flag signal: M2824 Zero flag
  M2825 Borrow flag
  M2826 Carry flag

Command description

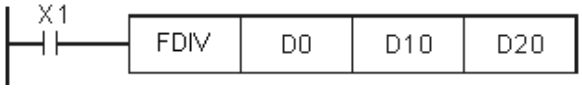
- (S1): Dividend. (S2): Divisor. (D): Quotient and remainder.
- Divide value contained in the register (S1) by value contained in the register (S2) and save the quotient in the register defined by (D) with all operations executed in binary floating point number format.
- The constant (K or F) contained in the source operand (S1) or (S2) will be converted into a binary floating point number before the division operation.
- In case value in (S2) is zero, then the command is ignored with error message "computing error" and error code H'0E19 sets to M1067, M1068=On, and D1067.
- In case the absolute value of the quotient is greater than the maximum value of floating point, the carry flag M2826 turns On.
- In case the absolute value of the quotient is less than the minimum value

of floating point, the carry flag M2825 turns On.

◆ In case the quotient equals 0, the zero flag M2824 turns On.

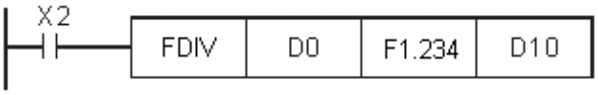
Example I

◆ In case X1=On places the remainder of the binary floating point number (D1, D0) divided by binary floating point number (D11, D10) in the register assigned by (D21, D20).



Example 2

◆ In case X2=On places the outcome of the binary floating point number (D1, D0) ÷ K1,234 (after automatically converted into a binary floating point format) in (D11, D10).



API		FCMP		(S1) (S2) (D)	Binary floating point number compare	Model
62						CMP-A
						✓

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

Notes on the use of operands:
The D operand takes consecutive 3 points.
See the specification included with each model for the use range of specific devices.
This command is valid for the 32-bit command version FCMP only.

16-bit command
- - - -

32-bit command (7 STEP)
FCMP Continuous running type

- Flag signal: None

Command description

◆ (S1): Comparison value 1 (in binary floating point format). (S2): Comparison value 2 (in binary floating point format). (D): Comparison output in consecutive 3 points.

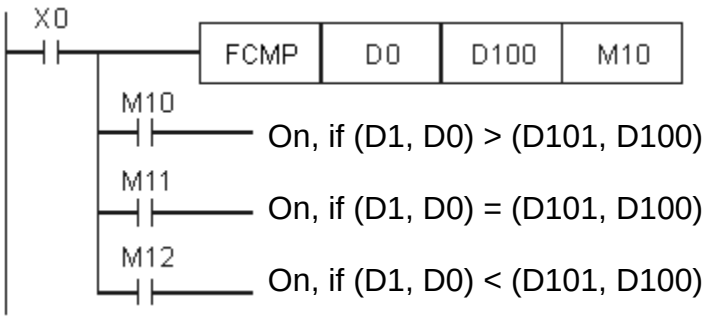
◆ Compare binary floating point number 1 and 2 and place the outcome (> , = , <) in (D).

◆ The constant (K or F) contained in the source operand (S1) or (S2) will be converted into a binary floating point number before the comparing operation.

Example

◆ If device M10 is assigned, the outcome takes points M10~M12 automatically.

- The FCMP command executes when X0=On and sets one of M10~M12 on. It is ignored when X0=Off and status of M10~M12 remain intact.
- In case a combined comparison like ≧, ≦, or ≠ is required, users can connect M10~M12 serially or in parallel to get the desired result.
- Use RST or ZRST command to reset comparison outcome.



API		FINT		(S) (D)	Binary floating point number convert to BIN integer	Model
63						NC300 ✓

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

Notes on the use of operands:
See the specification included with each model for the use range of specific devices.
The S operand takes consecutive 2 points.
This command is valid for the 32-bit command version FINT only.

16-bit command
- - - -

32-bit command (5 STEP)
FINT Continuous running type

- Flag signal: M2824 Zero flag
  M2825 Borrow flag
  M2826 Carry flag

Command description

- (S): The source device to be converted. (D): The conversion outcome.
- The value contained in the register assigned by (S) is converted from binary floating point number to BIN integer and saves in the register assigned by (D) with the integral floating point number being discarded.
- This command functions exactly the opposite against API 64 (FDOT).
- For conversion outcome in zero, the zero flag M2824 sets On.

  For conversion outcome with floating point number being discarded, sets borrow flag M2825 On.

  For conversion outcome exceeding ranges listed below (overflow), sets

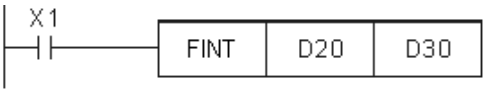
carry flag M2826 On.

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

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

Example

◆ In case X1=On convert binary floating point number (D21, D20) to BIN integer, save the outcome in (D31, D30), and discards the BIN integral floating point number.



API		FDOT		(S) (D)	Binary integer convert to binary decimal	Model
64						NC300
						✓

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

Notes on the use of operands:
See the specification included with each model for the use range of specific devices.
The D operand takes consecutive 2 points.
This command is valid for the 32-bit command version FDOT only.

16-bit command
- - - -

32-bit command (5 STEP)
FDOT Continuous running type

- Flag signal:
  M1081 FLT command function switch
  M2824 Zero flag
  M2825 Borrow flag
  M2826 Carry flag

Command description

◆ (S): The source device to be converted. (D): The device to keep conversion outcome.

◆ In case M1081=Off, it converts the BIN integer into a binary decimal number. Here the 16-bit command FDOT's source device (S) occupies one register and the target device (D) occupies two registers.

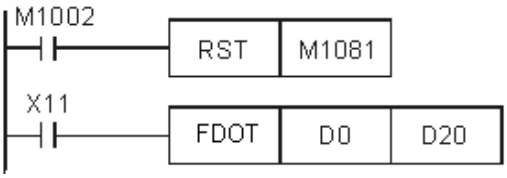
1. In case the absolute value of the outcome is greater than the maximum value of floating point, the carry flag M2826 turns On.
2. In case the absolute value of the outcome is less than the minimum value of floating point, the carry flag M2825 turns On.
3. In case the outcome equals 0, the zero flag M2824 turns On.

◆ In case M1081=On, it converts the binary decimal number into a truncated BIN integer. This 16-bit command FLT's source device (S) takes 2 registers while the target device (D) for outcome storage takes 1 register. It functions in the same way of an INT command.

1. If the outcome value exceeds the range of numbers that can be presented by 16-bit or 32-bit register assigned by (D) (-32,768~32,767 and -2,147,483,648~2,147,483,647 respectively) then the maximum or minimum value is saved in device (D) and the carry flag M2826 is set On.
2. If there are numbers being truncated then the sets borrow flag M2825 On.
3. If the value in (S) equals zero then sets zero flag M1020 On.
4. The conversion outcome is saved in (D) in 16-bit format.

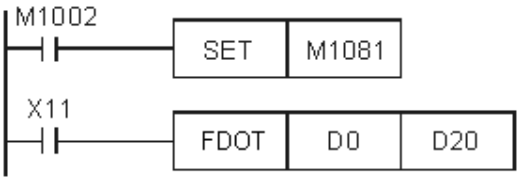
Example I

- In case M1081=Off, convert BIN integer into binary decimal number.
- In case X11=On, convert and save BIN integer in D1 and D0 to binary decimal in D21 and D20.
- In case 32-bit register D0(D1)=K100,000 and X11=On then it is converted to 32-bit floating point number H4735000 and saved in 32-bit register D20(D21).



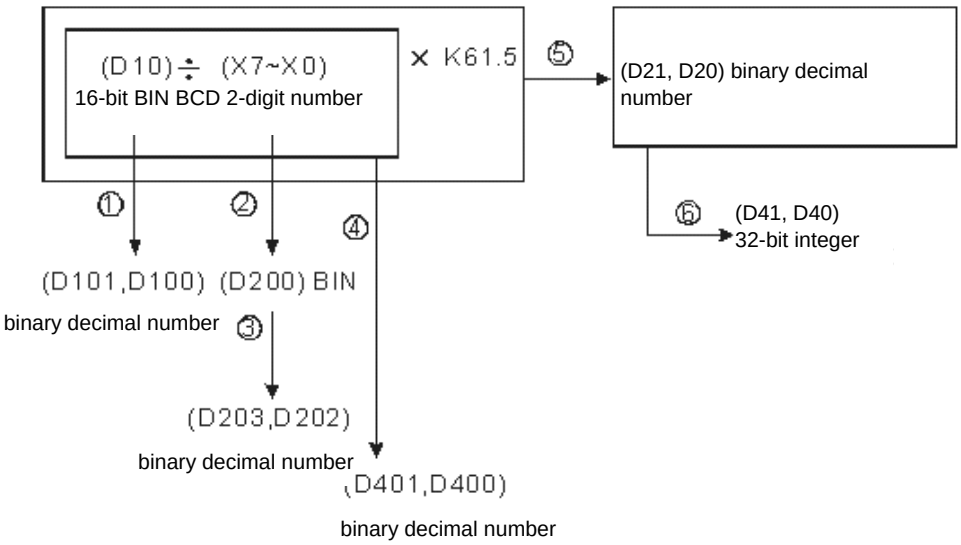
Example 2

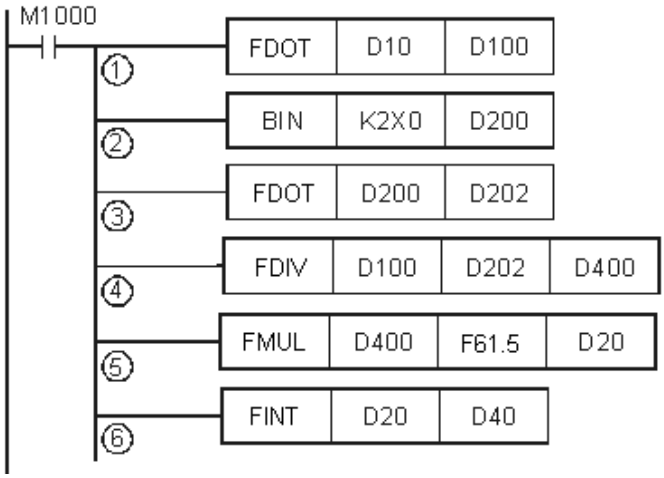
- In case M1081=On it converts the binary decimal number into BIN integer by truncating numbers after decimal point.
- In case X11=On it converts the binary decimal number contained in D1, D0 to BIN integer and save in D21, D20. In case D0(D1) = H47C35000 then it is converted to integer 100,000 and saved in 32-bit register D20(D21).



Example 3

- Compute equations below with application command.





1. Converts the BIN integer contained in D10 to a binary decimal number and saves it in D101, D100.
2. Converts the BCD value in X7~X0 to a BIN number and saves it in D200.
3. Converts the BIN integer contained in D200 to a binary decimal number and saves it in D203, D202.
4. Computes (D101, D100) ÷ (D203, D202) in binary decimal format and saves the binary decimal outcome in D401, D400.
5. Computes (D401, D400) × F61.5 in binary decimal format and saves the binary decimal outcome in D21, D20.
6. Converts the binary decimal number contained in D21, D20 to a BIN integer and saves it in D41, D40.

API		FRAD		(S) (D)	Convert value in degree to radian	Model
65						NC300
						✓

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

Notes on the use of operands:
See the specification included with each model for the use range of specific devices.
This command is valid for the 32-bit command version FRAD only.

16-bit command
- - - -

32-bit command (5 STEP)
FRAD Continuous running type

- Flag signal: M2824 Zero flag
  M2825 Borrow flag
  M2826 Carry flag

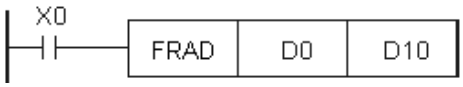
Command description

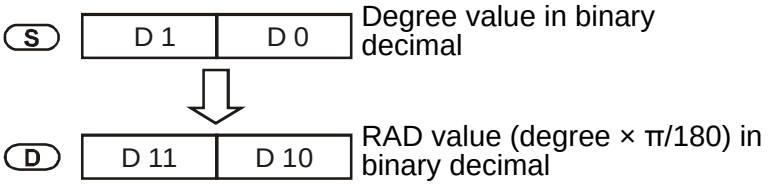
- (S): Source data (degree). (D): Conversion outcome (radian).
- Converts the value in the unit of degrees to radians.

  radian = degree × (π/180)
- In case the absolute value of the outcome is greater than the maximum value of floating point, the carry flag M2826 turns On.
- In case the absolute value of the outcome is less than the minimum value of floating point, the carry flag M2825 turns On.
- In case the outcome equals 0, the zero flag M2824 turns On.

Example

- In case X0=On convert binary floating point degree value contained in (D1, D0) to radian value in binary floating point format and save in (D11, D10).





API		FDEG		(S) (D)	Convert value in radian to degree	Model
66						NC300
						✓

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

Notes on the use of operands:
See the specification included with each model for the use range of specific devices.
This command is valid for the 32-bit command version FDEG only.

16-bit command
- - - -

32-bit command (5 STEP)
DDEG Continuous running type

- Flag signal: M2824 Zero flag
  M2825 Borrow flag
  M2826 Carry flag

Command description

- (S): Source data (radian). (D): Conversion outcome (degree).
- Converts the value in units of radians to degrees.

  degree = radian × (180/π)

- In case the absolute value of the outcome is greater than the maximum value of floating point, the carry flag M2826 turns On.
- In case the absolute value of the outcome is less than the minimum value of floating point, the carry flag M2825 turns On.
- In case the outcome equals 0, the zero flag M2824 turns On.

Example

- In case X0=On it converts the binary floating point radian value contained in (D1, D0) to a degree value in binary floating point format and saves it in (D11, D10).