Coolmay PLC Instructions Programming Manual

Shenzhen Coolmay Technology Co., Ltd

V20.81

Precautions

(Please read before using)

Basic instructions

- Thank you for choosing Coolmay series programmable controller.
- This manual mainly introduces the basics, application instructions and other contents of Coolmay's entire series of programmable controllers.
- Introduction to software and programming, compatible with Mitsubishi GX Developer/ GX Works2, please refer to the relevant manual.
- Please deliver this manual to the end user.

User notice

- Only operators with certain electrical knowledge can perform wiring and other operations on the product. If there is any unclear use, please consult our technical department.
- The examples listed in the manual and other technical materials are for user understanding and reference only, and certain actions are not guaranteed.
- When using this product in combination with other products, please confirm whether it meets the relevant specifications, principles and technical requirements.
- When using this product, please confirm whether it meets the requirements and safety by yourself. If the failure of this product may cause machine failure or loss, please set up backup and safety functions by yourself.

Liability statement

- Although the contents in the manual have been carefully checked, errors are inevitable, and we cannot guarantee complete consistency.
- We will always check the contents of the manual and make corrections in subsequent versions. We welcome your suggestions.
- The contents described in the manual are subject to change without notice.

Contact details

If you have any questions about the use of this product, please contact us.

- Tel: 0755-86950416-842
- Web: www.coolmay.com
- Email: m3@coolmay.com
- Mobile/Skype/Whatsapp: 86 13316892240
- Address: 6th Floor, Building 1, Zhongxing Industrial City, Chuangye Road, Nanshan District, Shenzhen, PRC.

Foreword

The content of the manual

This manual relates to the application of Coolmay's instructions for the entire series of programmable controllers. It mainly introduces the basic instructions, application instructions, and special instructions of programmable controllers. It also describes the main points and principles of programming.

Scope of the manual

Due to space limitations, certain abbreviations may be used in the manual to replace the original names. The names that may be involved are listed in the following table for easy comparison.

Abbreviation	Explanation
3G series PLC	EX3G series HMI/PLC all in one, (D)CX3G series PLC, FX3GC series PLC.
2N series PLC	EX2N series HMI/PLC all in one, (D)CX2N series PLC, FX2NC series PLC
MX2N series PLC	MX2N series HMI/PLC all in one, MX2N series PLC

Note: 1. 3G series products are compatible with Mitsubishi FX3G, FX3U (C), FX3S instructions. If you need to use the FX3U (C) instruction, you need to change the PLC type to FX3U (C) and then copy it to the FX3G program. Finally, you need to download the program in the FX3G type. For the instruction comparison table, please refer to 2.3 Main Application Instructions.

2. 2N series PLC and MX2N series PLC are compatible with Mitsubishi FX2N commands, and support positioning commands DRVI, DRVA, PLSV, etc. need to be compiled in FX1N and then copied to FX2N program.

CONTENT

1 Introduction

Programmable logic controller (PLC), a digital operation controller with a microprocessor for automatic control, can load control instructions into memory for storage and execution at any time. The programmable controller is composed of CPU, instruction and data memory, input / output interface, power supply, digital to analog conversion and other functional units. Early programmable logic controllers only had the function of logic control, so they were named programmable logic controllers. Later, with continuous development, these computer modules with simple functions had already included logic control, timing control, analog control, The names of various functions such as multi-machine communication are also changed to Programmable Controller, but because of its conflict with the abbreviation of PC and personal computer, and because of habits, people still use The term logic controller is programmed, and the acronym PLC is still used.

With the development of electronic technology and the needs of industrial applications, the functions of PLCs are becoming more and more powerful, such as position control and network functions. The input / output signals also include DI (Digital Input), AI (Analog Input), PI (Pulse Input) and NI (Numerical Input), DO (Digital Output), AO (Analog Output), PO (Pulse Output) and NO (Numerical Output), so PLC will still play a decisive role in future industrial control.

1.1 Types of programming languages

PLCs support the following three types of programming languages:

1.1.1 List programming

This method is the basis of programs.

1.1.1.1 Features

In this method, sequence instructions are input in the form of instruction words such as "LD", "AND" and "OUT". This input method is the basis of sequence programs.

1.1.1.2 Example of list display

Step	Instruction	Device number
0000	LD	X000
0001	OR	Y005
0002	ANI	X002
0003	OUT	Y005
...	...	...

1.1.2 Circuit programming

In this method, ladder formats are drawn on the graphic screen.

In a circuit program, a sequence circuit is drawn on the graphic screen by sequence formats and device numbers. Because a sequence circuit is expressed with contact symbols and coil symbols, the contents of a program can be understood easily.

In the circuit display status, the PLC operations can be monitored.

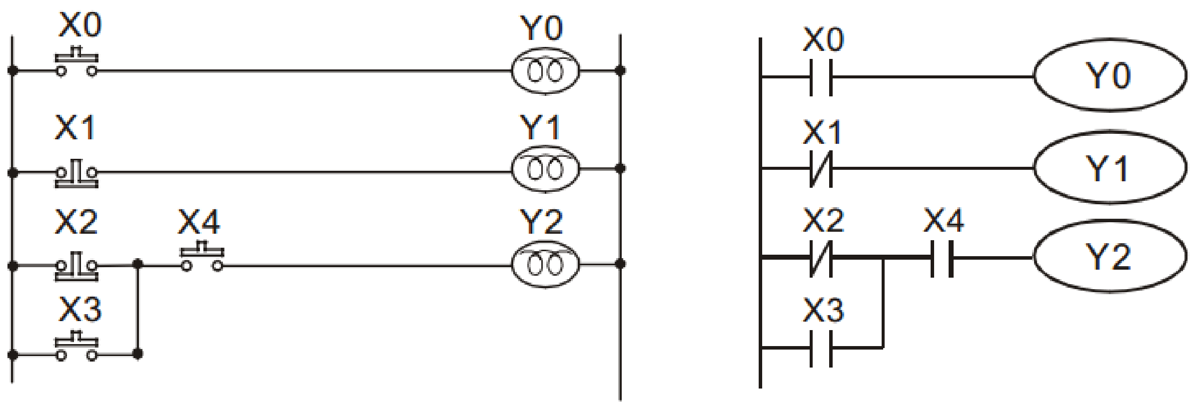
1.1.2.1 Working principle of ladder diagram

Ladder diagram is an automatic control language developed during World War II. It is the oldest and most widely used automatic control language. Initially, there were only A (normally open) contacts, B (normally closed) contacts, and output coils. , Timers, counters and other basic mechanisms and devices, until the appearance of the programmable controller PLC, the devices that can be represented in the ladder diagram, in addition to the above, also add devices such as differential contacts, holding coils and traditional distribution boards can not achieve Application instructions, such as numerical operations such as addition, subtraction, multiplication and division.

No matter the traditional ladder diagram or PLC ladder diagram, the working principle is the same, but the traditional ladder diagram is represented by a symbol that is closer to the entity in the symbolic representation, while the PLC uses a simpler and easier to represent on the computer or report. Ladder diagram logic can be divided into two kinds of combinational logic and sequential logic, which are described as follows:

1.1.2.2 Combination logic:

The traditional ladder diagram (left) and the PLC ladder diagram (right) represent examples of combinational logic.



Line 1: Use a normally open switch X0 (NO: Normally Open), also known as the "A" switch or contact. Its characteristic is that in normal (not pressed), its contacts are open (OFF), so Y0 is not conductive, but when the switch is activated (pressed the button), its contacts become conductive (ON), Therefore, Y0 is turned on.

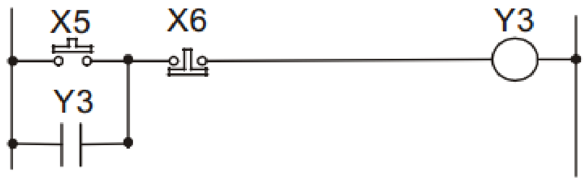
Line 2: Use a normally closed switch X1 (NC: Normally Close), also known as the "B" switch or contact. Its characteristic is that at normal times, its contact is conductive, so Y1 is conductive, and the When the switch is activated, its contact becomes an open circuit instead, so Y1 does not conduct.

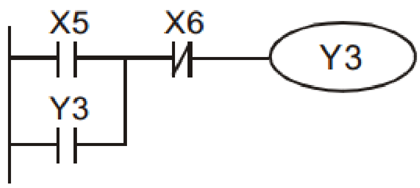
Line 3: For the application of combined logic output of more than one input device, its output Y2 will only be turned on when X2 is not in action or X3 is in action and X4 is in action.

1.1.2.3 Sequential logic:

Sequential logic is a loop with a feedback structure, that is, the loop output result is pulled back as the input condition. In this way, under the same input condition, different output results will be obtained due to the difference in the previous state or action sequence.

The traditional ladder diagram (left) and PLC ladder diagram (right) respectively represent examples of sequential logic.





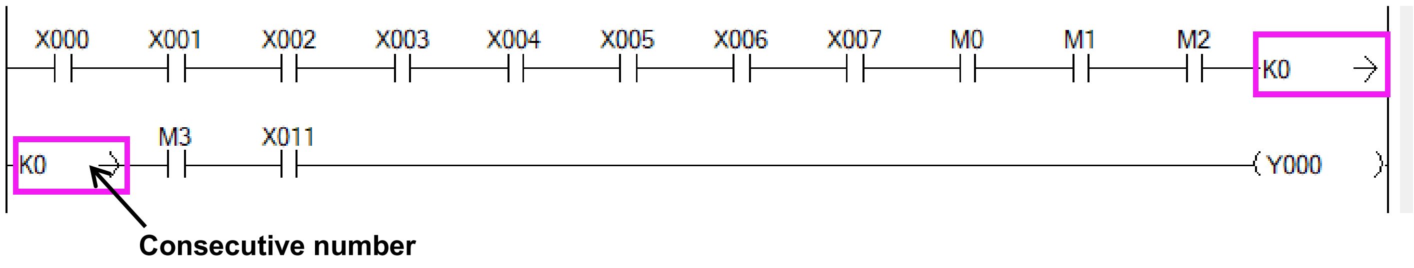
When the circuit is just connected to the power supply, although the X6 switch is ON, the X5 switch is OFF, so Y3 does not operate. After the start switch X5 is pressed, Y3 acts. Once Y3 acts, even if the start switch is released (X5 becomes OFF), Y3 can continue to maintain the action (this is the self-holding circuit) because of its own contact feedback. The actions can be expressed in the following table:

Status / Sequence	X5 ON/OFF	X6 ON/OFF	Y3 ON/OFF
1	No action	No action	OFF
2	Action	No action	ON
3	No action	No action	ON
4	No action	Action	OFF
5	No action	No action	OFF

It can be seen from the above table that under different sequences, although the input states are completely consistent, the output results may also be different. For example, the action sequences 1 and 3 in the table, the X5 and X6 switches are not activated, and under the condition of state 1, Y3 is OFF, but Y3 is ON in state 3. This Y3 output state is pulled back as an input (so-called feedback) to give the loop a sequential control effect is the main characteristic of the ladder circuit.

1.1.2.4 Editing points of PLC ladder diagram

The program editing method is from the left bus to the right bus. The editing of one line is followed by the next line. The number of contacts in a line can be up to 11, if it is not enough, continuous lines will be generated to continue the connection, and then continue to connect more For devices, consecutive numbers are automatically generated, and the same input point can be used repeatedly. As shown below:



The operation of the ladder program is scanning from the upper left to the lower right. The coil and application instruction operation box belong to the output processing, and are placed at the far right in the ladder diagram.

1.1.3 SFC(STL<step ladder >)programming

Sequence control using the SFC (sequential function chart) is available in PLCs.

In SFC programs, the role of each process and the overall control flow can be expressed easily based on machine operations, so sequence design is easy. Accordingly, machine operations can be easily transmitted to any person, and created programs are efficient in maintenance, specifications changes and actions against problems.

1.1.3.1 Features

In SFC programs, each process performed by the machine is expressed by a state relay.

1.1.3.2 Interchangeability of SFC procedures and other procedures

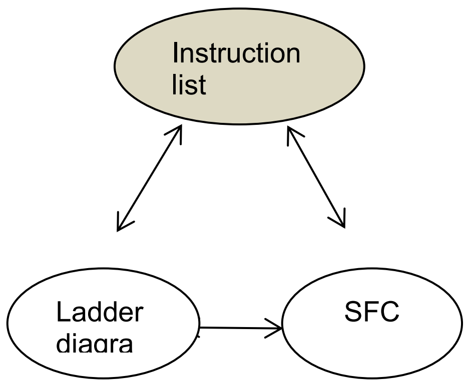
The instruction list program and ladder program that can be converted to each other can be converted into SFC charts if they are prepared according to certain rules.

Note: When Coolmay PLC uses SFC programming, it is not allowed to use positioning instructions or other pulsed instructions in SFC.

1.2 Program interchangeability

The sequence programs created by the above three methods are saved in the program memory of the programmable controller through instructions (the contents of the instruction table programming).

• Programs created using the various input methods shown below can be converted and displayed and edited.



Note: Our PLC programming is mainly based on ladder diagram programming. (Does not support structured engineering, does not support the use of tags)

2 Instruction List

2.1 Basic Instructions

Mnemonic	Name	Function	Applicable devices	Program step
Contact Instruction				
LD	Load	Initial logical operation contact type NO (normally open)	X,Y,M,S,T,C,D □.b	1
LDI	Load Inverse	Initial logical operation contact type NC (normally closed)	X,Y,M,S,T,C,D □.b	1
LDP	Load Pulse	Initial logical operation of Rising edge pulse	X,Y,M,S,T,C,D □.b	2
LDF	Load Falling Pulse	Initial logical operation of Falling/trailing edge pulse	X,Y,M,S,T,C,D □.b	2
AND	AND	Serial connection of NO (normally open) contacts	X,Y,M,S,T,C,D □.b	1
ANI	AND Inverse	Serial connection of NC (normally closed) contacts	X,Y,M,S,T,C,D □.b	1
ANDP	AND Pulse	Serial connection of Rising edge pulse	X,Y,M,S,T,C,D □.b	2
ANDF	AND Falling Pulse	Serial connection of Falling/trailing edge pulse	X,Y,M,S,T,C,D □.b	2
OR	OR	Parallel connection of NO (normally open) contacts	X,Y,M,S,T,C,D □.b	1
ORI	OR Inverse	Parallel connection of NC (normally closed) contacts	X,Y,M,S,T,C,D □.b	1
ORP	OR Inverse	Parallel connection of NC (normally closed) contacts	X,Y,M,S,T,C,D □.b	2
ORF	OR Falling Pulse	Parallel connection of Falling/trailing edge pulse	X,Y,M,S,T,C,D □.b	2
Connection Instruction				
ANB	AND Block	Serial connection of multiple parallel circuits	—	1
ORB	OR Block	Parallel connection of multiple contact circuits	—	1
MPS	Memory Point Store	Stores the current result of the internal PLC operations	—	1

MRD	Memory Read	Reads the current result of the internal PLC operations	—	1
MPP	Memory POP	Pops (recalls and removes) the currently stored result	—	1
INV	Inverse	Invert the current result of the internal PLC operations	—	1
MEP	M.E.P	Conversion of operation result to leading edge pulse	—	
MEF	M.E.F	Conversion of operation result to trailing edge pulse	—	
Out Instruction				
OUT	OUT	Final logical operation type coil drive	Y,M,S,T,C,D □.b	Note 1
SET	SET	SET Bit device latch ON	Y,M,S,D □.b	Note 2
RST	Reset	RESET Bit device OFF	Y,M,S,T,C,D,V,Z,D □.b	
PLS	Pulse	Rising edge pulse	Y,M	
PLF	Pulse Falling	Falling/trailing edge pulse	Y,M	
Master Control Instruction				
MC	Master Control	Denotes the start of a master control block	Y,M	3
MCR	Master Control Reset	Denotes the end of a master control block	—	2
Other Instruction				
NOP	No Operation	No operation or null step	—	1
End Instruction				
END	END	Program END, I/O refresh and Return to Step 0	—	1

2.2 Step Ladder Instructions

Mnemonic	Name	Function	Applicable devices	Program step
STL	Step Ladder	Starts step ladder	S	1
RET	Return	Completes step ladder	—	1

2.3 Applied Instructions

FNC NO.	Mnemonic	Function	Applicable PLC			
			3G series PLC		2N series PLC	MX2N series PLC
			FX3G	FX3U(C)		
00	CJ	Conditional Jump	★	★	★	★
01	CALL	Call Subroutine	★	★	★	★
02	SRET	Subroutine Return	★	★	★	★
03	IRET	Interrupt Return	★	★		★
04	EI	Enable Interrupt	★	★		★
05	DI	Disable Interrupt	★	★		★
06	FEND	Main Routine Program End	★	★	★	★
07	WDT	Watchdog Timer Refresh	★	★	★	★
08	FOR	Start a FOR/NEXT Loop	★	★	★	★
09	NEXT	End a FOR/NEXT Loop	★	★	★	★
10	CMP	Compare	★	★	★	★
11	ZCP	Zone Compare	★	★	★	★
12	MOV	Move	★	★	★	★
13	SMOV	Shift Move	★	★		★
14	CML	Complement	★	★	★	★
15	BMOV	Block Move	★	★	★	★
16	FMOV	Fill Move	★	★	★	★
17	XCH	Exchange		★	★	★
18	BCD	Conversion to Binary Coded Decimal	★	★	★	★
19	BIN	Conversion to Binary	★	★	★	★
20	ADD	Addition	★	★	★	★
21	SUB	Subtraction	★	★	★	★
22	MUL	Multiplication	★	★	★	★
23	DIV	Division	★	★	★	★
24	INC	Increment	★	★	★	★
25	DEC	Decrement	★	★	★	★
26	WAND	Logical Word AND	★	★	★	★
27	WOR	Logical Word OR	★	★	★	★
28	WXOR	Logical Exclusive OR	★	★	★	★
29	NEG	Negation		★	★	★

30	ROR	Rotation Right	★	★	★	★
31	ROL	Rotation Left	★	★	★	★
32	RCR	Rotation Right with Carry		★	★	★
33	RCL	Rotation Left with Carry		★	★	★
34	SFTR	Bit Shift Right	★	★	★	★
35	SFTL	Bit Shift Left	★	★	★	★
36	WSFR	Word Shift Right	★	★	★	★
37	WSFL	Word Shift Left	★	★	★	★
38	SFWR	Shift write [FIFO/FILO control]	★	★	★	★
39	SFRD	Shift Read [FIFO Control]	★	★	★	★
40	ZRST	Zone Reset	★	★	★	★
41	DECO	Decode	★	★	★	★
42	ENCO	Encode	★	★	★	★
43	SUM	Sum of Active Bits	★	★	★	★
44	BON	Check Specified Bit Status	★	★	★	★
45	MEAN	Mean	★	★	★	★
46	ANS	Timed Annunciator Set	★	★		★
47	ANR	Annunciator Reset	★	★		★
48	SQR	Square Root		★	★	★
49	FLT	Conversion to Floating Point	★	★	★	★
50	REF	Refresh	★	★	★	★
51	REFF	Refresh and Filter Adjust		★		
52	MTR	Input Matrix	★	★		
53	HSCS	High Speed Counter Set	★	★		
54	KSCR	High Speed Counter Reset	★	★		
55	HSZ	High Speed Counter Zone Compare	★	★		
56	SPD	Speed Detection	★	★	★	★
57	PLSY	Pulse Y Output	★	★	★	★
58	PWM	Pulse Width Modulation	★	★	★	★
59	PLSR	Acceleration/Deceleration Setup	★	★	★	★
60	IST	Initial State	★	★		
61	SER	Search a Data Stack	★	★		★
62	ABSD	Absolute Drum Sequencer	★	★		
63	INCD	Incremental Drum Sequencer	★	★		
64	TTMR	Teaching Timer		★		

65	STMR	Special Timer		★		
66	ALT	Alternate State	★	★	★	★
67	RAMP	Ramp Variable Value	★	★	★	★
68	ROTC	Rotary Table Control		★		
69	SORT	SORT Tabulated Data		★		
70	TKY	Ten Key Input		★		
71	KHY	Hexadecimal Input		★		
72	DSW	Digital Switch (Thumbwheel Input)	★	★		
73	SEGD	Seven Segment Decoder		★	★	★
74	SEGL	Seven Segment with Latch	★	★		
75	ARWS	Arrow Switch		★		
76	ASC	ASCII Code Data Input		★		
77	PR	Print (ASCII Code)		★		
78	FROM	Read From A Special Function Block	★	★		MODBUS-RTU master station function, used to read/write slave station data
79	TO	Write To A Special Function Block	★	★		
80	RS	Serial Communication	★	★	★	★
81	PRUN	Parallel Run (Octal Mode)	★	★		
82	ASCI	Hexadecimal to ASCII Conversion	★	★	★	★
83	HEX	ASCII to Hexadecimal Conversion	★	★	★	★
84	CCD	Check Code	★	★		★
85	VRRD	Volume Read	★			
86	VRSC	Volume Scale	★			
87	RS2	Serial Communication 2	★	★		
88	PID	PID Control Loop	★	★	★	
102	ZPUSH	Batch Store of Index Register		★		
103	ZPOP	Batch POP of Index Register		★		
110	ECMP	Floating Point Compare	★	★	★	★
111	EZCP	Floating Point Zone Compare		★	★	★
112	EMOV	Floating Point Move	★	★		
116	ESTR	Floating Point to Character String Conversion		★		
117	EVAL	Character String to Floating Point Conversion		★		
118	EBCD	Floating Point to Scientific Notation Conversion		★	★	★

119	EBIN	Scientific Notation to Floating Point Conversion		★	★	★
120	EADD	Floating Point Addition	★	★	★	★
121	ESUB	Floating Point Subtraction	★	★	★	★
122	EMUL	Floating Point Multiplication	★	★	★	★
123	EDIV	Floating Point Division	★	★	★	★
124	EXP	Floating Point Exponent		★		
125	LOGE	Floating Point Natural Logarithm		★		
126	LOG10	Floating Point Common Logarithm		★		
127	ESQR	Floating Point Square Root	★	★	★	★
128	ENEG	Floating Point Negation		★		
129	INT	Floating Point to Integer Conversion	★	★	★	★
130	SIN	Floating Point Sine		★	★	★
131	COS	Floating Point Cosine		★	★	★
132	TAN	Floating Point Tangent		★	★	★
133	ASIN	Floating Point Arc Sine		★		
134	ACOS	Floating Point Arc Cosine		★		
135	ATAN	Floating Point Arc Tangent		★		
136	RAD	Floating Point Degrees to Radians Conversion		★		
137	DEG	Floating Point Radians to Degrees Conversion		★		
140	WSUM	Sum of Word Data		★		
141	WTOB	WORD to BYTE		★		
142	BTOW	BYTE to WORD		★		
143	UNI	4-bit Linking of Word Data		★		
144	DIS	4-bit Grouping of Word Data		★		
147	SWAP	Byte Swap		★	★	★
149	SORT2	Sort Tabulated Data 2		★		
150	DSZR	DOG Search Zero Return	★	★		
151	DVIT	Interrupt Positioning		★		
152	TBL	Batch Data Positioning Mode	★	★		

155	ABS	Batch Data Positioning Mode	★	★	★	★
156	ZRN	Zero Return	★	★	★	★
157	PLSV	Variable Speed Pulse Output	★	★	★	★
158	DRVI	Drive to Increment	★	★	★	★
159	DRVA	Drive to Absolute	★	★	★	★
160	TCMP	RTC Data Compare	★	★	★	★
161	TZCP	RTC Data Zone Compare	★	★	★	★
162	TADD	RTC Data Addition	★	★	★	★
163	TSUB	RTC Data Subtraction	★	★	★	★
164	HTOS	Hour to Second Conversion		★		
165	STOH	Second to Hour Conversion		★		
166	TRD	Read RTC data	★	★	★	★
167	TWR	Set RTC data	★	★	★	★
169	HOUR	Hour Meter	★	★	★	★
170	GRY	Decimal to Gray Code Conversion	★	★		
171	GBIN	Gray Code to Decimal Conversion	★	★		
176	RD3A	Read form Dedicated Analog Block	★Modbus communication to read slave data (Note: The original analog block readout function is not available)			★
177	WR3A	Write to Dedicated Analog Block	★Modbus communication to write data to the slave (Note: The original analog block write function is not available)			★
182	COMRD	Read Device Comment Data		★		
184	RND	Random Number Generation		★		
186	DUTY	Timing Pulse Generation		★		
188	CRC	Cyclic Redundancy Check		★		
189	HCMOV	High Speed Counter Move		★		
192	BK+	Block Data Addition		★		
193	BK-	Block Data Subtraction		★		
194	BKCMP=	Block Data Compare**S1.=S2.**		★		

195	BKCMP>	Block Data Compare**S1. > S2.**		★		
196	BKCMP<	Block Data Compare**S1. < S2.**		★		
197	BKCMP<>	Block Data Compare**S1.≠S2.**		★		
198	BKCMP<=	Block Data Compare**S1.≤S2.**		★		
199	BKCMP>=	Block Data Compare**S1.≥S2.**		★		
200	STR	BIN to Character String Conversion		★		
201	VAL	Character String to BIN Conversion		★		
202	$+	Link Character Strings		★		
203	LEN	Character String Length Detection		★		
204	RIGHT	Extracting Character String Data from the Right		★		
205	LEFT	Extracting Character String Data from the Left		★		
206	MIDR	Random Selection of Character Strings		★		
207	MIDW	Random Replacement of Character Strings		★		
208	INSTR	Character string search		★		
209	$MOV	Character String Transfer		★		
210	FDEL	Deleting Data from Tables		★		
211	FINS	Inserting Data to Tables		★		
212	POP	Shift Last Data Read [FILO Control]		★		
213	SFR	Bit Shift Right with Carry		★		
214	SFL	Bit Shift Left with Carry		★		
224	LD=	Load Compare LD **S1.=S2.**	★	★	★	★
225	LD>	Load Compare LD **S1.>S2.**	★	★	★	★
226	LD<	Load Compare LD **S1.<S2.**	★	★	★	★
228	LD<>	Load Compare LD **S1.<>S2.**	★	★	★	★
229	LD<=	Load Compare LD **S1.<=S2.**	★	★	★	★
230	LD>=	Load Compare LD **S1.>=S2.**	★	★	★	★
232	AND=	Load Compare AND **S1.=S2.**	★	★	★	★
233	AND>	Load Compare AND **S1.>S2.**	★	★	★	★
234	AND<	Load Compare AND **S1.<S2.**	★	★	★	★
236	AND<>	Load Compare AND **S1.<>S2.**	★	★	★	★
237	AND<=	Load Compare AND **S1.<=S2.**	★	★	★	★

238	AND>=	Load Compare AND **S1.>=S2.**	★	★	★	★
240	OR=	Load Compare OR **S1.=S2.**	★	★	★	★
241	OR>	Load Compare OR **S1.>S2.**	★	★	★	★
242	OR<	Load Compare OR **S1.<S2.**	★	★	★	★
244	OR<>	Load Compare OR **S1.<>S2.**	★	★	★	★
245	OR<=	Load Compare OR **S1.<=S2.**	★	★	★	★
246	OR>=	Load Compare OR **S1.>=S2.**	★	★	★	★
256	LIMIT	Limit Control		★		
257	BAND	Dead Band Control		★		
258	ZONE	Zone Control		★		
259	SCL	Scaling (Coordinate by Point Data)		★		
260	DABIN	Decimal ASCII to BIN Conversion		★		
261	BINDA	BIN to Decimal ASCII Conversion		★		
269	SCL2	Scaling 2 (Coordinate by X/Y Data)		★		
280	HSCT	High Speed Counter Compare With Data Table		★		

3 Devices in Detail

3.1 Device Number List

3.1.1 3G series PLC

Device name	Description		
I/O relay			
Input relay	X000~X047	40 points	Device numbers are octal. The total number of inputs and outputs is 80.
Output relay	Y000~Y047	40 points	
Auxiliary relay			
General type	M0~M383	384 points	
EEPROM retention	M384~M1535	1152 points	
General type	M1536~M7679	6144 points	
Special type	M8000~M8511	512 points	
State relay			
Initial state (EEPROM retention)	S0～S9	10 points	
EEPROM retention	S10～S999	990 points	
General type	S1000～S4095	3096 points	
Timer (on-delay timer)			
100ms	T0～T199	200 points	0.1～3,276.7 sec
10ms※1	T200～T245	46 points	0.01～327.67 sec
Retentive type for 1 ms (EEPROM retention)	T246～T249	4 points	0.001～32.767 sec
Retentive type for 100 ms (EEPROM retention)	T250～T255	6 points	0.1～3,276.7 sec
1ms	T256～T319	64 points	0.001~32.767 sec
Counter			
General type up counter (16 bits)	C0～C15	16 points	Counts 0 to 32,767
EEPROM retention up counter (16 bits)	C16～C199	184 points	Counts 0 to 32,767
General type bi-directional counter (32 bits)	C200～C219	20 points	−2,147,483,648 to +2,147,483,647 counts
EEPROM retention bi-directional counter	C220～C234	15 points	−2,147,483,648 to +2,147,483,647 counts

(32 bits)			
High speed counter			
1-phase 1-counting input Bi-directional (32 bits) (EEPROM retention)	C235 ~ C245	−2,147,483,648 to 2,147,483,647 counts Software counter 1 phase: max 6 channels, max 60 KHz each Two-phase: 1-time frequency: up to 2-3 channels, maximum frequency 60KHz M8198 is the 4 times frequency mark of C251/C252 4-times frequency: max 2 channels, max frequency 24KHz M8199 is the 4 times frequency mark of C253/C255	
1-phase 2-counting input Bi-directional (32 bits) (EEPROM retention)	C246 ~ C250		
2-phase 2-counting input Bi-directional (32 bits) (EEPROM retention)	C251 ~ C255		

Device name	Description		
Data register (32 bits when used in pair form)			
General type (16 bits)	D0 ~ D127	128 points	
EEPROM retention (16 bits)	D128 ~ D7999	7872 points	
Special type (16 bits)	D8000 ~ D8511	512 points	
Index type (16 bits)	V0 ~ V7,Z0 ~ Z7	16 points	
Extension register/Extension file register			
Special type (16 bits)	R0 ~ R22999	23000 points Support blackout	
	R23000 ~ R23999	1000 points For internal use	
Pointer			
For jump and branch call	P0 ~ P255 P0 ~ P1280	256 points 1281 points (version 26232 and above)	For CJ and CALL instructions
Input interrupt	I0□□ ~ I5□□	6 points	
Timer interrupt	I6□□ ~ I8□□	3 points	
Counter interrupt	I0□□ ~ I0□□	6 points	
Nesting			
For master control	N0 ~ N7	8 points	For MC instruction
Constant			
Decimal (K)	16 points	-32,768 ~ +32,767	
	32 points	-2,147,483,648 ~ +2,147,483,647	
Hexadecimal (H)	16 points	0000 ~ FFFF	

	32 points	00000000～FFFFFFFF
Real number (E)	32 points	-1.0×2128～-1.0×2-126,0,1.0×2-126～1.0×2128 Both the decimal point expression and the exponent expression are available.

3.1.2 2N series PLC

Item		Substance
Operation control method		Cyclic scanning through stored programs
Input and output control method		Batch processing (when executing END instruction), input and output refresh, pulse capture
Programming language		Logic ladder diagram and instruction list (compatible with Mitsubishi software FXGP_WIN-C)
Operation time	Basic instructions	0.08μs
	Application instruction	10-30μs
RAM	Built-in	8000 step EEPROM
	Storage box	
Instruction	Basic sequence instruction	27
	Step ladder instructions	2
	Application instruction	94
Auxiliary relay	general	500 points M0 - M499
	lock	1036 points M500-M1535
	special	256 points M8000 -M8255
status	general	500 points S0 to S499
	initial	10 points S000-S009
	lock	500 points S500 - S999
Timer	100 millisecond	200 points T0-T199
	10 millisecond	46 points T200-T245
	1 millisecond integration	4 points T246-T249
	100 millisecond integration	6 points T250- T255
counter	General 16 bits	100 points C0-C99
	Lock 16 bits	100 points C100-C199
	General 32-bit	
	Lock 32 bits	35 points C200-C234

High-speed counter	Simplex	Max 6 points: C235-X0 C236-X1 C237-X7 C238-X3 C239-X4 C240-X5; Normal 2 points: C235-X0 C238-X3
	A/B phase	Max 3 points: C251-X0/X1 C253-X3/X4 C254-X10/X11 ; Normal 2 points: C251-X0/X1 C253-X3/X4
Data register (D.V.Z)	General	200 points D0-D199
	Retentive	800 points D200-D999
	File register	
	External regulation	
	special	256 points D8000-D8255
	Index	16 points V0-V7 Z0-Z7
pointer	JUMP,CALL	128 points P0-P127
	Input interrupt	
Nested	For master	8 points N0-N7
Constant	Decimal K	16 points: -32768 - +32767
		32 points: -2147483648 - +2147483647
	Hexadecimal H	16 points: 0000-FFFF
		32 points: 00000000-FFFFFFFF

3.1.3 MX2N series PLC

Input X	X0~X47 40 points		Output Y	Y0~Y47 40 points	
Auxiliary relay M	M0~M499 500 points (general)	M500~M1535 1036 points (For keeping)		M8000~M8255 255 points (For special)	
Status relay S	S0~S9 10 points (For keeping state)		S10~S999 990 points (For keeping)		
Timer T	T0~T199 200 points 100ms	T200~T245 46 points 10ms	T246~T249 4 points 1ms accumulation	T250~T255 6 points 100ms accumulation	
Counter C	16-bit up counter		32-bit up-down counter		
	C0~C15 16 points (general)	C16~C199 184 points (For keeping)	C200~C219 20 points (general)	C220~C234 15 points (For keeping)	C235~C255 20 points (Keep high speed)
Register D.V.Z	D0~D199 200 点 (general)	D200~D7999 7800 points (For keeping)	D8000~D8195 196 points (For special, keeping)	D8196~D8255 59 points (For special)	V0~V7 Z0~Z7 16 points (For indexing)
Nested pointer	N0~N7 8 points (For master)	P0~P127 128 points (Jump, subroutine)		IO 口口~I5 口口 6 points (For external interrupt)	
constant	K (decimal	16bit-32768~32767		32bit-2147483648~2147483647	

	number)		
	H (hexadecimal number)	16bit 0~FFFF	32bit 0~FFFFFFFF

3.2 Special device number and content

3.2.1 3G series PLC

Number	Content	Remarks	Number	Content	Remarks
M8000	Closed during RUN		M8224	C224 Up/down counting action	ON: Decrement OFF: Incremental action
M8001	Open when RUN		M8225	C225 Up/down counting action	
M8002	After RUN, output one scan cycle ON		M8226	C226 Up/down counting action	
M8003	After RUN, output a scan cycle of OFF		M8227	C227 Up/down counting action	
M8011	Oscillate with a period of 10ms		M8228	Start the handwheel function	
M8012	Oscillate with a period of 100ms		M8229	C229 Up/down counting action	
M8013	Oscillate with a period of 1s		M8230	C230 Up/down counting action	
M8014	Oscillate with a period of 1min		M8231	C231 Up/down counting action	
M8020	Zero mark		M8232	C232 Up/down counting action	
M8021	Borrow mark		M8233	C233 Up/down counting action	
M8022	Carry flag		M8234	C234 Up/down counting action	
M8024	Specify BMOV direction		M8235	C235 Up/down counting action	ON: Decrement OFF: Incremental action
M8028	Allow interruption during instruction execution		M8236	C236 Up/down counting action	
M8029	Instruction execution end flag		M8237	C237 Up/down counting action	
M8031	All non-retained memory is cleared		M8238	C238 Up/down counting action	
M8032	Keep all memory clear		M8239	C239 Up/down counting action	
M8033	Memory remains stopped		M8240	C240 Up/down counting action	
M8034	Suppress all output		M8241	C241 Up/down counting action	
M8035	Forced RUN mode		M8242	C242 Up/down counting action	
M8036	Forced RUN instruction		M8243	C243 Up/down counting action	
M8037	Force STOP instruction		M8244	C244 Up/down counting action	
M8045	Disable reset of all outputs		M8245	C245 Up/down counting action	
M8046	STL state action		M8246	C246 Up/down counting action	ON: Decrement OFF: Incremental action
M8047	STL control is effective		M8247	C247 Up/down counting action	
M8048	Signal alarm action		M8248	C248 Up/down counting action	
M8049	Signal alarm is effective		M8249	C249 Up/down counting action	
M8050	Input interrupt (I00 port		M8250	C250 Up/down counting action	

Number	Content	Remarks	Number	Content	Remarks
	prohibited)				
M8051	Input interrupt (I10 port prohibited)		M8251	C251 Up/down counting action	
M8052	Input interrupt (I20 port prohibited)		M8252	C252 Up/down counting action	
M8053	Input interrupt (I30 port prohibited)		M8253	C253 Up/down counting action	
M8054	Input interrupt (I40 port prohibited)		M8254	C254 Up/down counting action	
M8055	Input interrupt (I50 port prohibited)		M8255	C255 Up/down counting action	
M8056	Timer interrupt (I6 port is prohibited)		M8340	The first pulse operation monitoring	
M8057	Timer interrupt (I7 port is prohibited)		M8342	Interpolation mode flag	26233 and Previous version
M8058	Timer interrupt (I8 port is prohibited)		M8343	Interpolation mode flag	
M8059	Counter interrupt disable		M8344	Interpolated relative/absolute coordinate flag	
M8060	I/O composition error		M8348	Interpolate clockwise and counterclockwise flags	26233 and Previous version
M8061	PLC hardware error		M8341	Y000 clear signal output function is effective	26234 and Later version
M8062	Serial communication error 0		M8342	Y000 specifies the origin return direction	
M8063	Serial communication error 1		M8343	Y000 forward limit	
M8064	Parameter error		M8344	Y000 reverse limit	
M8065	Grammatical errors		M8345	Y000 Near-point DOG signal logic inversion	
M8066	Loop error		M8346	Y000 Zero signal logic reversal	
M8067	Operation error		M8347	Y000 interrupt signal logic inversion	
M8068	Operation error latch		M8348	Y000 positioning command driving	
M8069	I/O bus detection		M8349	The first pulse stop bit	
M8075	Sampling trace preparation start instruction		M8350	Second channel pulse operation monitoring	
M8076	Sampling trace execution start instruction		M8351	Y001 clear signal output function is effective	
M8077	Prompt control during sampling and tracking		M8352	Y001 specifies the origin return direction	
M8078	Sampling and tracking		M8353	Y001 forward limit	

Number	Content	Remarks	Number	Content	Remarks
	execution ended				
M8079	Sampling and tracking system area		M8354	Y001 reverse limit	
M8120	Not available		M8355	Y001 Near-point DOG signal logic inversion	
M8121	RS/RS2 command send standby flag		M8356	Y001 Zero signal logic reversal	
M8122	RS/RS2 command sending request		M8357	Y001 interrupt signal logic inversion	
M8123	RS/RS2 command reception end flag		M8358	Y001 positioning command driving	
M8124	RS/RS2 command data receiving		M8359	Second pulse stop bit	
M8125	Activating mark of MODBUS and Mitsubishi functions		M8360	Third pulse operation monitoring	
M8128	RD3A/WR3A received correct flag		M8361	Y002 clear signal output function is valid	
M8129	RD3A/WR3A communication timeout flag		M8362	Y002 specifies the origin return direction	
M8151	Fifth pulse operation control		M8363	Y002 forward limit	
M8152	Sixth pulse operation control		M8364	Y002 Reverse limit	
M8153	Seventh pulse operation control		M8365	Y002 Near-point DOG signal logic inversion	
M8154	Eighth pulse operation control		M8366	Y002 Zero signal logic reversal	
M8160	XCH SWAP function		M8367	Y002 interrupt signal logic inversion	
M8161	8-bit processing mode	26234 and Later version	M8368	Y002 positioning command driving	
M8170	Input X000 pulse capture		M8369	Third pulse stop bit	
M8171	Input X001 pulse capture		M8370	4th pulse operation monitoring	
M8172	Input X002 pulse capture		M8371	Y003 clear signal output function is effective	
M8173	Input X003 pulse capture		M8372	Y003 specifies the origin return direction	
M8174	Input X004 pulse capture		M8373	Y003 forward limit	
M8175	Input X005 pulse capture		M8374	Y003 Reverse limit	
M8176	Input X006 pulse capture		M8375	Y003 Near-point DOG signal logic inversion	
M8177	Input X007 pulse capture		M8376	Y003 Zero signal logic reversal	
M8192	Enable flag of programming port protocol and other	Serial port 3	M8377	Y003 Interrupt signal logic inversion	

Number	Content	Remarks	Number	Content	Remarks
	protocols				
M8196	Enable flag of programming port protocol and other protocols	Serial port 2	M8378	Y003 positioning command driving	
M8198	4 octave signs for C251, C252		M8379	Fourth pulse stop bit	
M8199	4 octave signs for C253, C255				
M8200	C200 Up/down counting action		M8401	RS2 Command sending standby flag	
M8201	C201 Up/down counting action		M8402	RS2 Instruction sending request	
M8202	C202 Up/down counting action		M8403	RS2 Command received end flag	
M8203	C203 Up/down counting action		M8404	RS2 Command data receiving	
M8204	C204 Up/down counting action		M8405	RS2 Command data set ready flag	
M8205	C205 Up/down counting action		M8408	RD3A/WR3AReceive complete flag	
M8206	C206 Up/down counting action		M8409	RD3A/WR3A Communication timeout flag	
M8207	C207 Up/down counting action		M8421	RS2 command to send standby flag	
M8208	C208 Up/down counting action		M8422	RS2 command sending request	
M8209	C209 Up/down counting action		M8423	RS2 command receiving end flag	
M8210	C210 Up/down counting action		M8424	RS2 command data receiving	
M8211	C211 Up/down counting action		M8425	RS2 command data transmission completion flag	
M8212	C212 Up/down counting action	ON: Decrement OFF: Incremental action	M8426	RS command master-slave and multi-machine mode flags	
M8213	C213 Up/down counting action		M8427	CAN data standard frame and extended frame flag	
M8214	C214 Up/down counting action		M8428	CAN communication MODBUS response correct flag	
M8215	C215 Up/down counting action		M8429	Communication timeout	
M8216	C216 Up/down counting action		M8432	Interpolation mode flag	
M8217	C217 Up/down counting action		M8433	Interpolation mode flag	26235 and Later version
M8218	C218 Up/down counting action		M8434	Interpolated relative/absolute coordinate flag	
M8219	C219 Up/down counting action		M8435	Interpolate clockwise and counterclockwise flags	
M8220	C220 Up/down counting action		M8450	Fifth pulse stop bit	
M8221	C221 Up/down counting action		M8451	Sixth pulse stop bit	
M8222	C222 Up/down counting action		M8452	Seventh pulse stop bit	
M8223	C223 Up/down counting action		M8453	Eighth pulse stop bit	

3.2.2 2N series PLC refer to 3.3

3.2.3 MX2N series PLC

Number	content	Number	content
M8000	Operation monitoring contact	M8112	Optional 1 channel weighing function is activated
M8001	Operation monitoring counter contact	M8113	Optional 1 channel weighing filter function is activated
M8002	Initialize pulse contact	M8114	Optional 1 channel weighing failure sign
M8003	Initialize pulse counter contact	M8115	Open thermocouple fault (no such function for now)
M8004	Error indicating contact	M8116	Optional 2 channel weighing function Channel 1 data overflow (no such function yet)
M8005	Random number generation relay	M8117	Optional 2-way weighing function Channel 2 data overflow (no such function yet)
M8006	Disable 6300-6399 fault flashing ERR light	M8235	Drive high-speed counting C235 to count down mode
M8008	Power failure detection (0N at power failure, OFF after power failure)	M8121~M8124	RS and MODBUS use
M8011	10 ms clock pulse	M8129	Serial 2 communication timeout flag
M8012	100 ms clock pulse	M8140	ZRN instruction clear output is valid
M8013	1 second clock pulse	M8145	Disable Y0 pulse output
M8014	1 minute clock pulse	M8146	Disable Y1 pulse output
M8015	Set the clock	M8147	Y0 pulse output
M8016	Clock display stopped	M8148	Y1 pulse output
M8017	Clock plus or minus 30 seconds correction	M8149	CAN communication timeout flag
M8018	Has real-time clock logo	M8150	CAN allowed work flag
M8019	Clock error sign	M8155	Disable Y2 pulse output
M8020	Zero mark	M8157	Y2 pulse output
M8021	Borrow mark	M8158	Y3 pulse output
M8022	Carry flag	M8161	16-bit/8-bit switching flag
M8029	Instruction execution end flag	M8168	SMOV instruction HEX processing function
M8031	Unlatched data clear	M8170	X0 pulse capture
M8032	Latch data clear	M8171	X1 pulse capture
M8034	Suppress all output	M8172	X2 pulse capture
M8039	Constant scan mode	M8173	X3 pulse capture
M8047	STL monitoring is effective	M8174	X4 pulse capture
M8048	S900-S999 has ON status	M8175	X5 pulse capture
M8049	Signal alarm is effective	M8196	C251 C252 C254 2 times frequency

			mark
M8050	I0 □□ interrupt disable	M8197	C253 C255 2 octave mark
M8051	I1 □□ interrupt disable	M8198	4 octave sign for C251 C252 C254
M8052	I2 □□ interrupt disable	M8199	C253 C255 4 octave logo
M8053	I3 □□ interrupt disable	M8200-M8234	C200-C234 count direction setting
M8054	I4 □□ interrupt disable	M8235-M8345	C235-C245 count direction setting
M8055	I5 □□ interrupt disable	M8246-M8255	C246-C255 count direction sign

3.3 Special register number and content

3.3.1 3G series PLC

Number	content	Remarks	Number	content	Remarks
D8000	Watchdog timer		D8148	Fifth to eighth pulse acceleration and deceleration time	
D8001	PLC type and system version		D8160	Eighth position pulse	Low position
D8002	PLC memory capacity	2...2K steps; 4...4K steps; 8...8K steps; At 16K steps or more, D8002=8, corresponding to 16, 32, 64 in D8102.	D8161		High position
D8003	Types of memory	10H: programmable controller built-in memory	D8169	Restricted access status	
D8010	Scan current value		D8182	Z1 Register contents	
D8011	Minimum scan time		D8183	V1 Register contents	
D8012	Maximum scan time		D8184	Z2 Register contents	
D8013	second		D8185	V2 Register contents	
D8014	Minute		D8186	Z3 Register contents	
D8015	Time		D8187	V3 Register contents	
D8016	day		D8188	Z4 Register contents	
D8017	month		D8189	V4 Register contents	
D8018	year		D8190	Z5 Register contents	

D8019	week		D8191	V5 Register contents	
D8020	Input filter adjustment		D8192	Z6 Register contents	
D8030	AD0 Analog input value		D8193	V6 Register contents	
D8031	AD1 Analog input value		D8194	Z7 Register contents	
D8032	AD2 Analog input value		D8195	V7 Register contents	
D8033	AD3 Analog input value		D8268	Custom PWM0~3 frequency division coefficient	Ranges: 840~ 16800000
D8034	AD4 Analog input value		D8269		
D8035	AD5 Analog input value		D8278	Custom PWM4~7 frequency division coefficient	
D8036	AD6 Analog input value		D8279		
D8037	AD7 Analog input value		D8340	First position pulse	Low position
D8038	AD8 Analog input value		D8341		High position
D8039	AD9 Analog input value		D8342	Y0 deviation speed Initial value: 0	
D8040	AD10 Analog input value		D8343	Maximum speed of the first pulse	Low position
D8041	AD11 Analog input value		D8344		High position
D8042	AD12 Analog input value		D8345	Y0 crawl speed Initial value: 1000	
D8043	AD13 Analog input value		D8346	Y0 origin return speed Initial value: 50000	Low position
D8044	AD14 Analog input value		D8347		High position
D8045	AD15 Analog input value		D8348	First pulse acceleration time	
D8050	DA0 Analog input value		D8349	The first pulse deceleration time	
D8051	DA1 Analog input value		D8350	Second position pulse	Low position
D8052	DA2 Analog input value		D8351		High position
D8053	DA3 Analog input value		D8352	Y1 deviation speed Initial value: 0	
D8054	DA4 Analog input value		D8353	Maximum speed of the second pulse	Low position
D8055	DA5 Analog input value		D8354		High position
D8056	DA6 Analog input value		D8355	Y1 crawl speed Initial value: 1000	
D8057	DA7 Analog input value		D8356	Y1 origin return speed Initial value: 50000	Low position

D8058	DA is the current time setting	Reference 5.2	D8357		High position
D8059	Constant scan time		D8358	Second pulse acceleration time	
D8074	X0 rising edge ring counter value [1/6μs unit]	Low position	D8359	Second pulse deceleration time	
D8075		High position	D8360	Third position pulse	Low position
D8076	X0 falling edge ring counter value [1/6μs unit]	Low position	D8361		High position
D8077		High position	D8362	Y2 deviation speed Initial value: 0	
D8078	X0 pulse width/pulse period [10μs unit]	Low position	D8363	Maximum speed of the third pulse	Low position
D8079		High position	D8364		High position
D8080	X1 rising edge ring counter value [1/6μs unit]	Low position	D8365	Y2 crawl speed Initial value: 1000	
D8081		High position	D8366	Y2 origin return speed Initial value: 50000	Low position
D8082	X1 falling edge ring counter value [1/6μs unit]	Low position	D8367		High position
D8083		High position	D8368	The third pulse acceleration time	
D8084	X1 pulse width/pulse period [10μs unit]	Low position	D8369	The third pulse deceleration time	
D8085		High position	D8370	Fourth position pulse	Low position
D8086	X3 rising edge ring counter value [1/6μs unit]	Low position	D8371		High position
D8087		High position	D8372	Y3 deviation speed Initial value: 0	
D8088	X3 falling edge ring counter value [1/6μs unit]	Low position	D8373	The fourth pulse maximum speed	Low position
D8089		High position	D8374		High position
D8090	X3 pulse width/pulse period [10μs unit]	Low position	D8375	Y3 crawl speed Initial value: 1000	
D8091		High position	D8376	Y3 origin return speed Initial value: 50000	Low position
D8092	X4 rising edge ring counter value [1/6μs unit]	Low position	D8377		High position
D8093		High position	D8378	The fourth pulse acceleration	

				time	
D8094	X4 falling edge ring counter value [1/6μs unit]	Low position	D8379	The fourth pulse deceleration time	
D8095		High position	D8395	Network setting function logo	
D8096	X4 pulse width/pulse period [10μs unit]	Low position	D8397	ADPRW command serial port location	
D8097		High position	D8398	0~2147483647(1ms)	
D8101	PLC type and system version		D8399	Incremental ring count	
D8102	PLC memory capacity	16...16K steps	D8400	Modbus RTU protocol Communication parameters	
D8108	Number of special modules connected		D8401	Communication mode	
D8109	Y number where output refresh error occurred		D8406	Number of intervals	
D8120	Communication parameters of Modbus RTU protocol		D8409	overtime time	
D8121	Master station number		D8410	RS2 header 1, 2 <initial value: STX>	
D8122	RS command sending data remaining points		D8411	RS2 header 3, 4	
D8123	RS command receiving point monitoring		D8412	RS2 trailer 1, 2 <initial value: ETX>	
D8124	RS command header <initial value: STX>		D8413	RS2 trailer 3, 4	
D8125	RS command trailer <initial value: ETX>		D8414	Master station number	
D8126	The value of serial port 2 when using ADPRW instruction is 0	The version before 26232	D8415	RS2 receives the sum calculation result	
D8126	Serial port 2 interval period	26232 and Later version	D8416	RS2 send summation	
D8127	Specify the start number of the lower computer communication request		D8420	Communication parameters	
D8128	Specify the number of data requested by the lower computer		D8421	Communication mode	
D8129	Set timeout		D8426	Number of intervals	
D8140	Fifth position pulse	Low position	D8429	overtime time	
D8141		High position	D8430	RS2 header 1, 2 <initial value: STX>	

D8142	Sixth position pulse	Low position	D8431	RS2 header 3, 4	
D8143		High position	D8432	RS2 trailer 1, 2 <initial value: ETX>	
D8144	Seventh position pulse	Low position	D8433	RS2 trailer 3, 4	
D8145		High position	D8434	RS2 receive sum receive data	
D8146	5th to 8th pulse maximum speed	Low position	D8435	RS2 receives the sum calculation result	
D8147		High position	D8436	RS2 send summation	

3.3.2 2N series PLC

Number	content	Number	content
M8000	closed during RUN	D8001	PLC type and version
M8001	open when RUN	D8002	Memory capacity
M8002	After RUN, output one scan cycle ON	D8003	Memory type
M8003	After RUN, output a scan cycle of OFF	D8011	Minimum scan time (unit 0.1ms)
M8011	Oscillate with a period of 10ms	D8012	Maximum scan time (unit 0.1ms)
M8012	Oscillate with a period of 100ms	D8013-D8019	Corresponding to seconds, minutes, hours, day, month, year, week
M8013	Oscillate with a period of 1s	D8020	Input filter adjustment (0-60ms) initial 10
M8014	Oscillate with a period of 1min	Class A analog	Refer to the table below
M8020	Zero mark	D8030-D8041	The value of analog input AD0-AD11
M8021	Borrow mark	D8042	Analog input cold junction ambient temperature value
M8022	Carry flag	D8213	E-type and K-type thermocouple switching
M8029	Instruction execution end flag	D8200-D8211	Corresponding to AD0-AD11 magnification correction
M8039	Constant scan mode	D8220-D8231	Corresponding to the size correction of AD0-AD11
M8035	The programmable controller continues to run	D8212, D8232	Corresponding to cold end magnification correction and size correction
M8037	The programmable controller stops running	D8039/ D39	Constant scan time (initial value 0ms); Note: If it is occupied by analog quantity, use D39
M8068	Saving of M8067	Class B analog	Refer to the table below
M8080	Analog output start	D8030-D8037	Value of analog input AD0-AD7
M8235	Drive high-speed counting C235 to count down mode	D8038	Analog input cold junction ambient temperature value

M8236	Drive high-speed counting C236 to count down mode	D8049	E-type and K-type thermocouple switching
M8238	Drive high-speed counting C238 to count down mode	D8040-D8047	Corresponding to AD0-AD7 magnification correction
M8239	Drive high-speed counting C239 to count down mode	D8070-D8077	Corresponding to the size correction of AD0-AD7
M8240	Drive high-speed counting C240 to count down mode	D8048, D8078	Corresponding to cold end magnification correction and size correction
Class C analog	Refer to the table below	D8039	Constant scan time (initial value 0ms)
D8030-D8049	Value of analog input AD0-AD19	EX2N-30A	Refer to the following table (others refer to category B)
D8049(Only as the thermocouple is the cold junction)	Analog input cold junction ambient temperature value	D8034	Analog input cold junction ambient temperature value
D8240	E-type and K-type thermocouple switching	D8045	E-type and K-type thermocouple switching
D8200-D8219	Corresponding to AD0-AD19 magnification correction	D8044, D8039	Corresponding to cold end magnification correction and size correction
D8220-D8239	Corresponding to the size correction of AD0-AD19	Some FX2NC purchased before 2016	Refer to the table below for some FX2NC purchased before 2016 (Others refer to Class B analog quantity)
D8212, D8232	Corresponding to cold end magnification correction and size correction	D8030-D8033	Value of analog input AD0-AD3
D8039/ D39	Constant scan time (initial value 0ms); Note: If it is occupied by analog quantity, use D39	D8034	Analog input cold junction ambient temperature value
D8050-69	Adjustment of the scanning period corresponding to the analog quantity	D8045	E-type and K-type thermocouple switching
D8065	Grammatical error occurrence step	D8040-D8043	Corresponding to AD0-AD3 magnification correction
D8068	The number of steps in the operation error record	D8035-D8038	Corresponding to the size correction of AD0-AD7
D8080-D8087	Analog output DA0-DA7 value	D8044, D8039	Corresponding to cold end magnification correction and size correction

		D8039/ D39	Constant scan time (initial value 0ms); Note: If it is occupied by analog quantity, use D39

3.3.3 MX2N series PLC

Number	content	Number	content
D8000	Monitoring timer setting value (default 200)	D8126	MODBUS master\slave communication delay time (1=1ms)
D8005	The lower 16 bits of the random number	D8127	MODBUS master station communication real time (1=10ms)
D8006	16 higher random numbers	D8128	MODBUS master communication maximum time (1=10ms)
D8007	End address of D register after power down	D8129	RS/MODBUS master communication timeout (1=10ms, default 500)
D8008	Power failure detection time (setting value: 1~100, default 10ms)	D8136	Y0 Y1 High-speed output count accumulation: 32 bits
D8010	Current value of scan time (0.1ms)	D8140	Y0 pulse output count register
D8011	Minimum scan time (0.1ms)	D8142	Y1 pulse output count register
D8012	Maximum scan time (0.1ms)	D8145	ZRN\DRVI\DRVA instruction Y0 Y1 minimum speed
D8013-D8019	Corresponding to seconds, minutes, hours, day, month, year, week	D8146	ZRN\DRVI\DRVA instruction Y0 Y1 maximum speed
D8020	X0-X17 filter coefficient (setting value: 0~60ms, default 10)	D8148	ZRN\DRVI\DRVA instruction Y0 Y1 acceleration and deceleration time
D8021	X20-X47 filter coefficient (setting: 1~60ms, default 10)	D8149	CAN master/slave communication timeout (1=1ms)
D8028	Z0 index register content	D8150	Master/slave station number (0~32)
D8029	V0 index register content	D8151	Number of slaves (1~32, default: 8)
D8030-D8038	Sampling address of analog input AD0-AD8	D8152	Number of shared registers (1~32, default: 8)
D8050-D8052	Sampling address of analog input AD9-AD11	D8153	CAN communication baud rate (20K~100K, default: 250K)
D8039	Constant scan time (unit: 1ms, default 0)	D8154	Y2 pulse output count register
D8040-D8047	1- 8th active STL status	D8156	Y3 pulse output count register
D8049	Minimum active STL status	D8159	ZRN\DRVI\DRVA instruction Y2 Y3 minimum speed

D8058	Optional 2-channel weighing function Channel 1 data divisor (no such function yet)	D8160	ZRN\DRVI\DRVA instruction Y2 Y3 maximum speed
D8059	Optional 2-channel weighing function Channel 2 data divisor (no such function yet)	D8162	ZRN\DRVI\DRVA instruction Y2 Y3 acceleration and deceleration time
D8090	Thermocouple sampling filter times (0-22, default 0) (no such function for now)	D8166	Y2 Y3 High-speed output count accumulation: 32 bits
D8091	Thermocouple type (K-0, E-1, J-2) (No such function yet)	D8182	Z1 index register content
D8093	Thermocouple cold junction temperature (no such function for now)	D8183	V1 index register content
D8094	Temperature of the first thermocouple (no such function yet)	D8184	Z2 Index register content
D8095	Temperature of the second thermocouple (no such function yet)	D8185	V2 index register content
D8096	Analog DA0 output data (0~4095)	D8186	Z3 Index register content
D8097	Analog DA1 output data (0~4095)	D8187	V3 Index register content
D8112	Optional 1 channel weighing data low bit	D8188	Z4 Index register content
D8113	Optional 1 channel weighing data high	D8189	V4 Index register content
D8114	Optional 1 channel weighing filter times	D8190	Z5 Index register content
D8115	Optional 2-way weighing function filtering times (0-80) (no such function for now)	D8191	V5 Index register content
D8116	Optional 2 channel weighing function channel 1 data high (No such function yet)	D8192	Z6 Index register content
D8117	Optional 2 channel weighing function channel 1 data low bit (No such function yet)	D8193	V6 Index register content
D8118	Optional 2-channel weighing function communication 2 data high (No such function yet)	D8194	Z7 Index register content
D8119	Optional 2 channel weighing function channel 2 data low bit (No such function yet)	D8195	V7 Index register content D8196
D8120	Serial 2 communication parameter settings	D8196	Slave 1~16 with CAN communication failure
D8121	Serial port 2 MODBUS RTU slave station (1~255)	D8197	Slave 17~32 with CAN communication failure
D8122	RS instruction sends the remaining data	D8198	Slave summary of CAN communication failure 1~16
D8123	Number of RS commands received	D8199	Summary of slaves with CAN communication failure 17~32
		D8200	CAN communication success time

			(1-1ms)

3.4 Input and output relays [X, Y]

The numbers of the input relay and output relay are fixed numbers held by the basic unit, and are counted from X0 and Y0. Since these numbers use octal numbers, there is no value of "8" or "9".

3.4.1 I/O relay number

The numbers of input relay (X) and output relay (Y) are shown in the table below. (Numbers are assigned in octal numbers)

I/O relay			
Input relay	X000~X047	40 points	The device number is an octal number Total input and output is 80 points
Output relay	Y000~Y047	40 points	

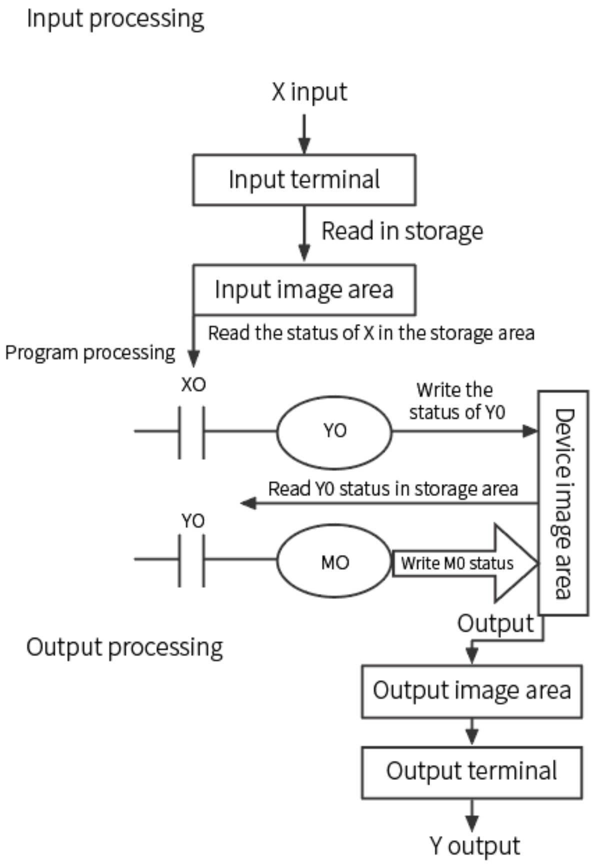
3.4.2 Function and effect

3.4.2.1 The function of input contact X:

The input contact X is connected to the input device, and the input signal is read to enter the PLC. There is no limit to the number of times that the normally open contact or normally closed contact of each input contact X can be used in the program. The ON/OFF of the input contact X will only change with the ON/OFF of the input device. You cannot use a program to force the ON/OFF of the input contact X.

3.4.2.2 Function of output contact Y:

The task of output contact Y is to send an ON/OFF signal to drive the load connected to output contact Y. There are two types of output contacts, one is Relay and the other is Transistor. There is no limit to the number of times the normally open contact or normally closed contact of each output contact Y can be used in the program, but the output coil Y The number is recommended to be used only once in the program, otherwise according to the principle of PLC program scanning, the decision of its output state will fall in the last output Y circuit in the program.



- Input processing:
  1. The PLC reads the ON/OFF status of the external input signal into the input image area once before executing the program.
  2. If the input signal changes ON/OFF during program execution, the state in the input image area will not change until the next ON/OFF state of the input signal is read again at the beginning of the next scan.
  3. There is a delay of about 10ms when the external signal ON→OFF or OFF→ON changes to when the contact in the program is recognized as ON/OFF (but may be affected by the program scan cycle).
- Program processing:
  The PLC reads the ON/OFF status of each input signal in the input image area and starts to execute each instruction in the program in sequence from address 0. The processing result, that is, the ON/OFF of each output coil, is also sequentially stored in each device image area Inside.
- Output processing:
  1. When the END instruction is executed, the ON/OFF status of Y in the device image area is sent to the output image area latch, and this image area is actually the coil of the output relay.
  2. There is a delay of about 10ms when the relay coil ON→OFF or OFF→ON changes to the contact ON/OFF.
  3. Using a transistor module, there is a delay of about 10~20us when ON→OFF or OFF→ON changes to the contact ON/OFF.

3.5 Auxiliary relay [M]

There are multiple auxiliary relays in the PLC. The coils of these auxiliary relays are the same as the output relays, and are driven by the contacts of various soft elements in the PLC. Auxiliary relays have countless electronic normally open contacts and normally closed contacts, which can be freely used in PLC. However, the external load cannot be directly driven through this contact, and the external load must be driven through the output relay Y.

3.5.1 Auxiliary relay number

1. 3G series PLC

Auxiliary relay		
General use	M0~M383, M1536~M7679	6528 points, fixed as a non-blackout holding area
EEPROM retention	M384~M1535	1152 points, fixed as the blackout holding area
Special use	M8000~M8511	512 points

2. 2N series PLC

Auxiliary relay	general	500 points M0 to M499
	lock	1036 points M500-M1535
	special	256 points M8000 to M8255

3. MX2N series PLC

Auxiliary relay M	M0~M499 500 points (general use)	M500~M1535 1036 points (for holding)	M8000~M8255 255 points (special use)

3.5.2 Function of auxiliary relay

1. Auxiliary relays for general use: Auxiliary relays for general use are turned off when the PLC is in operation, and all of their statuses are reset to OFF. When power is supplied, except for the input condition of ON, the status is all OFF.
2. Auxiliary relay for power failure maintenance: The auxiliary relay for power failure maintenance will be kept in its state when the power is turned off during PLC operation, and the state will be the state before power failure when the power is re-transmitted.
3. Special auxiliary relays: Each special auxiliary relay has its specific function. Do not use undefined special auxiliary relays.

3.6 Status [S]

State S is an important soft element for easy programming in engineering automation control. It needs to be used in combination with step ladder diagram (or Sequential Function Chart, Sequential FunctiON Chart, SFC) instructions STL and RET.

3.6.1 State number

3.6.1.1 3G series PLC

status		
For initial state (EEPROM hold)	S0 ~ S9	10 o'clock, fixed as the blackout holding area
Dedicated for power failure retention (for EEPROM retention)	S10 ~ S899	890 points
For signal alarm	S900 ~ S999	100 points
General use	S1000 ~ S4095	3096 points

3.6.1.2 2N series PLC

status	general	500 points S0 to S499
	initial	10 points S000-S009
	lock	500 points S500 to S999

3.6.1.3 MX2N series PLC

Status relay S	S0~S9 10 points (for state maintenance)	S10~S999 990 points (for holding)

3.6.2 State function

The state is the same as the auxiliary relay. There are countless normally open contacts and normally closed contacts, which can be used at will in the sequence program. Moreover, when not used for step ladder instructions, the state (S) is also the same as the auxiliary relay (M) and can be used in general sequence control.

1. Initial state: S0~S9, the step point used as the initial state in the sequential function chart (SFC).

2. For power retentive: When the power is disconnected during the PLC operation, its state will be maintained, and when the power is re-transmitted, the state will be the state before the power failure;
And when it is run again, it can be restarted from the middle of the process.
When the power-off holding state is used as a general-purpose state, you can add ZRST in front of the program to reset its state in batches.

3. For signal alarm: S900～S999, signal alarm status, can be used as output for diagnosing external faults. (For details, see 10.7 ANS/Signal Alarm Set)

4. General use: When the PLC power supply is disconnected, the general status will be cleared.

3.7 Timer [T]

The timer is a device that calculates the clock pulses of 1ms, 10ms, 100ms, etc. in the PLC by addition, and when the result of the addition calculation reaches the specified setting value, the output contact operates. As the setting value, the constant (K) in the program memory and the indirect designation by the contents of the data register (D) can be used.

3.7.1 Timer number

3.7.1.1 3G series PLC

Timer (ON delay timer, the number is assigned in decimal numbers)			
100ms	T0～T199	200 point	0.1～3,276.7 second

10ms※1	T200 ~ T245	46 point	0.01 ~ 327.67 second
1ms cumulative type (EEPROM keep)	T246 ~ T249	4 point	0.001 ~ 32.767 second
100ms cumulative type (EEPROM keep)	T250 ~ T255	6 point	0.1 ~ 3,276.7 second
1ms	T256 ~ T319	64 point	0.001~32.767 second

3.7.1.2 2N series PLC

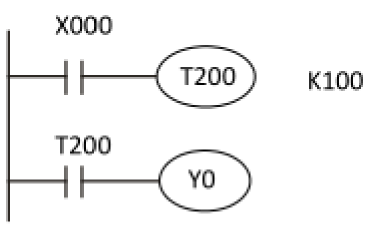
Timer	100 millisecond	200 points T0 to T199
	10 millisecond	46 points T200 to T245
	1 Millisecond integration	4 points T246 to T249
	100 Millisecond integration	6 points T250 to T255

3.7.1.3 MX2N series PLC

Timer T	T0~T199 200 points 100ms	T200~T245 46 points 10ms	T246~T249 4 points 1ms accumulation	T250~T255 6 points 100ms accumulation

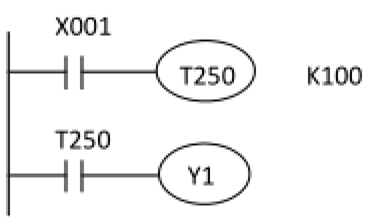
3.7.2 Timer functions and examples

3.7.2.1 General timer



- When X0=ON, the current value of timer T200 is counted by 10ms*k100, when the current value of timer=set value K100, the output coil T200=ON.
- When X0=OFF or power failure, the timer T200 will be cleared. And the output coil T200 becomes OFF.

3.7.2.2 Cumulative timer



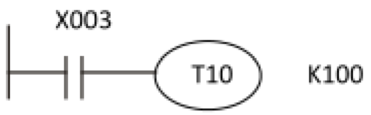
- When X1=ON, the current value of timer T250 is counted as 100ms*k100, when the current value of timer=set value K100, output coil T250=O
- When X1=OFF or power failure, the timer T250 pauses counting, and the current value remains unchanged; when the PLC runs again, that is, when X1=ON, the T250 continues to time, the accumulated time is 10s; the current value=set value K100, the output coil T250=ON

3.7.3 Setting method of setting value

1. Specify constant (K): The setting value directly specifies the constant K value.

- T10 is a timer in units of 100ms (0.1s).
- If the constant is specified as 100, then the timer works at 0.1s*100=10s

2. Indirect addressing (D): The set value uses data register D for indirect addressing.



- The contents of the indirectly designated data register are either written in the program in advance, or input through digital switches.

3.7.4 Notes in the subroutine

1. In the subroutine and interrupt subroutine, please use the timer of T192~T199. This timer counts when the coil instruction is executed or when the END instruction is executed.

If the set value is reached, the contact action will be output when the coil instruction or END instruction is executed. Since the general timer only counts when the coil instruction is executed, only under certain conditions, the subroutine and interrupt subroutine of the coil instruction are executed. If the timer is used to count, it cannot be executed and cannot operate normally.

2. In the subroutine and interrupt subroutine, if a 1ms accumulation timer is used, when it reaches the set value, the output contact will act at the coil command that was initially executed. Please pay attention.

3.8 Counter [C]

The counter counts while performing a cyclic operation on the actions of the internal signals X, Y, M, S, C and other contacts of the programmable controller.

3.8.1 Counter number

3.8.1.1 3G series PLC

Counter (numbers are assigned in decimal numbers)			
General use count-up (16 bits)	C0 ~ C15	16points	0 ~ 32,767 counter
EEPROM keeps counting up (16 bits)	C16 ~ C199	184points	0 ~ 32,767 counter
Generally use bidirectional (32 bit)	C200 ~ C219	20points	-2,147,483,648 ~ +2,147,483,647 Counter

EEPROM hold in both directions (32 bit)	C220 ~ C234	15points	-2,147,483,648 ~ +2,147,483,647 Counter

3.8.1.2 2N series PLC

counter	16 bits	100points C0-C99
	Lock 16 bits	100points C100-C199
	32-bit	
	Lock 32 bits	35points C200-C234

3.8.1.3 MX2N series PLC

Counter C	16-bit up counter		32-bit up-down counter	
	C0~C15 16 o'clock (General use)	C0~C15 16 o'clock (General use)	C200~C219 20 points (General use)	C220~C234 15 points (for holding)

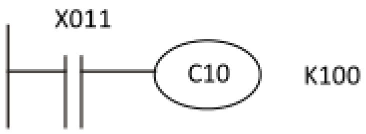
3.8.2 Counter functions and examples

Generally, a counter is used. If the PLC power supply is disconnected, the count value will be cleared. However, the power failure retention counter will remember the count value and the state of the contact before the power failure, so it can continue to count on the previous value. .

The setting value of the counter can be set directly using the constant K or indirectly using the value in the register D.

3.8.2.1 16-bit counter for general use / power failure retention

The setting value of the 16-bit binary up counter is valid in the range of K1 to K32767 (decimal constant). The operation of K0 is the same as that of K1, and the output contact operates at the first count.



- By counting input X011, the current value of the counter will increase every time the C0 coil is driven. When the coil command is executed for the 10th time, the output contact C0 will act; after that, even if the counting input X011 acts, the current value of the counter will not Variety.

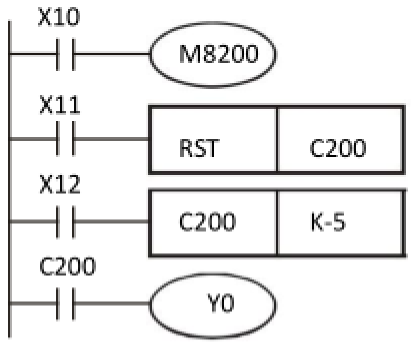
3.8.2.2 32-bit up/down counter for general use/maintained after power failure

The set value of the 32-bit binary up/down counter is valid in the range of -2,147,483,648 to +2,147,483,647 (decimal constant). You can use the auxiliary relays M8200 to M8234 to specify the direction of up/down counting.

The increase and decrease of the current value has nothing to do with the operation of the output contact. The current value of the counter starts from 2,147,483,647 to increase the number, and then becomes -2,147,483,648. Similarly, the current value of the counter starts to decrease from -2,147,483,648 and becomes 2,147,483,647. (This action is called ring counting)

For C△△△, after driving M8△△△, it is a down counter, when it is not driving, it is an up counter. (Refer to 3.3 Special Relay Number and Content)

```
LD    X10
OUT   M8200
LD    X11
RST   C200
LD    X12
OUT   C200    K-5
LD    C200
OUT   Y0
```



- X10 drives M8200 to determine whether C200 is up or down.
- When X11 turns from OFF→ON, the RST instruction is executed, the current value of C200 is cleared to 0, and the contact becomes OFF.
- When X12 turns from OFF→ON, the current value of the counter will perform the count-up (plus one) action or the count-down (decrement one) action.
- When the current value of the counter C200 changes from K-6→K-5, the C200 contact changes from OFF→ON. When the current value of the counter C200 changes from K-5→K-6, the C200 contact is turned from ON→OFF.

3.9 High-speed counter [C]

3.9.1 Types and numbers of high-speed counters

3.9.1.1 Types of high-speed counters

In the PLC, a high-speed counter (single-phase single-count, single-phase double-count and double-phase double-count) with a 32-bit up-down counter built-in is divided into a hardware counter and a software counter according to different counting methods.

3.9.1.2 High-speed counter number

1) 3G series PLC

High-speed counter (numbers are assigned in decimal numbers)		
Single-phase single-count input Bidirectional (32 bit) (EEPROM hold)	C235 ~ C245	-2,147,483,648 ~ +2,147,483,647 counter Software counter 1 phase: max 6 channels, max 60 KHz each Two-phase: 1-time frequency: up to 2-3 channels, maximum frequency 60KHz M8198 is the 4 times frequency mark of
Single-phase double-count input Bidirectional (32 bit) (EEPROM hold)	C246 ~ C250	

Two-phase double-count input Bidirectional (32 bit) (EEPROM hold)	C251～C255	C251/C252 4-times frequency: max 2 channels, max frequency 24KHz M8199 is the 4 times frequency mark of C253/C255

2) 2N series PLC

High-speed counter	Single phase	Max 6 points:C235-X0 C236-X1 C237-X7 C238-X3 C239-X4 C240-X5; Regular 2 points:C235-X0 C238-X3
	A/B phase	Max 3 points:C251-X0/X1 C253-X3/X4 C254-X10/X11 ; Regular 2 points: C251-X0/X1 C253-X3/X4

3) MX2N series PLC

High-speed counter C	32-bit up-down counter
	C235~C255 20 points (High-speed hold)

3.9.1.3 High-speed counter input allocation table

1) 3G series PLC

Counter type	Counter number	Input allocation							
		X000	X001	X002	X003	X004	X005	X006	X007
Single-phase single-count input	C235	U/D							
	C236		U/D						
	C237			U/D					
	C238				U/D				
	C239					U/D			
	C240						U/D		
	C241	U/D	R						
	C242			U/D	R				
	C243					U/D	R		
	C244	U/D	R					S	
	C245			U/D	R				S
Single-phase double counting input	C246	U	D						
	C247	U	D	R					
	C248				U	D	R		
	C249	U	D	R				S	
	C250				U	D	R		S
Two-phase double counting input	C251	A	B						
	C252	A	B	R					
	C253				A	B	R		
	C254							A	B
	C255				A	B	R		S

U: Up counting input | D: Down counting input | A: Phase A input | B: Phase B input | R: External reset input | S: External start input

Single phase: max 6 channels, max 60 KHz each

Two-phase: 1-time frequency: up to 2-3 channels, maximum frequency 60KHz. M8198 is the 4 times frequency mark of C251/C252.

4-times frequency: max 2 channels, max frequency 24KHz. M8199 is the 4 times frequency mark of C253/C255.

2) 2N series PLC

	Single-phase counting input						AB phase counting input			ABZ phase counting input		
	C235 10KHz/100KHz	C236 100KHz	C238 10KHz/100KHz	C239 100KHz	C240 10KHz	C237 10KHz	C251 10KHz/100KHz	C253 10KHz/100KHz	C254 10KHz	C252 10KHz/100KHz	C253 10KHz/100KHz	C254 10KHz
X000	U/D						A			A		
X001		U/D					B			B		
X002										Z		
X003			U/D					A			A	
X004				U/D				B			B	
X005					U/D			R			Z	
X007						U/D						
X010									A			A
X011									B			B
X012												Z

Conventional [U]: Up-counting input | [D]: Down-counting input | [A]: A-phase count input | [B]: B-phase count input | [R]: Reset input

- The maximum frequency of single-phase counting is 10KHz, which can be customized up to 6 channels of single-phase 10KHz-100KHz and 3 channels of AB(Z) phase 10KHz-100KHz.
- Single-phase counting 10KHz is usually X00/X03, corresponding to C235/238. Can be customized up to 6 single-phase counting, the counter corresponds to the X point relationship: C235-X0; C236-X1; C237-X7; C238-X3; C239-X4; C240-X5; C237 should be connected to X2 for high-speed counting, now Instead, connect to X7 for high-speed counting; among them, X0/X1/X3/X4 can be customized to 100KHz, X5/X7 can be customized to 10KHz.

- When using 6-way single-phase counting, it does not conflict with other counters and pulse outputs, but conflicts with the ZRN origin return instruction. The ZRN origin return instruction cannot be used; only when X3 is not used for counting, the Y7/X7 ZRN origin return instruction can be used.
- The AB phase count is 2 times the frequency, the conventional is 10KHz two-way X00-X01/X03-X04, corresponding to C251/C253. It can also be customized to 3-channel AB phase counting, add one channel X10-X11, corresponding to C254; X00-X01/X03-X04 can be customized to 100KHz, X10-X11 can be customized to 10KHz.

3) MX2N series PLC

Counter type	Counter number	Input allocation							
		X000	X001	X002	X003	X004	X005	X006	X007
Single-phase single-count input	C235	U/D							
	C236		U/D						
	C237			U/D					
	C238				U/D				
	C239					U/D			
	C240						U/D		
	C241	U/D	R						
	C242			U/D	R				
	C243					U/D	R		
	C244	U/D	R					S	
	C245			U/D	R				S
Single-phase double counting input	C246	U	D						
	C247	U	D	R					
	C248				U	D	R		
	C248(OP)*1				U	D			
	C249	U	D	R				S	
	C250				U	D	R		S
Two-phase double counting input	C251	A	B						
	C252	A	B	R					
	C253				A	B	R		
	C253(OP)*1				A	B			
	C254	A	B	R				S	
	C254(OP)*1							A	B
	C255				A	B	R		S

U: Up counting input | D: Down counting input | A: Phase A input | B: Phase B input | R: External reset input | S: External start input

- Enter X000~X007, classified as shown in the table above, corresponding to each high-speed counter number.

Inputs X000~X007 cannot be used repeatedly by high-speed counters. When the input terminal is not used as a high-speed counter, it can be used for general input.

- Input X000~X007 can not be used repeatedly. For example, once C251 is used, X000 and X001 are occupied, so C235, C236, C241, C244, C246, C247, C249, C252, C254 and interrupt input pointers *I00, *I01 and the corresponding input SPD instructions cannot be used.

1) C251 C252 C254 (AB phase) maximum response frequency: 60KHz;
2) C253 C255 (AB phase) maximum response frequency: 60KHz;
3) C235 C241 C244 C238 (single-phase) maximum response frequency: 60KHz;
4) The highest response frequency of other high-speed counters: 10KHz;
5) The AB phase high-speed counter can be set to 2 times the frequency and 4 times the frequency (the setting is only valid during the OUT drive cycle):

**When M8196-ON, C251 C252 C254 count pulse 2 times frequency;

**When M8197-ON, C253 C255 count pulse 2 times frequency;

**When M8198-ON, C251 C252 C254 count pulse 4 times frequency;

**When M8199-ON, C253 C255 count pulse 4 times frequency.

3.9.1.4 Attentions

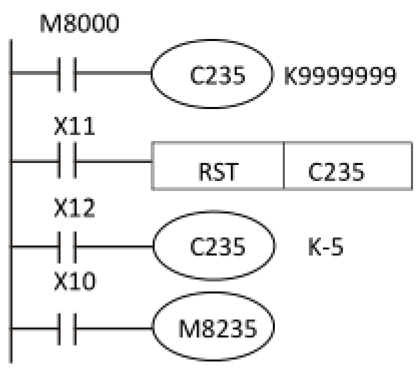
Input X000～X007 can be used for high-speed counter, input interruption, pulse capture, SPD, ZRN, DSZR DVIT instruction and general input. Therefore, do not reuse input terminals.

For example: X000 and X001 are occupied when using C251, so [C235, C236, C241, C244, C246, C247, C249, C252, C254], [input interrupt pointer I000, I101], [pulse capture contact M8170 , M8171] and [Use the corresponding input SPD, ZRN, DSZR, DVIT instructions] can not be used.

3.9.2 Use of high-speed counters

3.9.2.1 Single-phase single-count input

```
LD    M8000
OUT   C235
LD    X11
RST   C235
LD    X12
OUT   C235   K-5
LD    X10
OUT   M8235
```



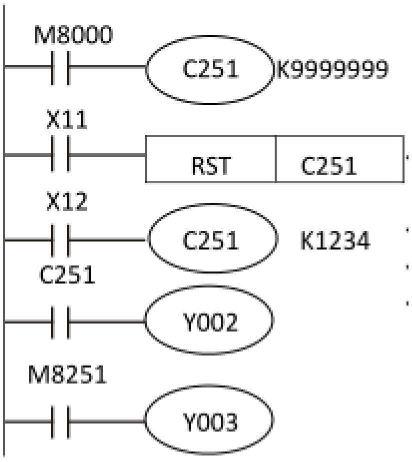
- ◆ C235 counts the OFF→ON of input X000 when X012 is ON.
- ◆ When X011 is ON, execute the RST instruction, at this time C235 will be reset.
- ◆ Through the ON/OFF of M8235～M8245, make the counters C235～C245 change between down/up counting

3.9.2.2 Single-phase double-count input

It is a 32-bit up/down binary counter. The action of the output contact corresponding to the current value is the same as the high-speed counter of single-phase single-count input.

The count-down/count-up actions of C246～C250 can be monitored by the ON/OFF actions of M8246～M8250. ON: count down; OFF: count up

3.9.2.3 Two-phase double-count input



- When X012 is ON, C251 counts the operation of input X000 (phase A) and X001 (phase B) by interrupt.
- X011 is ON, execute RST instruction, at this time C251 will be reset.
- If the current value exceeds the set value, Y002 will be ON, and it will be OFF when it changes within the set value.
- Y003 turns ON (minus) and OFF (increasing) according to the counting direction.

3.10 Data register [D], extension register [R]

3.10.1 Number of data register and extension register

The data register is used to store numeric data. Its data length is 16 bits (-32,768～+32,767), and the highest bit is a sign. It can store numeric data of -32,768～+32,767, or it can combine two 16-bit registers into A 32-bit register (D+1, the lower D number is the lower bit) is used, and the highest bit is a positive and negative sign, which can store the numerical data of -2,147,483,648～+2,147,483,647.

3.10.1.1 3G series PLC

Data register (32 bits when used in pairs)		
General use (16 bit)	D0 ~ D127	128points
EEPROM retention (16 bits)	D128 ~ D7999	7872points
Special use (16 bits)	D8000 ~ D8511	512points

For indexing (16 bits)	V0～V7,Z0～Z7	16 points
Extension Register•Extended File Register		
Extension register (16 bits)	R0～R22999	23000 points, support blackout
	R23000 ～ R23999	1000 points for internal use

3.10.1.2 2N series PLC

Data register (D.V.Z)	general	200 points D0 to D199
	Blackout	800 points D200-D999
	File register	
	External regulation	
	special	256 points D8000 to D8255
	Index	16 points V0-V7 Z0-Z7

3.10.1.3 MX2N series PLC

Register D.V.Z	D0~D199 200 points (general use)	D200~D7999 780 points (for keeping)	D8000~D8195 196 points (special use, hold)	D8196~D8255 59 points (for special use)	V0~V7 Z0~Z7 16 points (for indexing)

3.10.2 Function of data register and extension register

The register can be divided into the following five types according to its nature:

1. General register: When PLC is from RUN→STOP or power off, all data in the register will be cleared to 0. If M8033=ON, then the data will remain uncleared when PLC is from RUN→STOP, but it will still be Clear to 0.
2. Power failure retention register: The PLC keeps its contents during RUN→STOP and power failure; to clear the contents of the power failure retention register, you can use the RST or ZRST instruction.
3. Special register: Each special purpose register has its special definition and purpose, mainly used for storing system status, error information, and monitoring status. Please refer to section 3.2 Special register numbers and contents.
4. Index register: The index register is a 16-bit register, which is used in the same way as the data register. You can also use other device numbers and values in combination with the operand of the application instruction to change the special register of the device number and value in the program.
5. Extension register: Same as the data register, it is 16-bit data (the highest bit is a positive and negative

sign), but after combining two devices, you can save 32-bit (the highest bit is a positive and negative sign) numeric data.

3.10.3 Index register [V], [Z]

3.10.3.1 16 bits

The index register has the same structure as the data register

← 16bits →	← 16bits →
V	Z
V0~V7: 8 points	Z0~Z7: 8 points

3.10.3.2 32 bits

Z0~Z7 must be used when modifying the device in the 32-bit application instruction or processing the value beyond the 16-bit range.

← 32 bit →	
Vx(High)	Zx(Low)

As shown in the above figure, the combination of V and Z must specify Z when using 32-bit length. At this time, V is occupied and cannot be used in other places, otherwise it will cause Z (32-bit data) to be inaccurate.

- ◆ When X0=O, Z0=8, V0=14, D5Z0=D(5+8)=D13, D10V0=D(10+14)=D24, at this time the content of D13 will be moved to D24

3.10.3.3 Modification of soft components

◆ Decimal number device•Numerical value: M, S, T, C, D, R, KnM, KnS, P, K

For example, V0=K5, when executing D20V0, execute the instruction with the device number D25 (D20+5). In addition, you can modify the constant. When K30V0 is specified, the command executed is the decimal value K35 (30+5).

◆ Octal number devices: X, Y, KnX, KnY

For example, Z1=K8, when executing X0Z1, execute the instruction with the device number X10 (X0+8: octal number addition). When index modification is performed on a device whose device number is an octal number, the contents of V and Z will also be converted into an octal number and then added. Therefore, assume that Z1=K10 and X0Z1 is designated as X12. Please note that it is not X10 at this time.

- Hexadecimal value: H

For example, V5=K30, when the constant H30V5 is specified, it is regarded as H4E (30H+K30). In addition, when V5=H30 and the constant H30V5 is specified, it is regarded as H60 (30H+30H).

3.11 Pointer [P], [I]

3.11.1 Pointer number

3.11.1.1 3G series PLC

pointer			
For JUMP, CALL branch	P0～P255 P0～P1280	256 points 1281 points (26232 and above version)	For CJ instruction, CALL instruction
Input interrupt	I0□□～I5□□	6 points, X0~X5	
Timer interrupt	I6□□～I8□□	3 points	
Counter interrupt	I0□0～I0□0	6 points [□=1~6]	

3.11.1.2 2N series PLC

pointer	JUMP, CALL	128 points P0-P127
	Input interrupt	

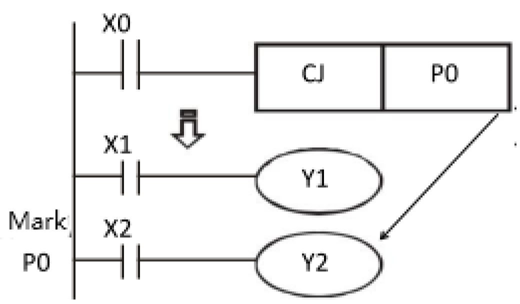
3.11.1.3 MX2N series PLC

Nested pointer	P0~P127 128 points (for jump and subroutine)	I0 □□~I5 □□ 6 points (for external interrupt)

Note: When using the pointer for input interrupt, the input number assigned to the pointer cannot be used together with [High Speed Counter] and [Pulse Density (FNC 56)] that use the same input range.

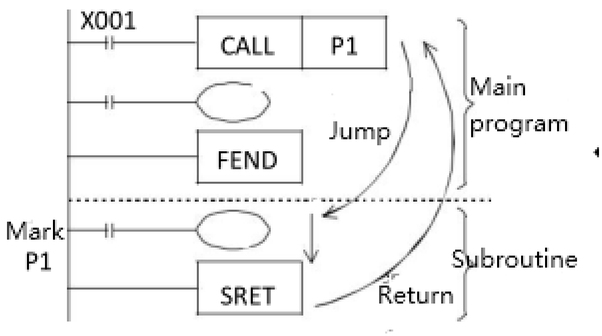
3.11.2 Pointer function

3.11.2.1 CJ conditional jump



- X0=ON, jump to the mark position specified by CJ instruction, and execute the following program;
- X0=OFF, the program is executed from top to bottom as a normal program, and the CJ instruction is not executed at this time.

3.11.2.2 CALL subroutine



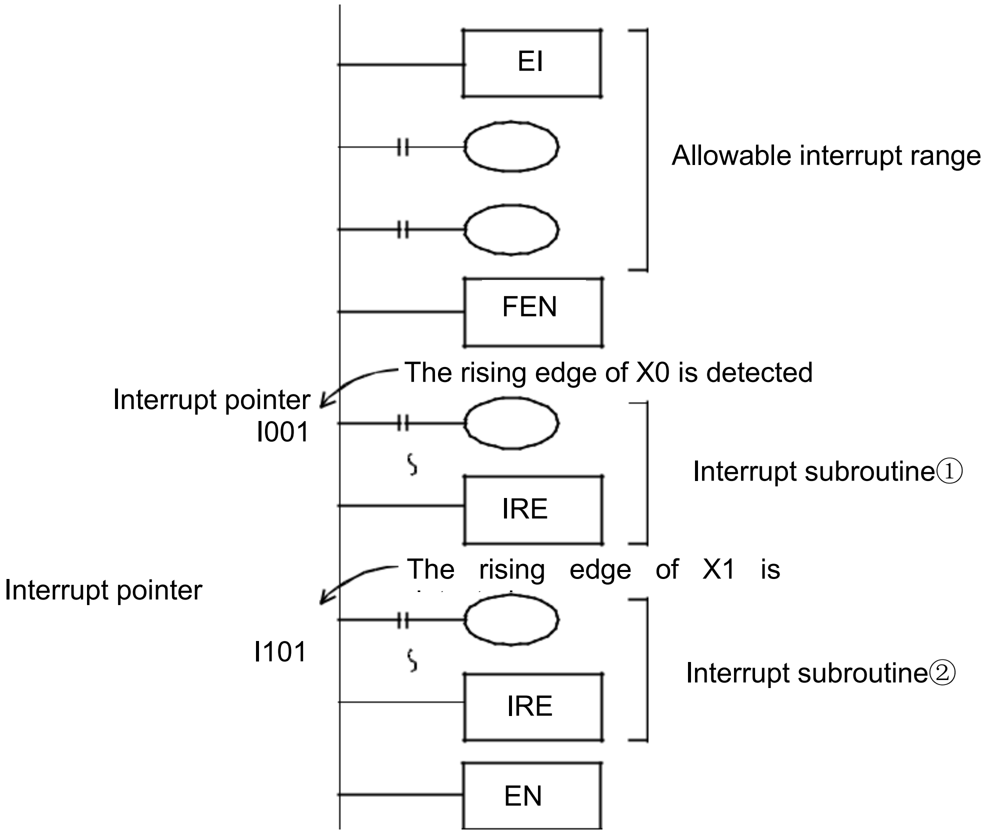
- X001=ON, execute the subroutine at the label position specified by the CALL instruction, and use the (SRET) instruction to return to the original position.

3.11.3 Function of interrupt pointer

The interrupt pointers include the following three types, which are used together with the application instruction IRET (FNC 03) interrupt return, EI (FNC 04) enable interrupt, and DI (FNC 05) disable interrupt.

3.11.3.1 Input interrupt (delayed interrupt): 6 points

Input	Input interrupt pointer		Disable interrupt flag
	Interrupt on rising edge	Falling edge interrupt	
X000	I001	I000	M8050
X001	I101	I100	M8051
X002	I201	I200	M8052
X003	I301	I300	M8053
X004	I401	I400	M8054
X005	I501	I500	M8055



- The EI instruction in the PLC allows interruption. During the scan program, X000 or X001=ON, execute the interrupt subroutine ① or ②, and then return to the main program through the IRET instruction.
- The interrupt pointer (I***) must be placed after the FEND instruction as a mark during programming.

Notes:

1) The number of the input relay used as an interrupt pointer should not be repeated with application instructions such as "high-speed counter", "pulse capture function", "pulse density" and other applications that use the same input range.

2) After the input interrupt pointer I port 0 port is specified, the input filter of the input relay will be automatically changed to high-speed reading.

Therefore, there is no need to use the REFF (FNC 51) command and the special data register D8020 (input filter adjustment) to change the filter adjustment.

In addition, the input filter without an input relay used as an input interrupt pointer operates at 10ms (initial value).

3) The rising edge interrupt and falling edge interrupt of the same input like I001 and I000 cannot be programmed at the same time.

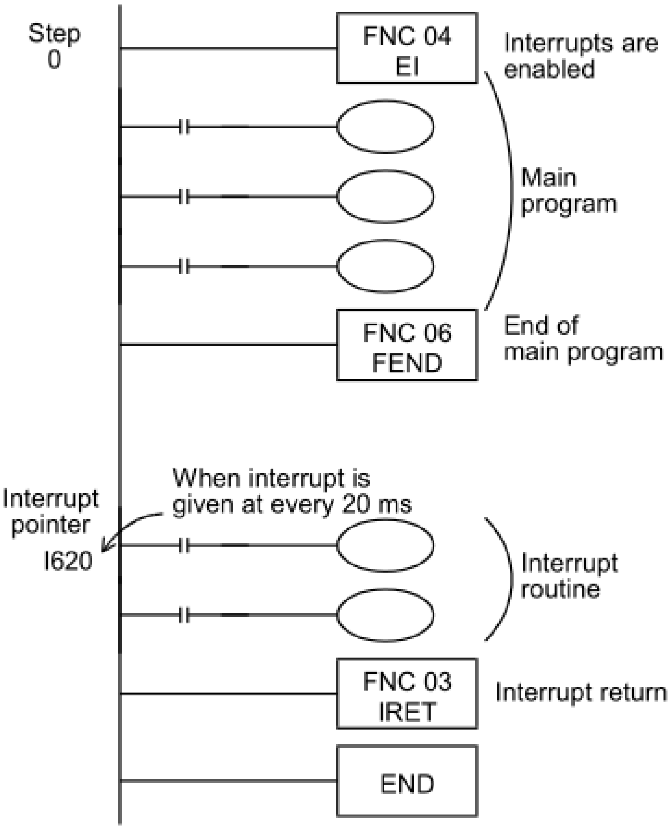
3.11.3.2 For timer interrupt: 3 points

Every designated interrupt cycle time (10ms～99ms), execute interrupt subroutine. Outside the operation cycle of the programmable controller, the timer interrupt becomes valid after the EI instruction.

Input number	Interrupt cycle	Disable interrupt flag
I6 □□	In the mouth of the pointer name, enter an integer from 10 to 99. For example: I610=Timer interrupt every	M8056
I7 □□		M8057

I8 □□	10ms	M8058

Note: The pointer numbers (I6, I7, I8) cannot be reused.



- The timer interrupt becomes valid after the EI instruction. In addition, when the prohibition interval of the timer interrupt is not required, there is no need to program DI (interrupt prohibition instruction).
- FEND indicates the end of the main program. The interrupt subroutine must be written after FEND.
- Execute interrupt subroutine every 20ms.
Use IRET instruction to return to the main program

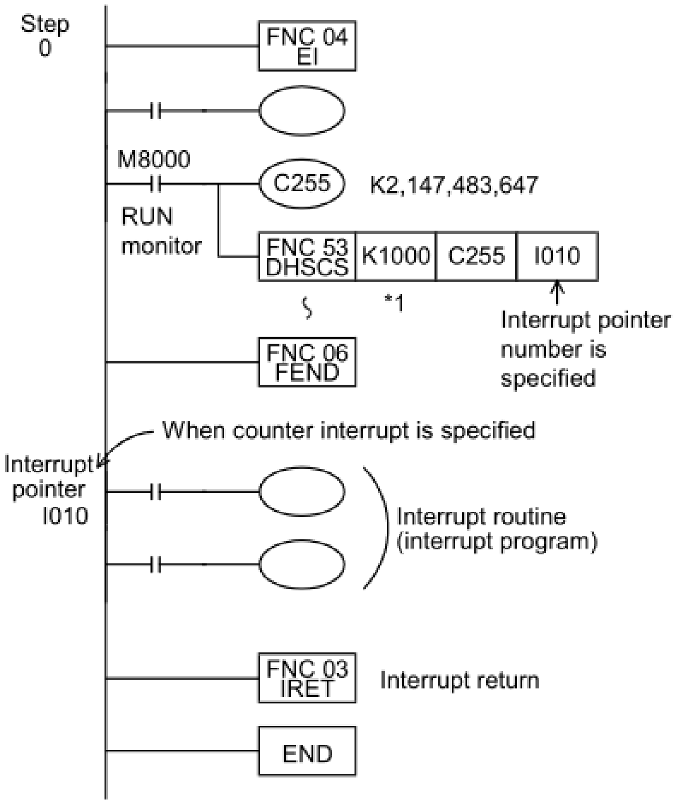
3.11.3.3 For counter interruption: 6 points

According to the comparison result of the high-speed counter comparison set instruction (DHSCS instruction), the interrupt subroutine is executed. It is used to control the priority processing of counting results using high-speed counters.

Pointer number	Disable interrupt flag
I010	M8059
I020	
I030	

Pointer number	Disable interrupt flag
I040	M8059
I050	
I060	

Note: The pointer number cannot be reused.



- Allow interruption after EI instruction, and write the main program.
- Drive the coil of the high-speed counter, and specify the interrupt pointer in the DHSCS instruction.
- When the current value of C255 changes from 999→1000 or 1001→1000, the interrupt subroutine is executed.

4 How to Specify Devices and Constants to Instructions

4.1 Data processed by PLC

4.1.1 Types of numeric values

1. Decimal numbers (DEC: DECIMAL NUMBER), Specify with constant K

- ◆ The specified range of decimal constant: When using word data (16 bits) • • • • • K-32768～K32767
  When using 2 word data (32 bits) • • K-2,147,483,648 to K2,147,483,647
- ◆ As the setting value of timer T and counter C, for example: OUT C10 K50. (K constant)
- ◆ S, M, T, C, D, Z, V, P, I and other device numbers, for example: M10, T30. (Device number)
- ◆ Used as an operand in application instructions, for example: MOV K123 D0. (K constant)

2. Hexadecimal number (HEX: HEXADECIMAL NUMBER), specified by the constant H

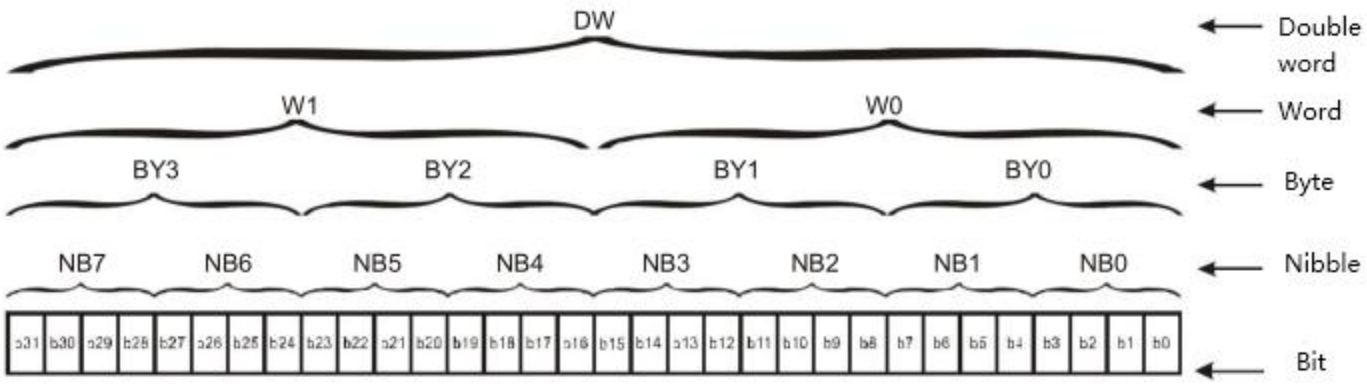
- ◆ The specified range of hexadecimal constant: When using word data (16 bits) • • • • • H0 to HFFFF (H0 to H9999 for BCD data)
  When using 2 word data (32 bits) • • H0 to HFFFFFFFF (H0 to H99999999 for BCD data)
- ◆ Used as an operand in application instructions, for example: MOV HAB10 D0. (H constant)

3. Binary numbers (BIN: BINARY NUMBER)

The numerical calculation or storage inside the PLC adopts binary system. The binary value and related terms are as follows:

Bit:	Bit is the most basic unit of binary value, and its state is either 1 or 0.
Nibble:	It is composed of 4 consecutive digits (such as b3 ~ b0), which can be used to represent a decimal number 0 ~ 9 or hexadecimal 0 ~ F.
Byte:	It is composed of two consecutive nibbles (that is, 8 bits, b7 ~ b0), which can represent 00 ~ FF in hexadecimal.
Word:	It is composed of two consecutive bytes (that is, 16 bits, b15 ~ b0), which can represent the hexadecimal 4-digit value 0000 ~ FFFF.
Double Word:	It is composed of two consecutive words (that is, 32 bits, b31 ~ b0), which can represent the 8-bit value 00000000 ~ FFFFFFFF in hexadecimal.

The relationship between bits, nibbles, bytes, words, and double words in the binary system is shown in the following figure:



4. Octal numbers (OCT: OCTAL NUMBER)

In PLC, the device numbers of input relays and output relays are all assigned in octal numbers.

5. BCD(BCD: BINARY CODE DECIMAL)

A decimal data is represented by half a byte or 4 bits, so continuous 16 bits can represent 4-digit decimal numerical data; it is mainly used in BCD output type digital switch and 7-segment display control.

6. Real number (floating point number data), specified with constant E

- In Coolmay PLC, it has the floating point arithmetic function that can perform high-precision arithmetic. Use binary floating-point numbers (real numbers) for floating-point operations, and use decimal floating-point numbers (real numbers) for monitoring. Such as E12.34=12.34.
- The specified range of real numbers: -1.0×2^{128}～-1.0×2^{-126}, 0, 1.0×2^{-126}～1.0×2^{128}。
- In the sequence program, the real number can be specified with "normal expression" and "exponential expression".

Ordinary representation Specify the set value directly. For example, 10.2345 is specified as E10.2345.

Index representation The set value is specified by (numerical value)×10n. For example, 1234 is specified as E1.234+3. The [+3] of [E1.234+3] represents 10 to the nth power (+3 is 103).

4.1.2 Value conversion

The data processed in the PLC can be converted according to the following table.

Decimal number (DEC)	Octal number (OCT)	Hexadecimal number (HEX)	Binary number (BIN)		BCD	
0	0	00	0000	0000	0000	0000
1	1	01	0000	0001	0000	0001
2	2	02	0000	0010	0000	0010
3	3	03	0000	0011	0000	0011
4	4	04	0000	0100	0000	0100
5	5	05	0000	0101	0000	0101
6	6	06	0000	0110	0000	0110
7	7	07	0000	0111	0000	0111
8	10	08	0000	1000	0000	1000
9	11	09	0000	1001	0000	1001
10	12	0A	0000	1010	0001	0000

11	13	0B	0000	1011	0001	0001
12	14	0C	0000	1100	0001	0010
13	15	0D	0000	1101	0001	0011
14	16	0E	0000	1110	0001	0100
15	17	0F	0000	1111	0001	0101
16	20	10	0001	0000	0001	0110
…	…	…	…	…	…	…
99	143	63	0110	0011	1001	1001
…	…	…	…	…	…	…

4.1.3 Numerical processing in floating-point arithmetic

In the division operation of an integer, an answer such as 23÷2=11 with a remainder of 1 will be obtained; in the same way, the decimal point should not be discarded in the square root operation of an integer. In the PLC, in order to perform these operations with higher accuracy, floating-point operations can be performed.

◆ **Binary floating point number (real number)**

1. When handling binary floating-point numbers (real numbers) in data registers, use a pair of data registers with consecutive numbers. For example, (D11,D10), as shown in the figure below:

← D11(b15~b0) →										← D10(b15~b0) →				
	2^7	2^6	2^5		2^1	2^0	2^{-1}	2^{-2}	2^{-3}		2^{-20}	2^{-21}	2^{-22}	2^{-23}
S	E7	E6	E5	......	E1	E0	A22	A21	A20	......	A3	A2	A1	A0
b31	b30	b29	b28	......	b24	b23	b22	b21	b20	......	b3	b2	b1	b0
Mantissa part symbol (0: positive, 1: negative)	← 8 digits for the exponent → E0~E7=0 or 1						← Mantissa part 2 → A0~A22=0 or 1 When b0~b31=0 is 0							

Binary floating point number (real number) = ±(20+A22×2-1+A21×2-2+…+A0×2-23)×2 $^{(E7X27+E6X26+...+E0X20)}/2^{127}$

(E.g) A22=1,A21=0,A20=1,A19～A0=0,E7=1,E6～E1=0,E0=1

Binary floating point number (real number) = ±(20+1×2-1+0×2-2+1×2-3+…+0×2-23)×2 $^{(1X27+0X26+...+1X20)}/2^{127}$

$= \pm 1.625 \times 2^{129}/2^{127}$

$= \pm 1.625 \times 2^{2}$

2. The effective digits of a binary floating-point number, such as a decimal number, are about 7 digits. The processing range of binary floating-point numbers is as follows:

—Minimum absolute value 1175494×10-44

—Maximum absolute value 3402823×1032

3. Handling of the zero (M8020), borrow (M8021) and carry (M8022) flags

—Zero flag : 1 when the result is 0

—Borrow flag : 1 when the result does not reach the minimum unit but is not 0

—Carry flag : 1 when the absolute value of the result exceeds the available numeric value range.

4. Monitoring of binary floating point (real number)

In the PLC programming software, GX Developer supports floating-point number display and can directly monitor binary floating-point numbers (real numbers). In addition, in programming tools that do not support floating-point number display, you can convert binary floating-point numbers (real numbers) into decimal floating-point numbers (real numbers) before monitoring.

◆ **Decimal floating point number (real number)**

1. For users, binary floating-point numbers (real numbers) are difficult to understand, so they can also be converted into decimal floating-point numbers (real numbers). However, the operation inside the PLC still uses binary floating-point numbers (real numbers).

2. When dealing with decimal floating-point numbers (real numbers) in data registers, use a pair of data registers with consecutive numbers, but with binary floating-point numbers (real numbers)

Different, the smaller number is the mantissa part, and the larger number is the exponent part.

For example, when using the data registers (D1, D0) as shown below, use the MOV instruction to write to D0 and D1.

Decimal floating point numbers (real number) = [Mantissa D0] × 10 [Exponent D1]

Mantissa D0 = ± (1000 to 9999) or 0

Exponent D1 = −41 to +35

The mantissa D0 does not allow "100", for example. In the case of "100", it is handled as "1000 × 10 −1 ".

3. Decimal floating point numbers (real number) range is as follows:

—Minimum absolute value 1175×10^{-41}

—Maximum absolute value 3402×10^{35}

4.2 Bit Specification

4.2.1 Specification of Digits for Bit Devices（Kn 口***)

◆ Handling of bit devices

1. For bit devices, numerical values are processed by the combination of the number of digits Kn and the number of the start device. The number of bits is in units of 4 bits, K1～K4 (16-bit data), K1～K8 (32-bit data). For example, K2M0 is 2-digit data because it is M0 to M7.

2. After transmitting 16-bit data to K1M0～K3M0, the upper part of the insufficient data length will not be transmitted. The same is true for 32-bit data.

3. In the process of 16-bit (or 32-bit) operation, when the number of bits K1～K3 (or K1～K7) is specified for the bit device, the insufficient high bits are always regarded as 0. Therefore, always handle positive numbers.

Such as BIN K2X004 D0 (convert BCD 2-digit data into BIN through X004～X013 and then send it to DO)

4. As long as there are no special restrictions, the number of the designated bit device can be arbitrary, but it is recommended to set the lowest bit number to 0 in the case of X and Y. (Specify X000, X010, X020...Y000, Y010, Y020... etc.)

In the case of M, S, the most ideal is a multiple of 8, but in order to avoid confusion, it is recommended to set M0, M10, M20... etc.

◆ Designation of consecutive words

A series of data registers starting with D1 are D1, D2, D3, D4... In the case of words, it can be treated as a series of words by specifying the number of digits. As shown below:

• K1X000, K1X004, K1X010, K1X014……

• K2Y010, K2Y020, Y2X030……

• K3M0, K3M12, K3M24, K3M36……

• K4S16, K4S32, K4S48……

In other words, without skipping the device, use the device in the unit of the number of bits as shown above.

However, when K4Y000 is used in a 32-bit operation, the upper 16 bits are regarded as 0. When 32-bit data is required, K8Y000 is used.

4.2.2 Bit Specification of a Word Device（D 口.b)

Specify the bit of a word device, you can use it as bit data, a word device has 16 bits; when specifying the bit of a word device, please use the word device number and bit number (hexadecimal number) to set .

Such as: MOV K4M0 D0 (transmit the state of the 16 bits of M0-M15 to b0-b15 of D0)

5 Basic Instruction

Mnemonic	Name	Function	Applicable devices	Program step
Contact Instruction				
LD	Load	Initial logical operation contact type NO (normally open)	X,Y,M,S,T,C,D □.b	1
LDI	Load Inverse	Initial logical operation contact type NC (normally closed)	X,Y,M,S,T,C,D □.b	1
LDP	Load Pulse	Initial logical operation of rising edge pulse	X,Y,M,S,T,C,D □.b	2
LDF	Load Falling Pulse	Initial logical operation of falling/trailing edge pulse	X,Y,M,S,T,C,D □.b	2
AND	AND	Serial connection of NO (normally open) contacts	X,Y,M,S,T,C,D □.b	1
ANI	AND Inverse	Serial connection of NC (normally closed) contacts	X,Y,M,S,T,C,D □.b	1
ANDP	AND Pulse	Serial connection of rising edge pulse	X,Y,M,S,T,C,D □.b	2
ANDF	AND Falling Pulse	Serial connection of falling/ trailing edge pulse	X,Y,M,S,T,C,D □.b	2
OR	OR	Parallel connection of NO (normally open) contacts	X,Y,M,S,T,C,D □.b	1
ORI	OR Inverse	Parallel connection of NC (normally closed) contacts	X,Y,M,S,T,C,D □.b	1
ORP	OR Pulse	Parallel connection of rising edge pulse	X,Y,M,S,T,C,D □.b	2
ORF	OR Falling Pulse	Parallel connection of falling/trailing edge pulse	X,Y,M,S,T,C,D □.b	2
Connection Instruction				
ANB	AND Block	Serial connection of multiple parallel circuits		1
ORB	OR Block	Parallel connection of multiple contact circuits		1
MPS	Memory Point Store	Stores the current result of the internal PLC operations		1
MRD	Memory Read	Reads the current result of the initial PLC operations		1
MPP	Memory POP	Pops (recalls and removes) the currently stored result		1
INV	Inverse	Invert the current result of the internal		1

		PLC operations		
MEP	M.E.P	Conversion of operation result to leading edge pulse		
MEF	M.E.F	Conversion of operation result to trailing edge pulse		
Out Instruction				
OUT	OUT	Final logical operation type coil drive	Y,M,S,T,C,D □.b	Note 1
SET	SET	Set bit device latch ON	Y,M,S,D □.b	Note 2
RST	Reset	Reset bit device OFF	Y,M,S,T,C,D,V,Z,D □.b	
PLS	Pulse	Rising edge pulse	Y,M	
PLF	Pulse Falling	Falling/trailing edge pulse	Y,M	
Master Control Instruction				
MC	Master Control	Denotes the start of a master control block	Y,M	3
MCR	Master Control Reset	Denotes the end of a master control block		2
Other Instruction				
NOP	No Operation	No operation or null step		1
End Instruction				
END	END	Program end, I/O refresh and return to step 0		1

5.1 LD, LDI instructions

LD and LDI instructions are the contacts connected to the bus. After being combined with the ANB instruction described later, it can also be used at the start of a branch.

Mnemonic	Function	Applicable models
LD (Load)	The logic operation of the normally open contact (A) starts	Coolmay series PLC
Operand	X, Y, M, S, T, C, D □.b	
Instructions	The LD instruction is used for the A contact at the beginning of the left bus or the A contact at the beginning of a contact circuit block. Its function is to save the current content and at the same time store the fetched contact status into the accumulator.	

Program example

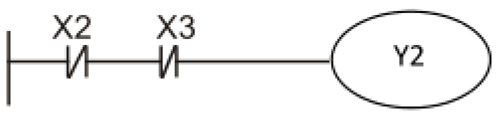
Ladder diagram:

Mnemonic:		Explanation:
LD	X0	Input A contact of X0
AND	X1	Series A contact of X1
OUT	Y1	Drive Y1 coil

Mnemonic	Function	Applicable models
LDI (Load Inverse)	The logic operation of the normally closed contact (B) starts	Coolmay series PLC
Operand	X, Y, M, S, T, C, D □.b	
Instructions	The LD instruction is used for the B contact at the beginning of the left bus or the A contact at the beginning of a contact circuit block. Its function is to save the current content and at the same time store the fetched contact status into the accumulator.	

Program example

Ladder diagram:



Mnemonic:		Explanation:
LDI	X2	Input B contact of X2
ANI	X3	Series B contact of X3
OUT	Y2	Drive Y2 coil

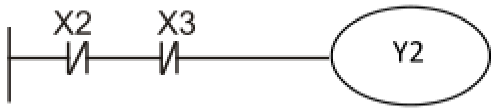
5.2 OUT instructions

OUT instruction drives coils of output relays (Y), auxiliary relays (M), state relays (S), timers (T) and counters (C).

Mnemonic	Function	Applicable models
OUT (OUT)	Drive coil	Coolmay series PLC
Operand	X, Y, M, S, T, C, D □.b	
Instructions	Output the result of the logic operation before the OUT instruction to the specified device.	

Program example

Ladder diagram:



Mnemonic:

LDI	X2	Input B contact of X2
ANI	X3	Series B contact of X3
OUT	Y2	Drive Y2 coil

Explanation:

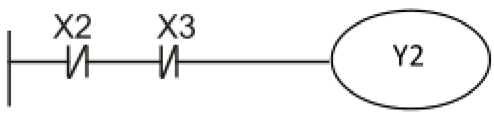
5.3 AND, ANI instructions

The AND and ANI instructions are executed to connect 1 contact in series. There is no limit to the number of serial contacts, this command can be used multiple times in succession.

Mnemonic	Function	Applicable models
AND (AND)	Series A contact	Coolmay series PLC
Operand	X, Y, M, S, T, C, D □.b	
Instructions	The AND instruction is used for the series connection of A contacts.	

Program example

Ladder diagram:



Mnemonic:

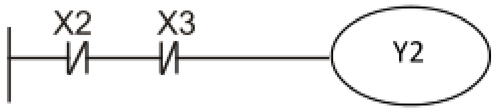
LDI	X2	Input B contact of X2
ANI	X3	Series B contact of X3
OUT	Y2	Drive Y2 coil

Explanation:

Mnemonic	Function	Applicable models
ANI (AND Inverse)	Series B contact	Coolmay series PLC
Operand	X, Y, M, S, T, C, D □.b	
Instructions	ANI instruction is used for series connection of B contacts	

Program example

Ladder diagram:



Mnemonic:

Mnemonic		Explanation
LDI	X2	Input B contact of X2
ANI	X3	Series B contact of X3
OUT	Y2	Drive Y2 coil

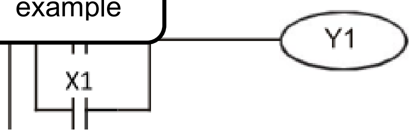
5.4 OR, ORI instructions

OR and ORI instructions can be used as instructions for connecting 1 contact in parallel. When two or more contacts are connected in series, to connect such a series circuit block with other circuits in parallel, use the ORB instruction described later. OR and ORI start from the step of this instruction and are connected in parallel with the steps of the previous LD and LDI instructions. The number of parallel connections is not limited.

Mnemonic	Function	Applicable models
OR	Parallel A contact	Coolmay series PLC
Operand	X, Y, M, S, T, C, D □.b	
Instructions	OR instruction is used for parallel connection of A contacts	

Program example

Ladder diagram:

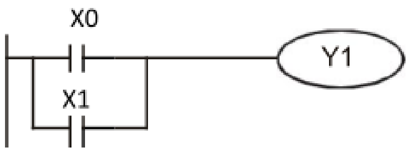


Mnemonic:		Explanation:
LD	X0	Load the A contact of X0
OR	X1	Parallel A contact of X1
OUT	Y1	Drive Y1 coil

Mnemonic	Function	Applicable models
ORI (OR Inverse)	Parallel B contact	Coolmay series PLC
Operand	X, Y, M, S, T, C, D □.b	
Instructions	ORI instruction is used for parallel connection of B contacts	

Program example

Ladder diagram:



Mnemonic:		Explanation:
LD	X0	Load the A contact of X0
ORI	X1	Parallel B contact of X1
OUT	Y1	Drive Y1 coil

5.5 LDP, LDF, ANDP, ANDF, ORP, ORF instructions

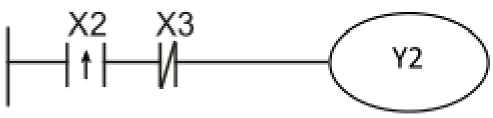
LDP, ANDP, ORP instructions are contact instructions that detect the rising edge. Only when the rising edge of the specified bit device (changes from OFF to ON) is turned on, one operation cycle is turned on.

LDF, ANDF, ORF instructions are contact instructions that detect the falling edge. Only when the falling edge of the specified bit device (changes from ON to OFF), one operation cycle is turned on.

Mnemonic	Function	Applicable models
LDP **(Load Pulse)**	Start of rising edge detection action	Coolmay series PLC
Operand	X, Y, M, S, T, C, D □.b	
Instructions	The usage of LDP instruction is the same as LD, but the action is different. Its function is to save the current content, and at the same time save the fetched contact rising edge detection state into the accumulator.	

Program example

Ladder diagram:



Mnemonic:		Explanation:
LDP	X2	X2 rising edge detection starts
ANI	X3	Series B contact of X3
OUT	Y2	Drive Y2 coil

Mnemonic	Function	Applicable models
LDF **(Load Falling Pulse)**	Start of falling edge detection	Coolmay series PLC
Operand	X, Y, M, S, T, C, D □.b	
Instructions	The usage of the LDF instruction is the same as LD, but the action is different. Its function is to save the current content, and at the same time store the fetched contact rising edge detection state into the accumulator.	

Program example

Ladder diagram:

Mnemonic:		Explanation:
LDP	X2	X2 falling edge detection action starts
ANI	X3	Series B contact of X3
OUT	Y2	Drive Y2 coil

Mnemonic	Function	Applicable models

ANDP **(AND Pulse)**	Rising edge detection series connection	Coolmay series PLC
Operand	X, Y, M, S, T, C, D □.b	
Instructions	The ANDP instruction is used for series connection with the detection of the rising edge of the contact.	

Program example

Ladder diagram:

Mnemonic:

LD	X2
ANDP	X3
OUT	Y2

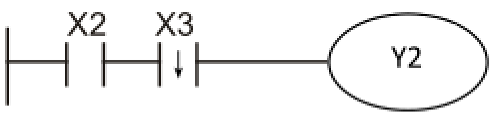
Explanation:

Load the A contact of X2
X3 rising edge detection series connection
Drive Y2 coil

Mnemonic	Function	Applicable models
ANDF **(AND Falling Pulse)**	Falling edge detection series connection	Coolmay series PLC
Operand	X, Y, M, S, T, C, D □.b	
Instructions	The ANDF instruction is used for series connection where the falling edge of the contact is detected.	

Program example

Ladder diagram:



Mnemonic:

LD	X2
ANDP	X3
OUT	Y2

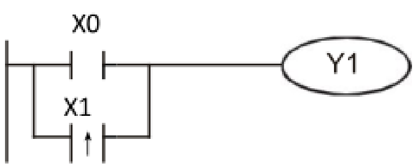
Explanation:

Load the A contact of X2
X3 falling edge detection series connection
Drive Y2 coil

Mnemonic	Function	Applicable models
ORP **(OR Pulse)**	Parallel connection detected on rising edge	Coolmay series PLC
Operand	X, Y, M, S, T, C, D □.b	
Instructions	ORP instruction is used for parallel connection with detection of rising edge of contact	

Program example

Ladder diagram:



Mnemonic:

LD	X2
ORP	X1
OUT	Y1

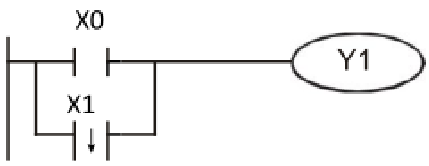
Explanation:

Load the A contact of X0
X1 rising edge detection parallel connection
Drive Y1 coil

Mnemonic	Function	Applicable models
ORF **(OR Falling Pulse)**	Falling edge detection parallel connection	Coolmay series PLC
Operand	X, Y, M, S, T, C, D □.b	
Instructions	ORF instruction is used for parallel connection with detection of rising edge of contact	

Program example

Ladder diagram:



Mnemonic:		Explanation:
LD	X0	Load the A contact of X0
ORF	X1	X1 falling edge detection parallel connection
OUT	Y1	Drive Y1 coil

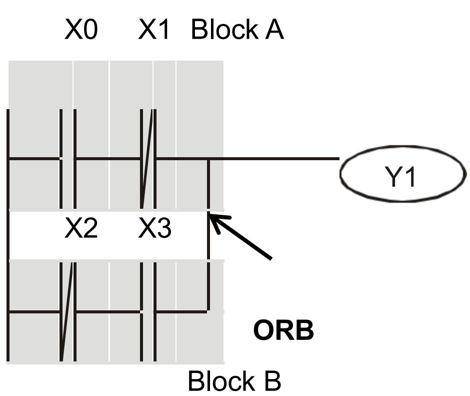
5.6 ORB Instructions

A circuit in which two or more contacts are connected in series is called serial circuit block.

Mnemonic	Function	Applicable models
ORB **(OR Block)**	Parallel connection of circuit blocks	Coolmay series PLC
Operand	Null	
Instructions	ORB is the "OR" operation between the previously saved logic result and the current accumulator content.	

Program example

Ladder diagram:



Mnemonic:		Explanation:
LD	X0	Load the A contact of X0
ANI	X1	Series B contact of X1
LDI	X2	Load B contact of X2
AND	X3	A contact of X3 in series
ORB		Parallel circuit block
OUT	Y1	Drive Y1 coil

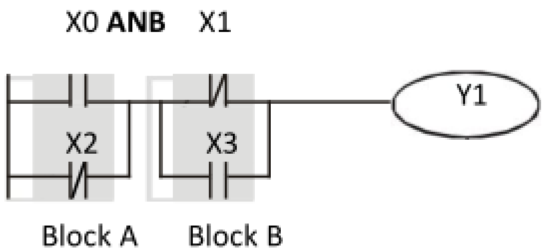
5.7 ANB Instructions

When the branch circuit (parallel circuit block) is connected in series with the previous circuit, use the ANB instruction. The starting point of the branch uses the LD and LDI instructions. After the parallel circuit block ends, you can use the ANB instruction to connect to the previous circuit in series. When there are multiple parallel circuits, use the ANB instruction for each circuit block to connect.

Mnemonic	Function	Applicable models
ANB (AND Block)	Series connection of circuit blocks	Coolmay series PLC
Operand	Null	
Instructions	ANB is the AND operation between the previously saved logic result and the current accumulator content.	

Program example

Ladder diagram:



Mnemonic:		Explanation:
LD	X0	Load the A contact of X0
ORI	X2	Parallel the B contact of X2
LDI	X1	Load the B contact of X1
OR	X3	Parallel A contact of X3
ANB		Series loop block
OUT	Y1	Drive Y1 coil

5.8 MPS, MRD, MPP Instructions

Mnemonic	Function	Applicable models
MPS (Memory Point Store)	Save to stack	Coolmay series PLC
Operand	Null	
Instructions	Store the contents of the current accumulator on the stack. (The stack pointer plus one).	

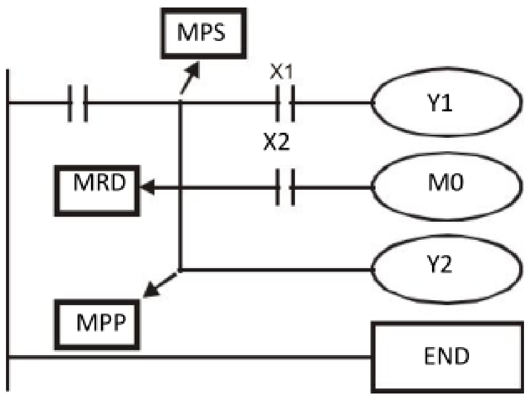
Mnemonic	Function	Applicable models
MPD (Memory Read)	Read the stack (pointer does not move)	Coolmay series PLC
Operand	Null	
Instructions	Read the stack content and store it in the accumulator. (The stack pointer does not move)	

Mnemonic	Function	Applicable models

MPP (Memory POP)	Read stack	Coolmay series PLC
Operand	Null	
Instructions	Retrieve the previously saved logical operation result from the stack and store it in the accumulator. (The stack pointer minus one)	

Program example

Ladder diagram:



Mnemonic:		Explanation:
LD	X0	Load the A contact of X0
MPS		**Save to stack**
AND	X1	Series A contact of X1
OUT	Y1	Drive Y1 coil
MRD		**Read the stack (pointer does not move)**
AND	X2	A contact of X2 in series
OUT	M0	Drive M0 coil
MPP		**Read stack**
OUT	Y2	Drive Y2 coil
END		End of program

5.9 MC, MCR Instructions

After executing the MC instruction, the bus (LD, LDI point) moves to behind the MC contact; using the MCR instruction, it can be returned to the original bus position.

By changing the device numbers Y and M, the MC instruction can be used multiple times. However, when the same device number is used, it is the same as the OUT instruction, and double coil output will appear.

<table>
<tr><th>Mnemonic</th><th colspan="2">Function</th><th>Applicable models</th></tr>
<tr><td>MC/MCR
(Master Control /
Master Control
Reset)</td><td colspan="2">Connection/disconnection of common series contacts</td><td>Coolmay series PLC</td></tr>
<tr><td>Operand</td><td colspan="3">Y, M (except special auxiliary relays)</td></tr>
<tr><td rowspan="4">Instructions</td><td colspan="3">MC is the main control start instruction. After the MC instruction is executed, the instructions between MC and MCR instructions are executed as usual; when the MC instruction is OFF, the instructions between MC and MCR instructions are executed as follows:</td></tr>
<tr><td>Command distinction</td><td colspan="2">Description</td></tr>
<tr><td>General timer / timer for subroutine</td><td colspan="2">The timing value returns to zero, the coil is de-energized, and the contact does not operate</td></tr>
<tr><td>Accumulative timer/counter</td><td colspan="2">The coil is de-energized, and the count value</td></tr>
</table>

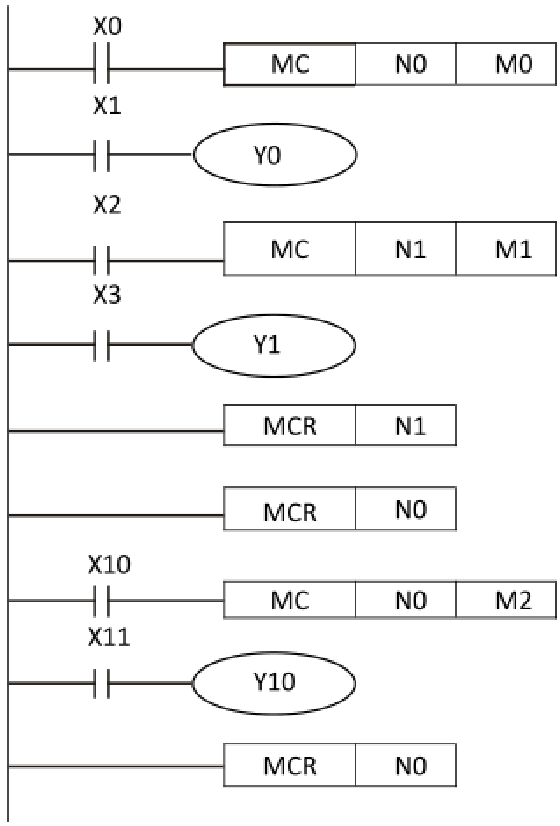
		and contacts keep the current state
	Coil driven by OUT instruction	All turned off
	SET, RST instruction driven device	Keep current

Note: When using MC instruction, the number of nesting level N increases in sequence: (N0→N1→N2→N3→N4→N5→N6→N7)

When returning, use MCR instruction to release from the larger nesting level: (N7→N6→N5→N4→N3→N2→N1→N0)

Program example

Ladder diagram:



Mnemonic:		Explanation:
LD	X0	Load the A contact of X0
MC	N0	N0 Connection of common series contacts
LD	X1	Load the A contact of X1
OUT	Y0	Drive Y0 coil
LD	X2	Load the A contact of X2
MC	N1	N1 Connection of common series contacts
LD	X3	Load the A contact of X3
OUT	Y1	Drive Y1 coil
MCR	N1	N1 Release of common series contact
MCR	N0	N0 Release of common series contact
LD	X10	Load the A contact of X10
MC	N0	N0 Connection of common series contacts
LD	X11	Load the A contact of X11
OUT	Y10	Drive Y10 coil
MCR	N0	Release of N0 common series contact

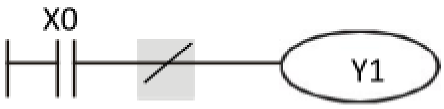
5.10 INV instructions

The NV instruction is an instruction that reverses the calculation result before the INV instruction is executed, and there is no need to specify the device number.

Mnemonic	Function	Applicable models
INV (Inverse)	Inversion of operation result	Coolmay series PLC
Operand	Null	
Instructions	Invert the result of the logic operation before the INV instruction and store it in the accumulator.	

Program example

Ladder diagram:



Mnemonic:

LD	X0
INV	
OUT	Y1

Explanation:

Load the A contact of X0
Inverted result
Drive Y1 coil

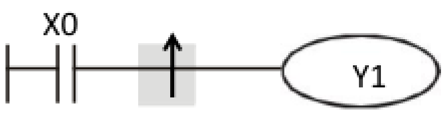
5.11 MEP, MEF instructions

The MEP and MEF instructions are instructions that pulse the result of the operation, and there is no need to specify the device number.

Mnemonic	Function	Applicable models
MEP	On on rising edge	Coolmay series PLC
Operand	Null	
Instructions	The result of the operation up to the MEP instruction changes from OFF to ON to turn on.	

Program example

Ladder diagram:

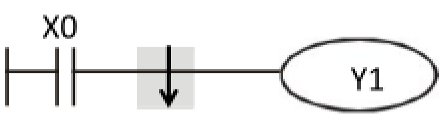


Mnemonic:		Explanation:
LD	X0	Load the A contact of X0
MEP		On rising edge
OUT	Y1	Drive Y1 coil

Mnemonic	Function	Applicable models
MEF	On at falling edge	Coolmay series PLC
Operand	Null	
Instructions	The result of the operation up to the MEF instruction is turned on when it changes from ON to OFF.	

Program example

Ladder diagram:



Mnemonic:		Explanation:
LD	X0	Load the A contact of X0
MEF		On at falling edge
OUT	Y1	Drive Y1 coil

5.12 PLS, PLF instructions

After using the PLS instruction, the target device operates only in one operation cycle after the drive input is turned on.

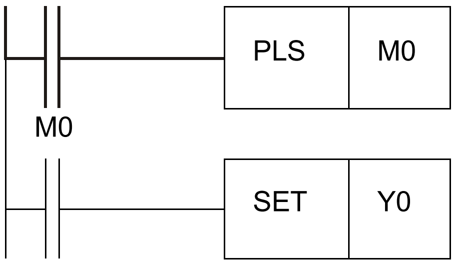
After the PLF instruction is used, the target device operates only in one operation cycle after the drive input is turned off.

Mnemonic	Function	Applicable models
PLS (Pulse)	Rising edge differential output	Coolmay series PLC
Operand	Y, M (except special auxiliary relays)	

Instructions	When X0=OFF→ON (rising edge trigger), PLS instruction is executed, S sends out a pulse, and the pulse width is one scan period.

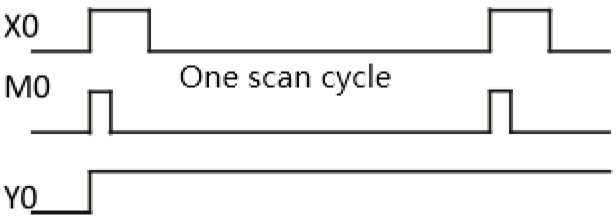
Program example

Ladder diagram:



Mnemonic:		Description:
LD	X0	Load the A contact of X0
PLS	M0	M0 rising edge detection
LD	M0	Load the A contact of M0
SET	Y0	Y0 action hold (ON)

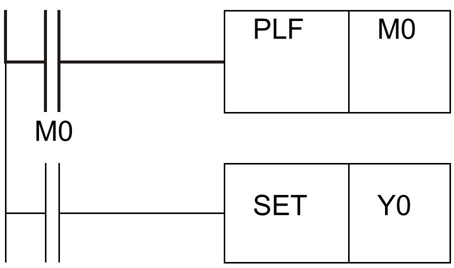
Timing diagram:



Mnemonic	Function	Applicable models
PLF (Pulse Falling)	Falling edge differential output	Coolmay series PLC
Operand	Y, M (except special auxiliary relays)	
Instructions	When X0=ON→OFF (rising edge trigger), PLS instruction is executed, S sends out a pulse, and the pulse width is one scan period.	

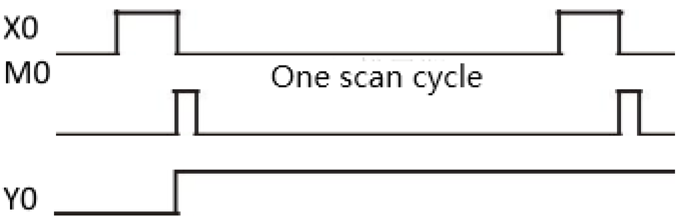
Program example

Ladder diagram:



Mnemonic:		Description:
LD	X0	Load the A contact of X0
PLF	M0	M0 falling edge detection
LD	M0	Load the A contact of M0
SET	Y0	Y0 action hold (ON)

Timing diagram:

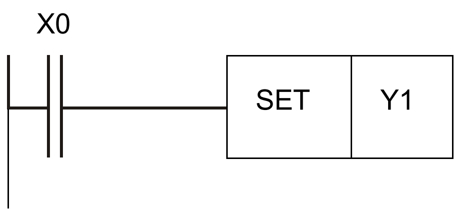


5.13 SET, RST instructions

Mnemonic	Function	Applicable models
SET（SET)	Action keep (ON)	Coolmay series PLC
Operand	Y，M（Except for special auxiliary relays), S, D □.b	
Instructions	When the SET instruction is driven, the specified element is set to ON, and the set element will remain ON, regardless of whether the SET instruction is still driven. The RST instruction can be used to turn this component OFF.	

Program example

Ladder diagram:

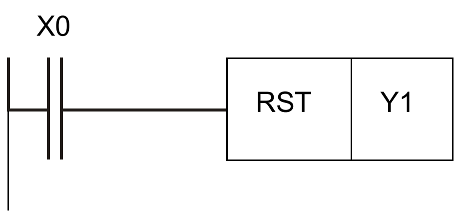


Mnemonic:		Description:
LD	X0	Load the A contact of X0
SET	Y1	Y1 action hold

Mnemonic	Function	Applicable models
RST（Reset)	Clear contact or register	Coolmay series PLC
Operand	Z，M（Except for special auxiliary relays), S, D □.b, T, C, D, R, V, Z	
Instructions	When the RST instruction is driven, the actions of the specified components are as follows: Components S, Y, M: coils and contacts are all set to OFF; Component T, C: The current timing or count value is cleared, and the coil and contact status are set to OFF; Component D, Z, V: The content value is cleared to 0.	

Program example

Ladder diagram:



Mnemonic:		Description:
LD	X0	Load the A contact of X0
RST	Y1	Y1 contact clear

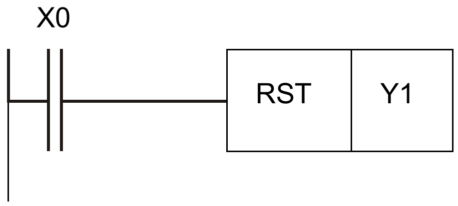
5.14 NOP instruction

Mnemonic	Function	Applicable models
NOP（No Operation)	No treatment	Coolmay series PLC
Operand	Null	
Instructions	The NOP instruction does not perform any operation in the program, so the original	

	logic operation result will be maintained after execution. The component machine is as follows: If you want to delete a certain instruction but do not want to change the program length, you can replace it with the NOP instruction.

Program example

Ladder diagram:



Mnemonic:		Description:
LD	X0	Load the A contact of X0
NOP		No action
RST	Y1	Y1 contact clear

5.15 END instruction

Mnemonic	Function	Applicable models
END (END)	End of the program and input and output processing and return to step 0	Coolmay series PLC
Operand	Null	
Instructions	The END instruction must be added at the end of the ladder program or instruction program. The PLC scans from address 0 to the END instruction, and after execution, returns to address 0 to scan again.	

6 Program Flow

FNC NO.	Mnemonic	Function	Support models		
			3G series PLC	2N series PLC	MX2N series PLC
00	CJ	Conditional Jump	★	★	★
01	CALL	Call Subroutine	★	★	★
02	SRET	Subroutine Return	★	★	★
03	IRET	Interrupt Return	★		★
04	EI	Enable Interrupt	★		★
05	DI	Disable Interrupt	★		★
06	FEND	Main Routine Program End	★	★	★
07	WDT	Watchdog Timer Refresh	★	★	★
08	FOR	Start a FOR/NEXT Loop	★	★	★
09	NEXT	End a FOR/NEXT Loop	★	★	★

6.1 CJ/ Conditional jump

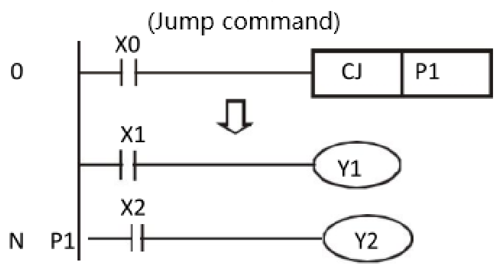
It is an instruction that does not execute the sequence program starting from the CJ and CJP instructions to the pointer (P). It is possible to shorten the cycle time (calculation cycle) and execute the program using double coils.

Instruction		Operand type	Function						
			16-bit Instruction	Mnemonic	Operation Condition		32-bit Instruction	Mnemonic	Operation Condition
FNC 00 CJ		Pn.	3 steps	CJ	Continuous Operation			——	
	P			CJP	Pulse (Single) Operation			——	
Operand		Pn.	Pointer P supports index modification; FX2N: n=0~127, FX3G: n=0~1280; but P63 is an END jump.						

Instruction

- When the command input is ON, the program with the designated mark (pointer number) is executed.
- The program pointed to by the pointer P can be before the CJ instruction, but when the scan time exceeds 200ms (default setting), a watchdog timer (WDT) error will occur. Please be careful.
- The CJ instruction can specify the same pointer P repeatedly, but CJ and CALL cannot specify the same pointer P, otherwise an error will occur.
- here is no need to input the mark of pointer P63, otherwise the PLC will prompt an error and stop running.

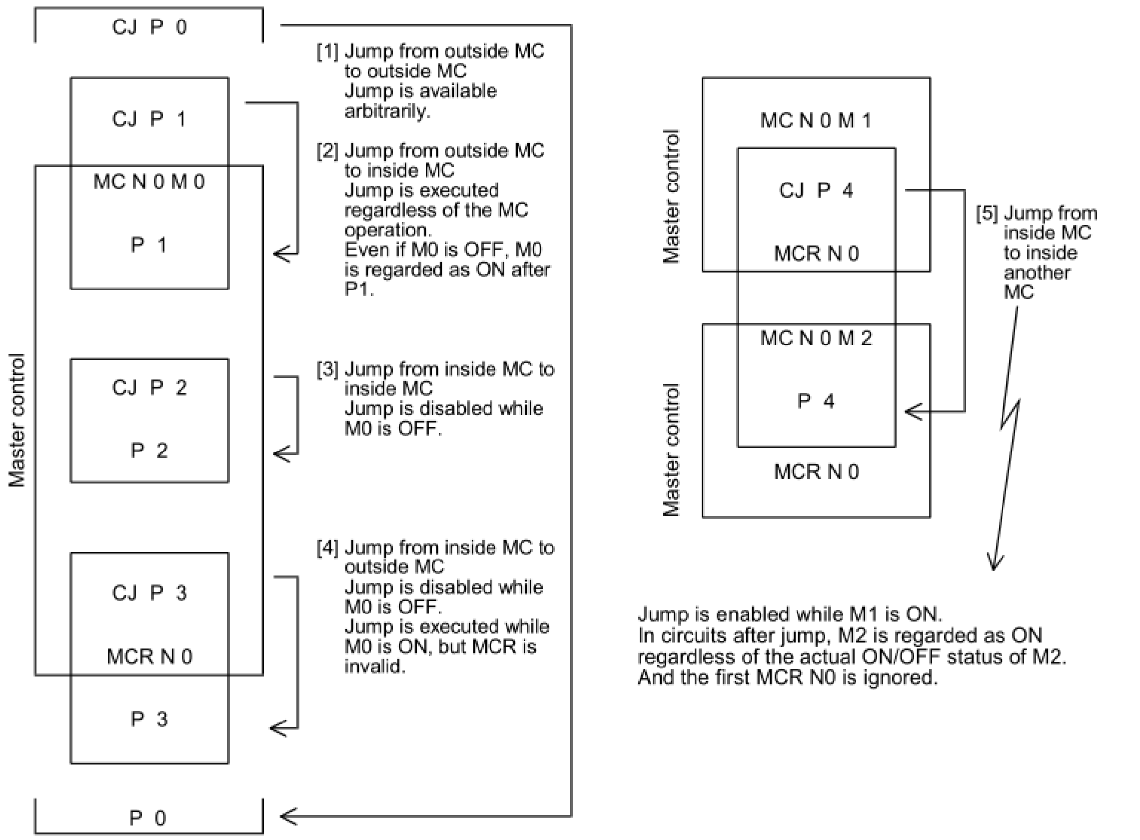
Sample program (1)



- When X0=ON, the program automatically jumps from address 0 to address N (namely the designated pointer P1) to continue execution, and the intermediate address skips and does not execute.
- When X0=OFF, the program continues to execute from address 0 as a normal program, and the CJ instruction is not executed at this time.

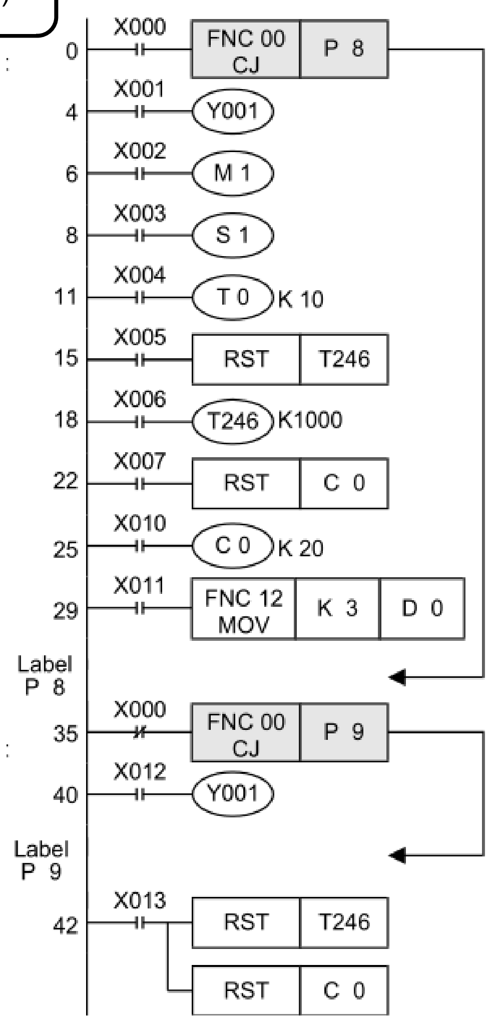
The relationship between the main control instruction and the jump instruction and the action content are as follows: (Because the operation of ②, ④, ⑤ will become complicated, please avoid using it)

Sample program (2)



Jump is enabled while M1 is ON.
In circuits after jump, M2 is regarded as ON regardless of the actual ON/OFF status of M2.
And the first MCR N0 is ignored.

Sample program (3)



- Double coil operation of output Y001
  While X000 is OFF, output Y001 is activated by X001.
  While X000 is ON, output Y001 is activated by X012.
  Even in a program divided by conditional jumps, if a same coil (Y000 in this case) is programmed two or more times within the jump area or outside the jump area, such a coil is handled as double coil.

- When the reset (RST) instruction for the retentive type timer T246 is located outside the jump area
  Even if the counting coil (OUT T246) is jumped, reset (return of the contact and clearing of the current value) is valid.

- When the reset (RST) instruction for the counter C0 is located outside the jump area
  Even if the counting coil is jumped, reset (return of the contact and clearing of the current value) is valid.

- Operation of the routine timers T192 to T199
  A routine timer continues its operation even if it is jumped after the coil is driven, and the output contact is activated.

- Operation of the high speed counters C235 to C255
  A high speed counter continues its operation even if it is jumped after the coil is driven, and the output contact is activated.

6.2 CALL/ Subroutine call

In a sequence program, an instruction to call a program that you want to process together. You can reduce the number of steps in the program and design the program more effectively.

In addition, when writing subroutines, you also need to use FEND (FNC 06) and SRET (FNC 02) instructions.

<table>
<tr><th colspan="2">Instruction</th><th>Operand type</th><th colspan="7">Function</th></tr>
<tr><td rowspan="3">FNC 01
CALL</td><td rowspan="2"></td><td rowspan="3">Pn.</td><td>16-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td><td rowspan="3"></td><td>32-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td></tr>
<tr><td rowspan="2">3 steps</td><td>CALL</td><td>Continuous Operation</td><td rowspan="2"></td><td>——</td><td rowspan="2"></td></tr>
<tr><td>P</td><td>CALLP</td><td>Pulse (Single) Operation</td><td>——</td></tr>
<tr><td colspan="2">Operand</td><td>Pn.</td><td colspan="7">Pointer P, supports index modification; where FX2N: n=0~62,64~127, FX3G: n=0~62,64~1280;
P63 is a dedicated END jump for CJ and cannot be used as a pointer to the CALL instruction.</td></tr>
</table>

Instruction

- The subroutine specified by the pointer must be written after the FEND instruction;
- When the number of pointer P is used by CALL instruction, CJ designation cannot specify the same number;
- When only using CALL instruction, the subroutine with the same pointer number can be called unlimited times;
- The CALL instruction in the subroutine can be used up to 4 times, and overall, up to 5 levels of nesting are allowed.

6.3 SRET/ Subroutine return

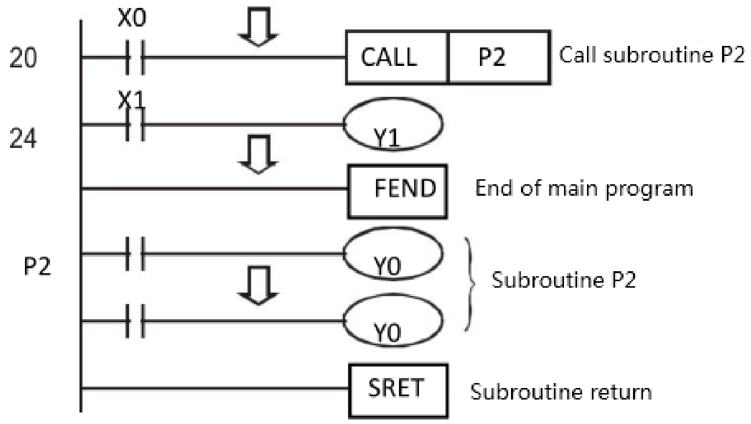
An instruction to return from a subroutine to the main program.

Instruction		Operand type	Function			
FNC 02 SRET		Null	Independent instructions	Mnemonic	Execution condition	
			1step	SRET	Continuous execution	No independent instructions to drive the contacts are required.

Instuctions

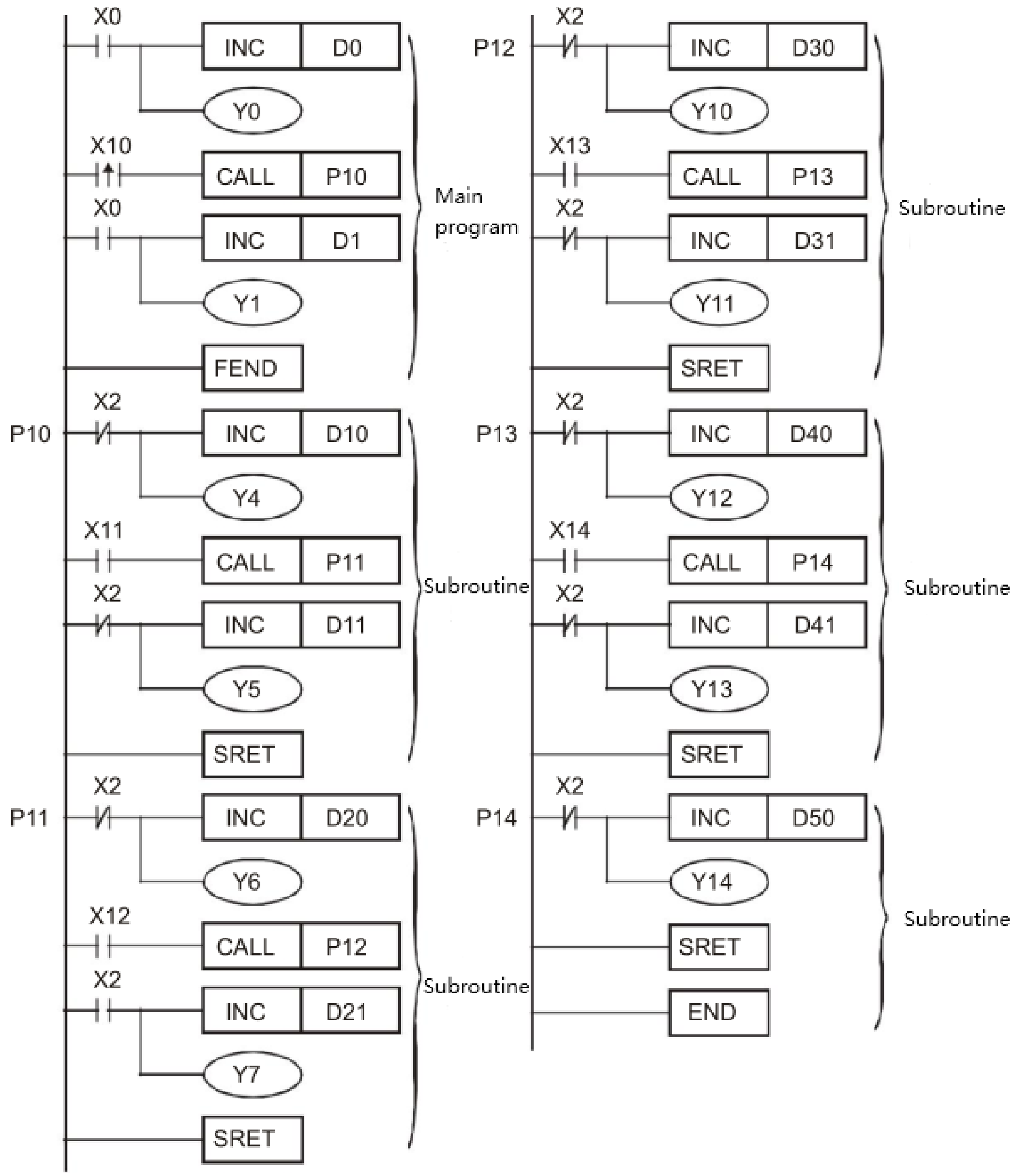
◆ After executing the CALL instruction in the main program, jump to the subroutine, and then use the SRET instruction to return to the main program.

Sample program (1)



◆ When X0=ON, execute the CALL instruction and jump to the subroutine designated by P2. When the SRET instruction is executed, it will return to address 24 and continue to execute.

Sample program (2)



- When X10 is triggered by the rising edge of OFF → ON, the CALL P10 instruction is executed, and the subroutine is executed at P10.
- When X10 is triggered by the rising edge of OFF → ON, the CALL P10 instruction is executed, and the subroutine is executed at P10.
- When X11=ON, CALL P11 will be executed, and it will shift to P11 to execute the designated subroutine.
- When X12 = ON, CALL P12 will be executed, and it will shift to P12 to execute the designated subroutine.
- When X13=ON, CALL P13 is executed and transfer to P13 to execute the designated subroutine.
- When X14=ON, CALL P14 will be executed, and it will be transferred to P14 to execute the designated subroutine. When the SRET instruction is executed, it will return to the previous Pn subroutine and continue to execute.
- After executing the SRET instruction in the P10 subroutine, return to the main program.

6.4 IRET/ Interrupt Return

An instruction to return to the main program from an interrupted subroutine.

Instruction	Operand type	Function			
FNC 03 IRET	Null	Independent Inst.	Mnemonic	Operation Condition	
		1step	IRET	Continuous Operation	This instruction is the independent type, and does not require drive contact.

6.5 EI/ Enable Interrupt

The programmable controller is usually in a state where interrupts are prohibited. Using this instruction, the programmable controller can be changed to a state that allows interrupts, such as input interrupts, timer interrupts, and counter interrupts.

Instruction	Operand type	Function			
FNC 04 EI	Null	Independent Inst.	Mnemonic	Operation Condition	
		1step	EI	Continuous Operation	No independent instructions to drive the contacts are required.

6.6 DI/ Disable Interrupt

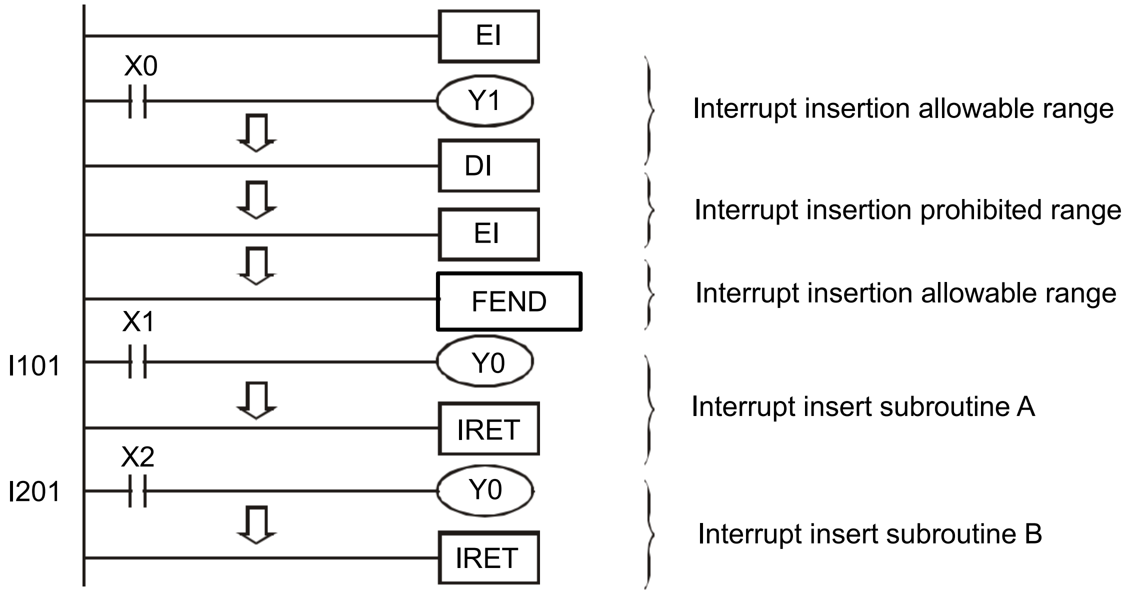
After changing to enable interrupt, use EI (FNC 04) instruction to change to disable interrupt again. Note that the interrupt (required) generated after the DI instruction can only be processed after the EI (FNC 04) instruction is executed.

Instruction	Operand type	Function			
FNC 05 DI	Null	Independent Inst.	Mnemonic	Operation Condition	
		1step	DI	Continuous Operation	No independent instructions to drive the contacts are required.

Instruction

Interrupt subroutines are allowed between the EI instruction and the DI instruction in the program. If there is no interruption in the forbidden section in the program, the DI instruction may not be used.

Sample program



- When the PLC is executing, when the program scans between the EI instruction and the DI instruction, X1=ON or X2=ON, the interrupt insertion subroutine A or B will be executed, and when the subroutine is executed to IRET, it will return to the main program and continue Execute down.

6.7 FEND/ Main Routine Program End

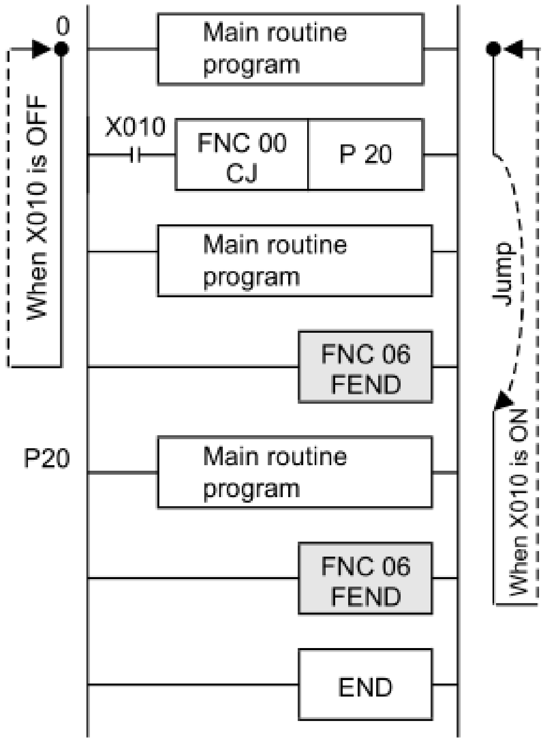
Instruction	Operand type	Function			
		Independent Inst.	Mnemonic	Operation Condition	
FNC 06 FEND	Null	1step	FEND	Continuous Operation	This instruction is the independent type, and does not require drive contact.

Instruction

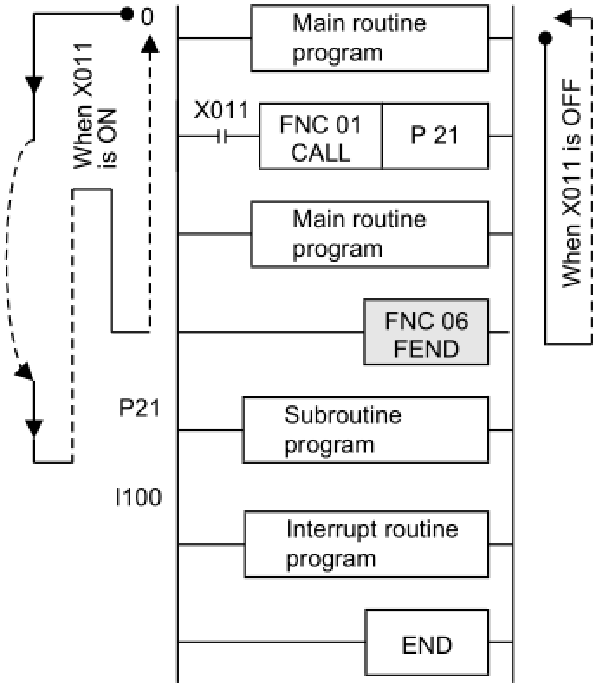
- After the FEND instruction is executed, the same output processing, input processing, and watchdog timer refresh as the END instruction will be executed, and then the program will return to step 0. This instruction is needed when writing subroutines and interrupt programs.
- The CALL instruction program must be written after the FEND instruction, and the SRET instruction must be added at the end of the subroutine; the interrupt program must also be written after the FEND, and the IRET instruction must be added at the end of the program.
- **When FEND instruction is written multiple times:** Please design the subroutine and interrupt program between the last FEND and END instructions.
- **After the CALL instruction is executed:** Before executing the SRET instruction, if the FEND instruction is executed, a program error will occur.
- **After the FOR instruction is executed:** Before executing the NEXT instruction, if the FEND instruction is executed, a program error will occur.

Sample program

1. **In the case of CJ instruction**



2. In the case of CALL instruction



6.8 WDT/Watchdog Timer Refresh

Instruction		Operand type	Function		
FNC 07 WDT	**P**	Null	Independent Inst.	Mnemonic	Operation Condition
			1 step	WDT	Continuous Operation
				WDTP	Pulse (Single) Operation

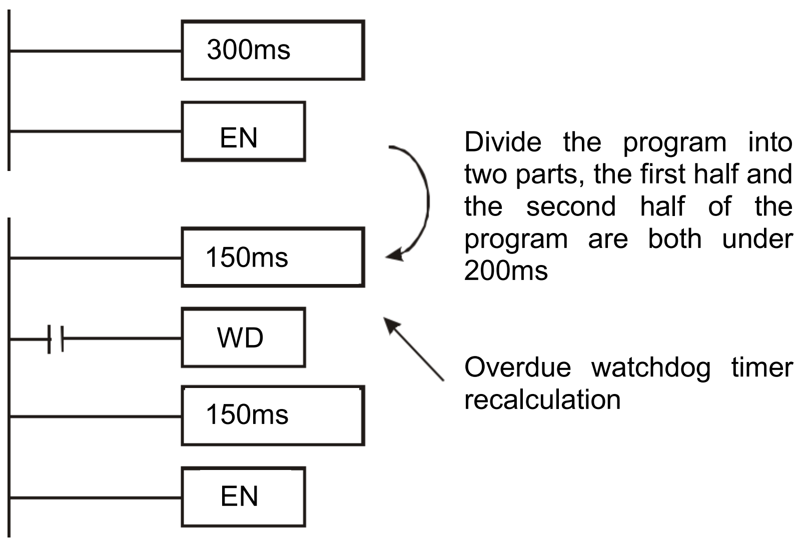
Instructions

- If the operation cycle of the PLC (0 ~ END or FEND instruction execution time) exceeds 200ms, the PLC will have a watchdog timer error (operation abnormality detected), then CPU error, the PLC will stop after the ERROR light is on. In the case of a long operation cycle like this, inserting the WDT instruction in the middle of the program can avoid such errors.
- D8000, watchdog timer time, the maximum can be set to 32767ms (initial value: 200)

As shown in the figure, if the scan cycle of the program is 300ms, at this time, the program can be divided into 2 parts, and the WDT instruction can be placed in the middle. Make the first half and the second half of the program are less than 200ms.

You can also modify the watchdog timer time directly through D8000, such as MOV K300 D8000. When no WDT instruction is programmed, the value of D 8000 becomes valid during END processing.

Sample program



6.9 FOR/Start a FOR/NEXT Loop

The program from the FOR instruction to the NEXT (FNC 09) instruction is repeated for the specified number of times.

Instruction	Operand type	Function						
FNC 08 FOR	S.	16-bit Instruction	Mnemonic	Operation condition		32-bit Instruction	Mnemonic	Operation condition
		3 steps	FOR	Continuous operation			——	——
Operand	S.	The number of repetitions between FOR~NEXT instructions. S.=K1~K32767 (-32768~0 is treated as 1) Target devices: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modification						

6.10 NEXT/ End a FOR/NEXT Loop

Instruction	Operand type	Function			
FNC 09 NEXT	Null	Independent Inst.	Mnemonic	Operation Condition	
		1 step	NEXT	Continuous Operation	This instruction is the independent type, and does not require drive contact.

Instructions

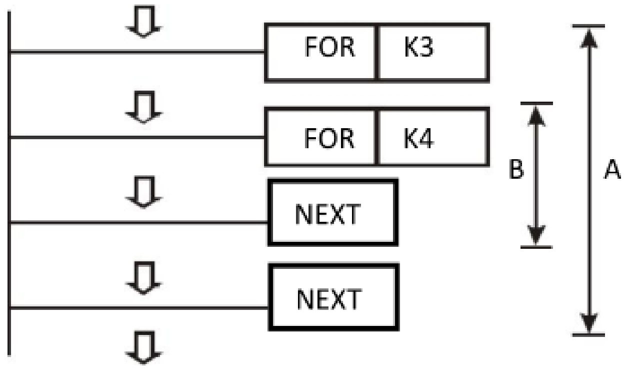
The processing of FOR to NEXT instructions is repeated n times (the number of times specified in the source data). After repeating the specified number of times, execute the step processing after the NEXT instruction.

Notes

1. Between FOR-NEXT instructions, when FOR-NEXT instructions are nested programming, up to 5 levels are allowed.
2. When the number of repetitions is set larger, the cycle time (operation cycle) (D8010) will become larger, and a watchdog timer error will occur. It is necessary to change the time of the watchdog timer or execute the watchdog timer refresh.
3. The situation where the error occurred:
   - ✧ The NEXT instruction comes before the FOR instruction.
   - ✧ There is FOR instruction but no NEXT instruction.
   - ✧ The number of FOR instruction and NEXT instruction is inconsistent.
   - ✧ FEND instruction or END instruction followed by NEXT instruction.

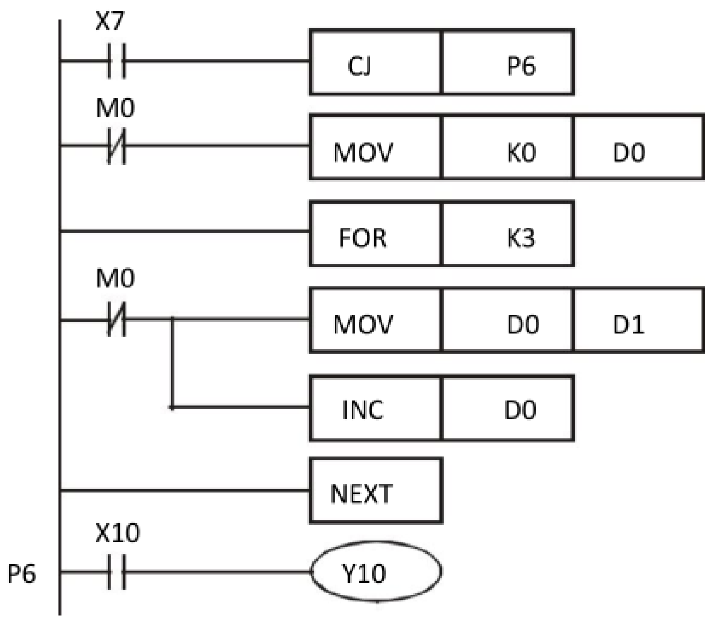
Sample program (1)

After the A program is executed 3 times, the program after the NEXT instruction continues to execute. And program A will execute every time program B is executed four times, that is, program B is executed 3*4=12 times.



Sample program (2)

When X7 = OFF, PLC will execute the program between FOR ~ NEXT;
When X7=ON, the execution of CJ instruction jumps to P6, the program between FOR ~ NEXT is skipped and not executed.



7 Move and Compare

FNC NO.	Mnemonic	Function	Support models		
			3G series PLC	2N series PLC	MX2N series PLC
10	CMP	Compare	★	★	★
11	ZCP	Zone Compare	★	★	★
12	MOV	Move	★	★	★
13	SMOV	Shift Move	★		★
14	CML	Complement	★	★	★
15	BMOV	Block Move	★	★	★
16	FMOV	Fill Move	★	★	★
17	XCH	Exchange	★	★	★
18	BCD	Conversion to Binary Coded Decimal	★	★	★
19	BIN	Conversion to Binary	★	★	★

7.1 CMP/Compare

This instruction compares two values, and outputs the result (smaller, equal or larger) to bit devices (3 points).

Instruction	Operand Type	Function						
D FNC 10 CMP P	S1. S2. D.	16-bit Instruction	Mnemonic	Operation Condition		32-bit Instruction	Mnemonic	Operation Condition
		7 steps	CMP	Continuous Operation		13 steps	DCMP	Continuous Operation
			CMPP	Pulse (Single) Operation			DCMPP	Pulse (Single) Operation

Operand			
Operand	**S1.**	Data or device number handled as comparison value **Applicable devices**: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, Modify	BIN16/32 bit
	S2.	Date or device number handled as comparison source **Applicable devices**: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, Modify	BIN16/32 bit
	D.	Head bit device number to which comparison result is output **Applicable devices**: Y, M, S, Modify	Bit

Explanation

1. **16-bit operation (CMP, CMPP)**

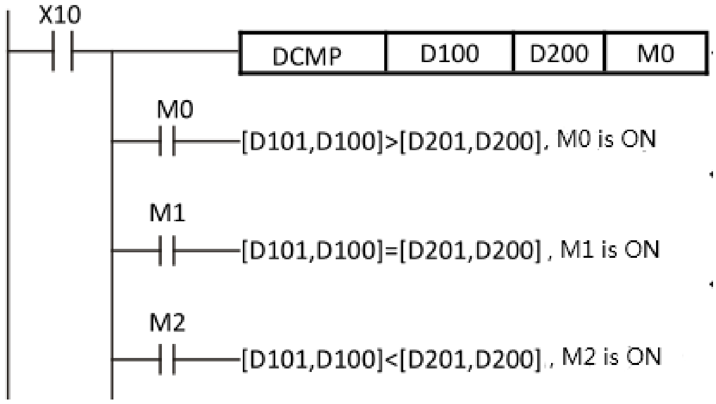
   The comparison value **S1.** and the comparison source **S2.** are compared with each other. According to the result (smaller, equal or larger), either one among **D.**, **D.+1** and **D.+2** turns ON.

2. **32-bit operation (DCMP, DCMPP)**

   The comparison value [**S1.+1, S1**] and the comparison source [**S2+1, S2.**] are compared with each other. According to the result (smaller, equal or larger), either one among **D.**, **D.+1** and **D.+2** turns ON.

- The source data **S1.**, **S2.** or [**S1.+1**, **S1.**] [**S2.+1**, **S2.**] are handled as binary values.
- Comparison is executed algebraically. Example: -10 < 2, -125400 < 22466
- From the device specified as , three devices are occupied.

Program example



- Even if X10=OFF, the CMP instruction is not executed, M0~M2 remain in the state before X10=OFF.
- To clear the comparison result, please use RST or ZRST instruction.
- If you need to get ⩾, ⩽, ≠, you can get M0~M2 in series and parallel.

7.2 ZCP/ Zone Compare

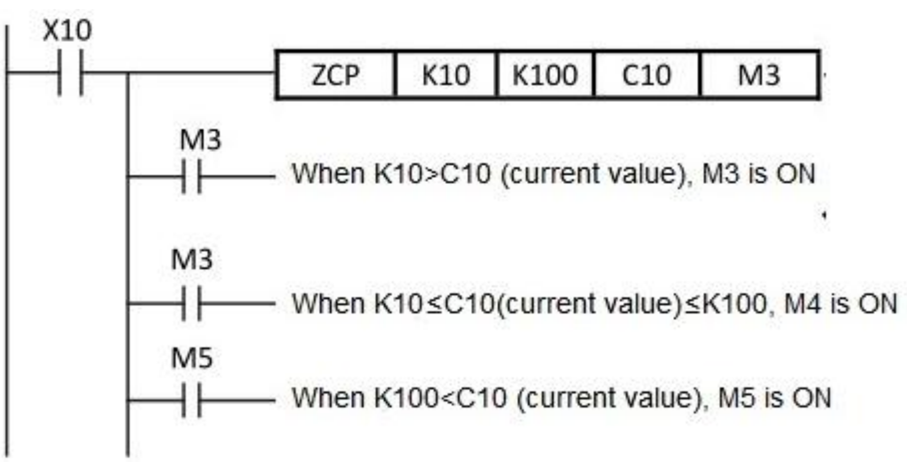
This instruction compares two values (zone) with the comparison source, and outputs the result (smaller, equal or larger) to bit devices (3 points).

<table>
<tr><th>Instruction</th><th>Operand Type</th><th colspan="7">Function</th></tr>
<tr><td rowspan="3">D FNC 11 ZCP P</td><td rowspan="3">S1.
S2.
S.
D.</td><td>16-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td><td rowspan="3"></td><td>32-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td></tr>
<tr><td rowspan="2">9 steps</td><td>ZCP</td><td>Continuous Operation</td><td rowspan="2">17 steps</td><td>DZCP</td><td>Continuous Operation</td></tr>
<tr><td>ZCPP</td><td>Pulse (Single) Operation</td><td>DZCPP</td><td>Pulse (Single) Operation</td></tr>
<tr><td rowspan="4">Operand</td><td>S1.</td><td colspan="6">Data or device number handled as lower comparison value
Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, Modify</td><td>BIN16/32 bit</td></tr>
<tr><td>S2.</td><td colspan="6">Data or device number handled as upper comparison value
Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, Modify</td><td>BIN16/32 bit</td></tr>
<tr><td>S.</td><td colspan="6">Data or device number handled as comparison source
Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, Modify</td><td>BIN16/32 bit</td></tr>
<tr><td>D.</td><td colspan="6">Head bit device number to which comparison result is output
Applicable devices: Y, M, S, Modify</td><td>Bit</td></tr>
</table>

<table>
<tr><td>Explanation</td><td>1. 16-bit operation (ZCP, ZCPP)
The lower comparison value S1. and upper comparison value S2. are compared with the</td></tr>
</table>

	comparison source **S.**. According to the result (smaller, within zone or larger), either one among **D.**, **D.+1** and **D.+2** turns ON. **2. 32-bit operation (DZCP, DZCPP)** The lower comparison value **[S1.+1, S1.]** and upper comparison value **[S2.+1, S2.]** are compared with the comparison source **[S.+1, S.]**. According to the result (smaller, within zone or larger), either one among **D,**, **D.+1** and **D.+2** turns ON. ● Comparison is executed algebraically. Example: −10 < 2 <10 ● When the lower limit value **S1.**>the upper limit value **S2.**, the lower limit value **S1.** is instructed as the upper and lower limit values for comparison. ● Start with the device specified in **D.** and occupy 3 points.

Sample program



◆ Even if X10=OFF, the ZCP instruction is not executed, M0~M2 remain in the state before X10=OFF.

◆ To clear the comparison result, please use RST or ZRST instruction.

7.3 MOV/Move

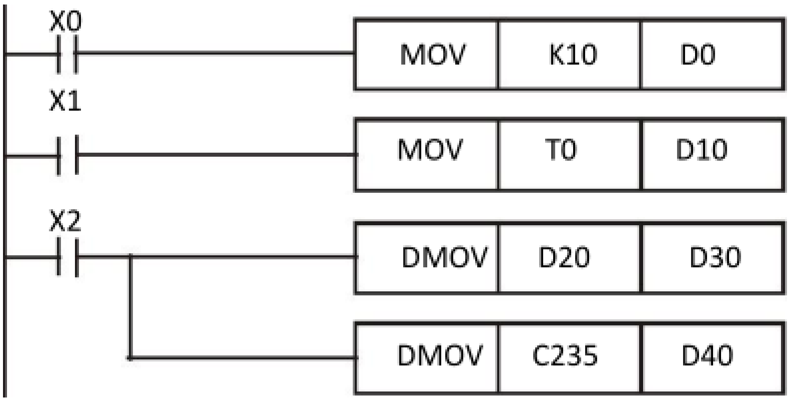
This instruction transfers (copies) the contents of a device to another device.

Instruction	Operand Type	Function					
		16-bit Instruction	**Mnemonic**	**Operation Condition**	**32-bit Instruction**	**Mnemonic**	**Operation Condition**
D **FNC 12 MOV** P	**S.** **D.**	5 steps	MOV	Continuous Operation	9 steps	DMOV	Continuous Operation
			MOVP	Pulse (Single) Operation		DMOVP	Pulse (Single) Operation

Operand number			
Operand number	**S.**	Transfer source data or device number of storing data Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, Modify	BIN16/32bit
	D.	Transfer destination device number Applicable devices: KnY, KnM, KnS, T, C, D, R, V, Z, K, H, Modify	BIN16/32bit

Instruction Explanation	**1. 16-bit operation (MOV and MOVP)** The contents of the transfer source **S.** are transferred to the transfer destination **D.** . When specifying digits of a bit device (K1X000 →K1Y000),The bit device transfers a maximum of 16 points(multiple of 4). ; For example: MOV K4X000 K4Y000(Move X0~X17 State to Y0~Y17) **2. 32-bit operation (DMOV and DMOVP)** The contents of the transfer source**[S.+1**,**S.]** are transferred to the transfer destination **[D.+1,D.]**.（word device transfers 2 point.) Make sure to use DMOV instruction for transferring the operation result of an applied instruction (such as MUL) whose operation result is output in 32 bits, and for transferring a 32-bit numeric value or transferring the current value of a high speed counter (C235 to C255) which is a 32-bit device. When specifying digits of a bit device (K8X000 → K8Y000),The bit device transfers a maximum of 32 points (multiple of 4) ; For example: MOV K8X000 K8Y000(Move X0~X37 State to Y0~Y37) ● While the command input is OFF, the transfer destination **D.** does not change. ● When a constant **K** is specified as the transfer source **S**., it is automatically converted into binary.

Program example



◆ **When transferring 16-bit data,need to use MOV instruction.**

1. when X0=OFF, D10 content no change; when X0=ON, transfer K10 value to D10 data register.
2. when X1=OFF, D10 content no change; whenX1=ON, transfer T0 current value to D10 data register.

◆ **When transferring 32-bit data,need to use DMOV instruction.**

when X2=OFF，(D31,D30),(D41,D40) content no change; when X2=ON, transfer (D21,D20) current value to (D31,D30) data register. Meantime,transfer C235current value to (D41,D40) data register.

7.4 SMOV/Shift Move

This instruction distributes and composes data in units of digit (4 bits).

Instruction		Operand Type	Function						
			16-bit Instruction	**Mnemonic**	**Operation Condition**		**32-bit Instruction**	**Mnemonic**	**Operation Condition**
FNC 13 SMOV		**S. m1 m2 D. n**	11 steps	SMOV	Continuous Operation			——	
	P			SMOVP	Pulse (Single) Operation			——	

Operand number	**S.**	Word device number storing data whose digits will be moved Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, Modify	BIN16 bit
	m1	Head digit position to be moved Applicable devices: K, H	BIN16 bit
	m2	Number of digits to be moved Applicable devices: K, H	BIN16 bit
	D.	Word device number storing data whose digits are moved Applicable devices: KnY, KnM, KnS, T, C, D, R, V, Z, Modify	BIN16 bit
	n	Head digit position of movement destination Applicable devices: K, H	BIN16 bit

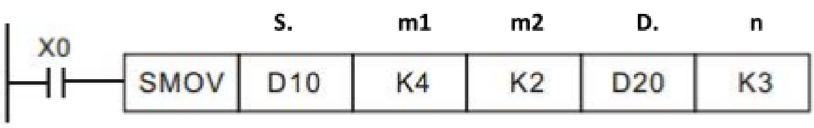
Instruction Explanation

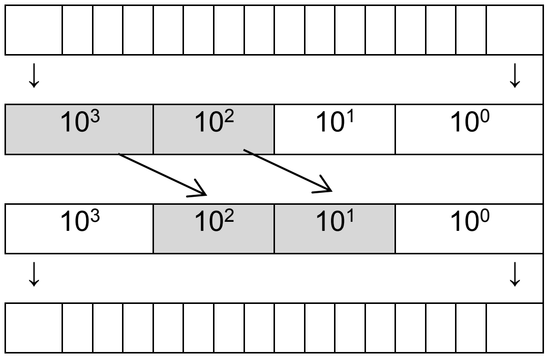
16-bit operation (SMOV, SMOVP)

The contents of the transfer source **S.** and transfer destination **D.** are converted into 4-digit BCD (0000 to 9999) respectively. "m2" digits starting from "m1"th digit are transferred (composed) to the transfer destination **D.** starting from "n"th digit, converted into binary, and then stored to the transfer destination **D.**.

- While the command input is OFF, the transfer destination **D.** does not change.
- When the command input turns ON,The transfer source **S.** and unspecified digits in the transfer destination **D.** do not change.

Program examples 1

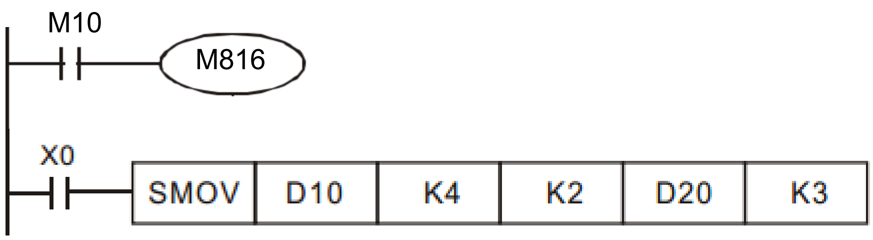


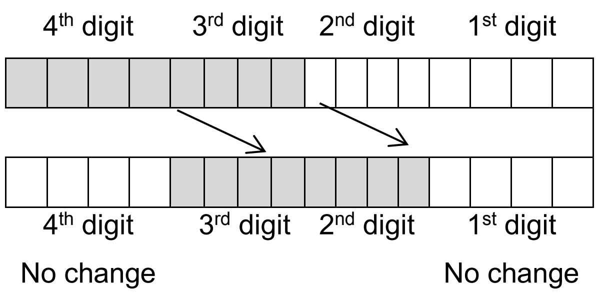


D10(BIN16-bit)
↓Auto convert
D10(BCD 4-digital)
↓Digital move
D20(BCD 4-digital)
↓Auto convert
D20(BIN16-bit)

- When M8168=OFF (BCD mode), X0=ON, Specify the 4th digit of the decimal value of D10 (i.e., thousands) from the lower 2nd digits to the 3rd digit of the decimal value of D20 (i.e., hundreds) from the lower 2nd 2 digits. The contents of 10^3 and 10^0 of D20 have not changed after this instruction is executed.
- Before execute, if D10=K1234, D20=K5678, After Being executed, D10 does not change, D20=K5128.

Program examples





D10(BIN16-bit)
↓Digital move
D20(BIN16-bit)

- When M8168=ON (BIN mode), if SMOV instruction is executed, D10,D20 will not be executed as BCD conversion, but is moved in units of 4 bits.
- Before execute , if D10=H1234, D20=H5678, After Being executed, D10 does not change, D20=H5128.

7.5 CML/Complement

This instruction inverts data in units of bit, and then transfers (copies) the inverted data.

Instruction	Operand Type	Function					
		16-bit Instruction	**Mnemonic**	**Operation Condition**	**32-bit Instruction**	**Mnemonic**	**Operation Condition**
D **FNC 14 CML** P	**S.** **D.**	5steps	CML	Continuous Operation	9steps	DCML	Continuous Operation
			CMLP	Pulse (Single) Operation		DCMLP	Pulse (Single) Operation

Operand number			
Operand number	**S.**	Data to be inverted or word device number storing data Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify	BIN16/32-bit
	D.	Word device number storing inverted data Applicable devices: KnY, KnM, KnS, T, C, D, R, V, Z, modify	BIN16/32-bit

Instruction Explanation

1. 16-bit operation (CML,CMLP)

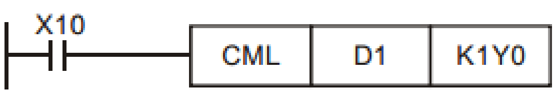
Each bit of a device specified as **S.** is inverted （0→1,1→0), and then transferred to **D.**.

2. 32-bit operation (DCML,DCMLP)

Each bit of devices specified as **[S.+1**,**S.]** is inverted （0→1,1→0), and then transferred to **[D.+1,D.]**.

- This operation is useful when a logically inverted output is required as an output from a PLC.
- When a constant K is specified as transfer source **S.**, it is automatically converted into BIN.

Program example 1

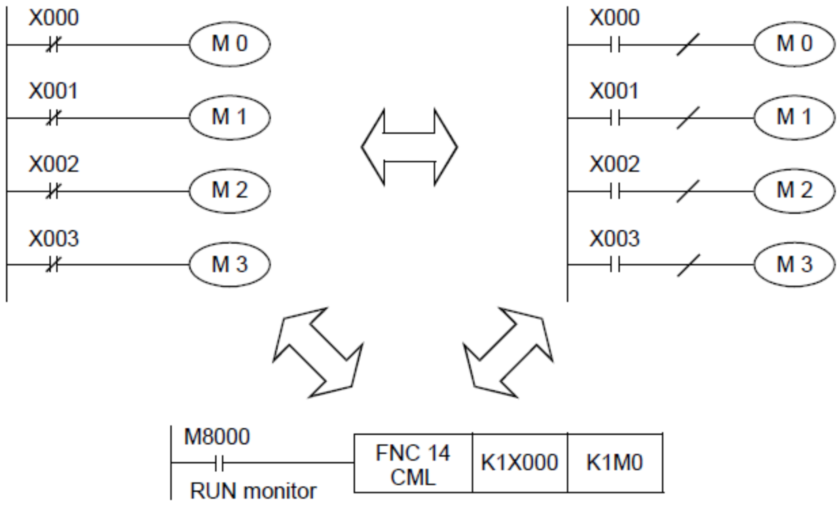


	b15												b3	b2	b1	b0
D1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
K1Y0													0	1	0	1
													Y3	Y2	Y1	Y0

←Inverted data is transferred→

◆ When X10=ON, Invert b0~b3 content of D1 and transfer to Y0~Y3.

Program example 2



- Can use CML instruction to easy corresponding ladder.

7.6 BMOV/Block Move

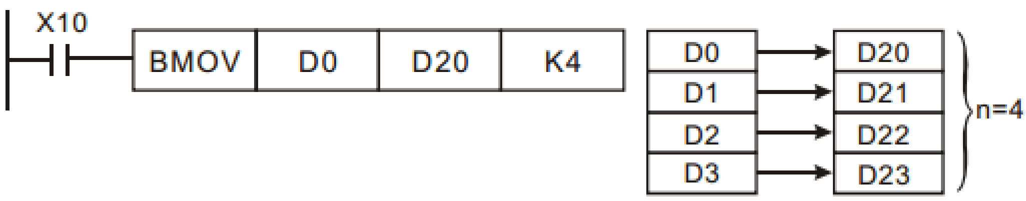
This instruction transfers (copies) a specified number of data at one time.

Instruction		Operand Type	Function						
FNC 15 BMOV		S. D. n	**16-bit Instruction**	**Mnemonic**	**Operation Condition**		**32-bit Instruction**	**Mnemonic**	**Operation Condition**
			7steps	BMOV	Continuous Operation			——	
	P			BMOVP	Pulse (Single) Operation			——	

Operand number			
Operand number	**S.**	Transfer source data or device number storing data Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, modify	BIN16-bit
	D.	Transfer destination device number Applicable devices: KnY, KnM, KnS, T, C, D, R, modify	BIN16-bit
	n	Number of transferred points (including file registers) [n ≤ 512] Applicable devices: D, K, H	BIN16-bit

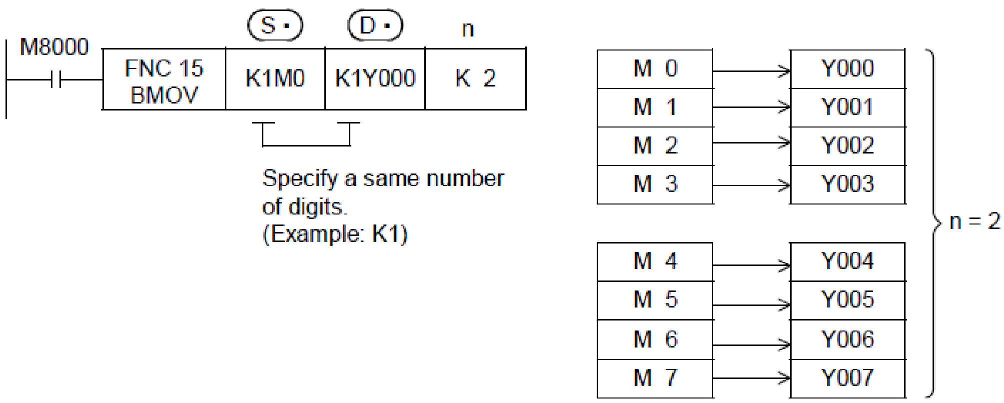
Instruction Explanation	Transfers "n" points of data from **S.** to "n" points of **D.** at one time. ● If the device number range is exceeded, data is transferred within the possible range.

Program example 1



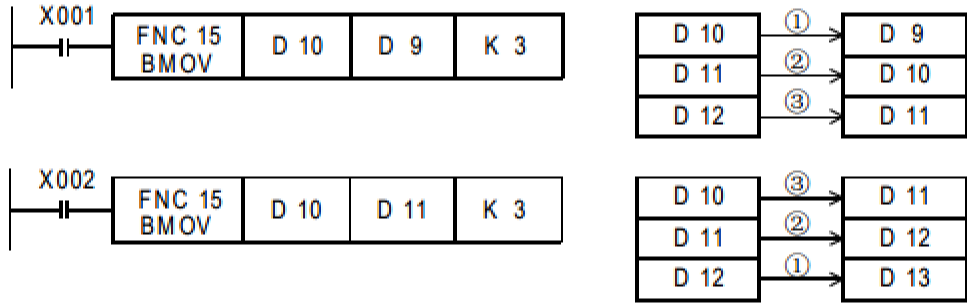
◆ When X10=ON, data of 4 registers of D0~D3 are transferred to 4 registers of D20~D23.

Program example 2



◆ When specifying digits of bit devices KnX, KnY, KnM, KnS to transfer, **S.** and **D.** digital numbers should be same, Namely **n** number should be same.

Program example 3



◆ When two operation specif transferred numbers overlap, transfer in the order of ①→②→③.

7.7 FMOV/Fill Move

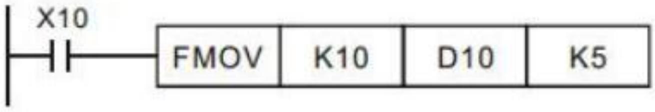
This instruction transfers same data to specified number of devices.

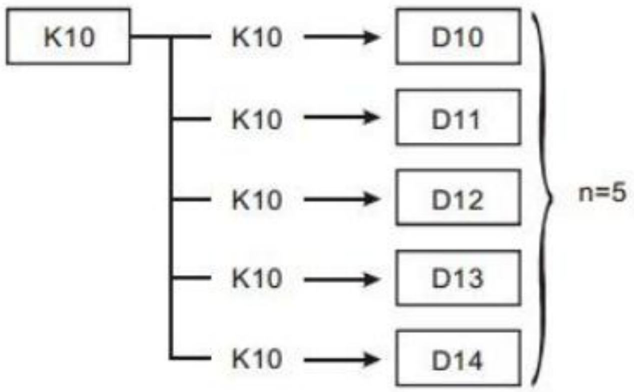
Instruction	Operand Type	Function					
D FNC 16 FMOV P	S. D. n	**16-bit Instruction**	**Mnemonic**	**Operation Condition**	**32-bit Instruction**	**Mnemonic**	**Operation Condition**
		7steps	FMOV	Continuous Operation	13steps	DFMOV	Continuous Operation
			FMOVP	Pulse (Single) Operation		DFMOVP	Pulse (Single) Operation
Operand number	**S.**	Transfer source data or device number storing data Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify					BIN16/32-bit
	D.	Head word device number of transfer destination (Same data is transferred from the transfer source at one time.) Applicable devices: KnY, KnM, KnS, T, C, D, R, V, Z, modify					BIN16/32-bit
	n	Number of transfer points [K1 ≤ n ≤ K512, H1 ≤ n ≤ H1FF] Applicable devices: K, H					BIN16-bit

Instruction Explanation	
	1. 16-bit operation (FMOV,FMOVP) The contents of **S.** are transferred to "n" devices starting from **D.**. **2. 32-bit operation (DFMOV,DFMOVP)** The contents of **[S.+1**,**S.]** are transferred to "n" 32-bit devices starting from **[D.+1,D.]**. ● The contents will be the same among all of "n" 32-bit devices. ● If the number of points specified by "n" exceeds the device number range, data is transferred within the possible range. ● While the command input is OFF, the transfer destination **D.** does not change. ● While the command input is ON, the data of the transfer source **S.** does not change.

	● When a constant (K) is specified as the transfer source [+1,], it is automatically converted into binary.

Program example





◆ When X10=ON, K10 is transferred to constant 5 registers which start from D10.

7.8 XCH/Exchange

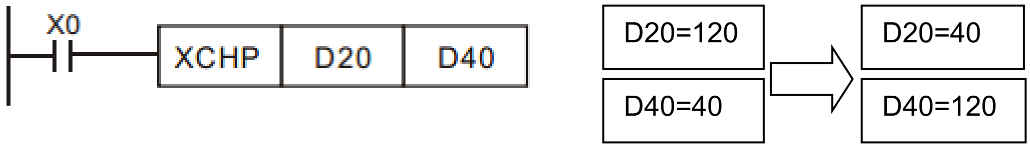
This instruction exchanges data between two devices.

<table>
<tr><th colspan="3">Instruction</th><th>Operand Type</th><th colspan="7">Function</th></tr>
<tr><td rowspan="2">D</td><td rowspan="2">FNC 17
XCH</td><td rowspan="2">P</td><td rowspan="2">D1.
D2.</td><td>16-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td><td rowspan="2"></td><td>32-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td></tr>
<tr><td>5 steps</td><td>XCH
XCHP</td><td>Continuous Operation
Pulse (Single) Operation</td><td>9 steps</td><td>DXCH
DXCHP</td><td>Continuous Operation
Pulse (Single) Operation</td></tr>
<tr><td colspan="3" rowspan="2">Operand number</td><td>D1.</td><td colspan="6">Store device number of exchanged data
Applicable devices: KnY, KnM, KnS, T, C, D, R, V, Z, modify</td><td>BIN16/32-bit</td></tr>
<tr><td>D2.</td><td colspan="6">Store device number of exchanged data
Applicable devices: KnY, KnM, KnS, T, C, D, R, V, Z, modify</td><td>BIN16/32-bit</td></tr>
</table>

Instruction Explanation	**1. 16-bit operation (XCH,XCHP)** Data is exchanged between **D1.** and **D2.**. **2. 32-bit operation (DXCH,DXCHP)** Data is exchanged between **[D1.+1,D1.]** and **[D2.+1,D2.]**.

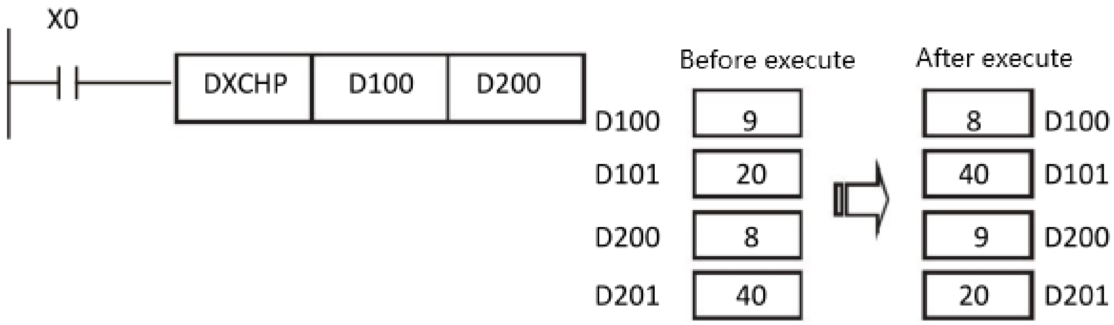
Program example 1

X0=OFF→ON,Data is exchanged between D20 and D40. Before Execute After Execute

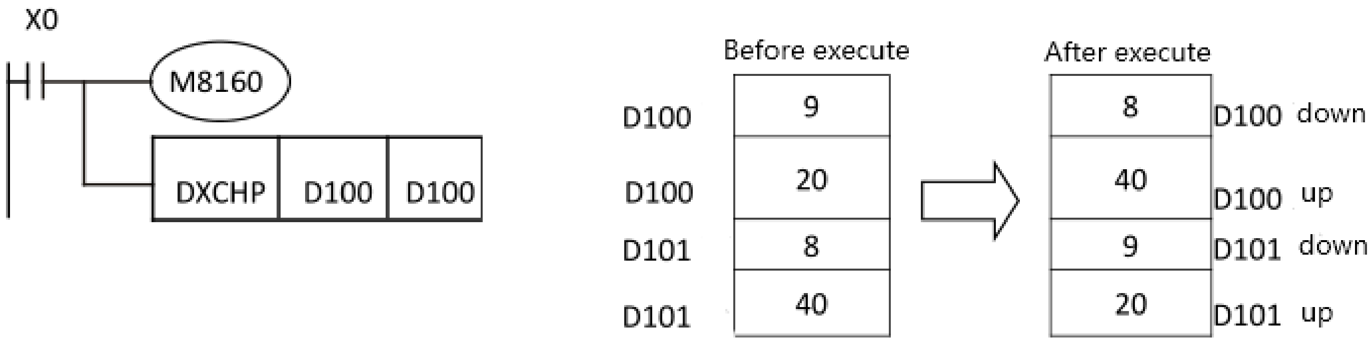


Program example 2

X0=OFF→ON, Data is exchanged between D100 and D200.



Extend function



- When the instruction is executed while M8160 is ON, high-order 8 bits (byte) and low-order 8 bits (byte) of a word device are exchanged each other. [while device number of **D1. and D2.** should be same]
- this instruction works in the same way as SWAP instruction
- In a 32-bit operation, high-order 8 bits (byte) and low-order 8 bits (byte) of each word device are exchanged for each other.

7.9 BCD/Conversion to Binary Coded Decimal

This instruction converts binary (BIN) data into binary-coded decimal (BCD) data.

Binary data is used in operations in PLCs. Use this instruction to display numeric values on the seven segment display unit equipped with BCD decoder.

<table>
<tr><th>Instruction</th><th>Operand Type</th><th colspan="7">Function</th></tr>
<tr><td rowspan="3">D FNC 18 BCD P</td><td rowspan="3">S.
D.</td><td>16-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td><td rowspan="3"></td><td>32-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td></tr>
<tr><td rowspan="2">5 steps</td><td>BCD</td><td>Continuous Operation</td><td rowspan="2">9 steps</td><td>DBCD</td><td>Continuous Operation</td></tr>
<tr><td>BCDP</td><td>Pulse (Single) Operation</td><td>DBCDP</td><td>Pulse (Single) Operation</td></tr>
<tr><td rowspan="2">Operand number</td><td>S.</td><td colspan="6">Store Word device number of the conversion source (binary) data
Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, modify</td><td>BIN16/32-bit</td></tr>
<tr><td>D.</td><td colspan="6">Store Word device number of the conversion destination (binary-coded decimal) data
Applicable devices: KnY, KnM, KnS, T, C, D, R, V, Z, modify</td><td>BIN16/32-bit</td></tr>
</table>

Instruction Explanation

1. 16-bit operation (BCD, BCDP)

This instruction converts the binary (BIN) data of **S.** into BCD(binary-coded decimal) data, and transfers the converted BCD data to **D.**.

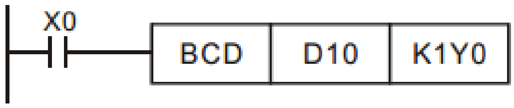
2. 32-bit operation (DBCD, DBCDP)

This instruction converts the binary (BIN) data of **[S.+1,S.]** into BCD (binary-coded decimal) data, and transfers the converted BCD data to **[D.+1,D.]**.

- In **16-bit operation,**The data of **S.** can be converted into BCD (K0 to K9999)
  In **32-bit operation,**The data of **[S.+1,S.]** can be converted into BCD (K0 to K99999999).
- The Arithmetic and Logical Operation of the PLC, and the INC and DEC instructions are all executed in BIN mode. So in terms of application, when you want to see a display with a decimal value, you can use BCD conversion to convert the BIN value into a BCD value output.
- Operand number specify bits,please check below table: (16-bit max is K4Y0, 32-bit max is K8Y0)

[D•+1, D•]	Number of digits	Data range
K1Y000	1	0 to 9
K2Y000	2	00 to 99
K3Y000	3	000 to 999
K4Y000	4	0000 to 9999
K5Y000	5	00000 to 99999
K6Y000	6	000000 to 999999
K7Y000	7	0000000 to 9999999
K8Y000	8	00000000 to 99999999

Program example



- When X0=ON, BIN value of D10 is converted to BCD value, Store the single digit of the result in K1Y0 (Y0 ~ Y3) four bit devices .
- If D10=001E (Hex)=0030(Decimal), Then execute result Y0~Y3=0000(BIN).

7.10 BIN/Conversion to Binary

This instruction converts binary-coded decimal (BCD) data into binary (BIN) data.
Use this instruction to convert a binary-coded decimal (BCD) value such as a value set by a digital switch into binary (BIN) data and to receive the converted binary data so that the data can be handled in operations in PLCs.

<table>
<tr><th colspan="3">Instruction</th><th>Operand Type</th><th colspan="8">Function</th></tr>
<tr><td rowspan="3">D</td><td rowspan="3">FNC 19
BIN</td><td rowspan="3">P</td><td rowspan="3">S.
D.</td><td>16-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td><td rowspan="3"></td><td>32-bit Instruction</td><td>Mnemonic</td><td colspan="2">Operation Condition</td></tr>
<tr><td rowspan="2">5 steps</td><td>BIN</td><td>Continuous Operation</td><td rowspan="2">9 steps</td><td>DBIN</td><td colspan="2">Continuous Operation</td></tr>
<tr><td>BINP</td><td>Pulse (Single) Operation</td><td>DBINP</td><td colspan="2">Pulse (Single) Operation</td></tr>
<tr><td colspan="3" rowspan="2">Operand number</td><td>S.</td><td colspan="7">Word device number storing the conversion source (binary-coded decimal) data
Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, Modify</td><td>BIN16/32-bit</td></tr>
<tr><td>D.</td><td colspan="7">Word device number of the conversion destination (binary)
Applicable devices: KnY, KnM, KnS, T, C, D, R, V, Z, Modify</td><td>BIN16/32-bit</td></tr>
</table>

Instruction Explanation

1. 16-bit operation (BIN,BINP)

This instruction converts the binary-coded decimal (BCD) data of **S.** into binary (BIN) data, and transfers the converted binary data to **D.** .

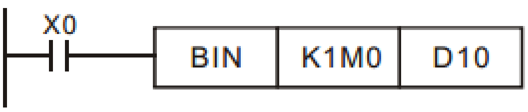
2. 32-bit operation (DBIN,DBINP)

This instruction converts the binary-coded decimal (BCD) data of **[S.+1,S.]** into binary (BIN) data, and transfers the converted binary data to **[D.+1,D.]**.

- Constants K and H will be automatically converted into BIN, so this instruction is not necessary.
- In **16-bit operation,**The data of **S.** can be converted between K0 to K9999 (BCD)
  In **32-bit operation,**The data of **[S.+1,S.]** can be converted between K0 to K99999999 (BCD).
- The Arithmetic and Logical Operation of the PLC, and the INC and DEC instructions are all executed in BIN mode. So in terms of application, when you want to see a display with a decimal value, you can use BCD conversion to convert the BIN value into a BCD value output.
- Operand number specify bits,please check below table: (16-bit max is K4Y0, 32-bit max is K8Y0)

[S·+1, S·]	Number of digits	Data range
K1X000	1	0 to 9
K2X000	2	00 to 99
K3X000	3	000 to 999
K4X000	4	0000 to 9999
K5X000	5	00000 to 99999
K6X000	6	000000 to 999999
K7X000	7	0000000 to 9999999
K8X000	8	00000000 to 99999999

Program Example



◆ When X0=ON, After the BCD value of K1M0 is converted to BIN value, the result is stored in D10.

8 Arithmetic and Logical Operation

FNC NO.	Mnemonic	Function	Supported PLC series		
			3G PLC	2N PLC	MX2N PLC
20	ADD	BIN Addition	★	★	★
21	SUB	BIN Subtraction	★	★	★
22	MUL	BIN Multiplication	★	★	★
23	DIV	BIN Division	★	★	★
24	INC	BIN Increment	★	★	★
25	DEC	BIN Decrement	★	★	★
26	WAND	Logical Word AND	★	★	★
27	WOR	Logical Word OR	★	★	★
28	WXOR	Logical Exclusive OR	★	★	★
29	NEG	Negation	★	★	★

8.1 ADD/BIN Addition

This instruction executes addition by two values to obtain the result (A + B = C).

<table>
<tr><th colspan="3">Instruction</th><th>Operand Type</th><th colspan="7">Function</th></tr>
<tr><td rowspan="3">D</td><td rowspan="3">FNC 20
ADD</td><td rowspan="3">P</td><td rowspan="3">S1.
S2.
D.</td><td>16-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td><td rowspan="3"></td><td>32-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td></tr>
<tr><td>7 steps</td><td>ADD</td><td>Continuous Operation</td><td>13 steps</td><td>DADD</td><td>Continuous Operation</td></tr>
<tr><td></td><td>ADDP</td><td>Pulse (Single) Operation</td><td></td><td>DADDP</td><td>Pulse (Single) Operation</td></tr>
<tr><td colspan="3" rowspan="3">Operand number</td><td>S1.</td><td colspan="6">Data for addition or word device number storing data
Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify</td><td>BIN16/32-bit</td></tr>
<tr><td>S2.</td><td colspan="6">Data for addition or word device number storing data
Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify</td><td>BIN16/32-bit</td></tr>
<tr><td>D.</td><td colspan="6">Word device number storing the addition result
Applicable devices: KnY, KnM, KnS, T, C, D, R, V, Z, modify</td><td>BIN16/32-bit</td></tr>
</table>

Instruction Explanation

1. 16-bit operation (ADD,ADDP)

The contents of **S2.**are added to **S1.** in the binary format, and the addition result is transferred to **D.**.

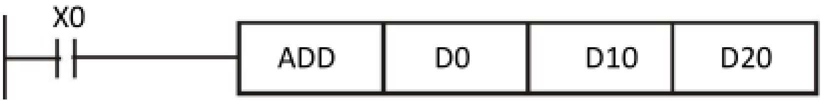
2. **32-bit operation (DADD,DADDP)**

The contents of **[S2.+1,S2.]** are added to **[S1.+1,S1.]** in the binary format, and the addition result is transferred to **[D.+1,D.]**.

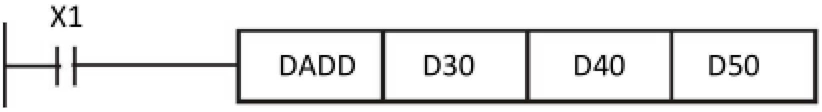
- When a constant (K) is specified in or , it is automatically converted into the binary format.
- The most significant bit of each data indicates the sign (positive: 0 or negative: 1), and data are added algebraically. 5 + (−8) = −3.
- Relationship between the flag operation and the sign (positive or negative) of a numeric value, as below

Device	Name	Description
M8020	Zero	ON : When the operation result is 0 OFF: When the operation result is not 0
M8021	Borrow	ON : When the operation result is less than –32768 (in 16-bit operation) or –2,147,483,648 (in 32-bit operation) OFF: When the operation result is not less than –32768 (in 16-bit operation) or –2,147,483,648 (in 32-bit operation)
M8022	Carry	ON : When the operation result is more than 32767 (in 16-bit operation) or 2,147,483,647 (in 32-bit operation) OFF: When the operation result is not more than 32767 (in 16-bit operation) or 2,147,483,647 (in 32-bit operation)

Program Example



- 16-bit BIN addition: When X0=ON, Result of adding D0 content to D10 content will be restored in D20 content



- 32-bit BIN addition: When X1=ON, Result of adding (D31,D30) content to (D41,D40) content will be restored in (D51、D50)content. (Among them, D30,D40,D50 is the low 16-bit data;D31,D41,D51 are the high 16-bit data)

- Source Operand number and Target Operand number can also be specified as the same device number
- In the above case,when ADD of Continuous Operation is used “ADD: X2=ON”,each scan cycle D100 will add K25 to D100.

8.2 SUB/BIN Subtraction

This instruction executes subtraction using two values to obtain the result (A − B = C).

<table>
<tr><td colspan="3">Instruction</td><td>Operand Type</td><td colspan="7">Function</td></tr>
<tr><td rowspan="3">D</td><td rowspan="3">FNC 21
SUB</td><td rowspan="3">P</td><td rowspan="3">S1.
S2.
D.</td><td>16-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td><td rowspan="3"></td><td>32-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td></tr>
<tr><td>7steps</td><td>SUB</td><td>Continuous Operation</td><td>13steps</td><td>DSUB</td><td>Continuous Operation</td></tr>
<tr><td></td><td>SUBP</td><td>Pulse (Single) Operation</td><td></td><td>DSUBP</td><td>Pulse (Single) Operation</td></tr>
<tr><td colspan="3" rowspan="3">Operand number</td><td>S1.</td><td colspan="6">Data for subtraction or word device number storing data
Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify</td><td>BIN16/32-bit</td></tr>
<tr><td>S2.</td><td colspan="6">Data for subtraction or word device number storing data
Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify</td><td>BIN16/32-bit</td></tr>
<tr><td>D.</td><td colspan="6">Word device number storing the subtraction result
Applicable devices: KnY, KnM, KnS, T, C, D, R, V, Z, modify</td><td>BIN16/32-bit</td></tr>
</table>

Instruction Explanation

1. 16-bit operation (SUB,SUBP)

The contents of **S2.**are subtracted from **S1.** in the binary format, and the subtraction result is transferred to **D.**.

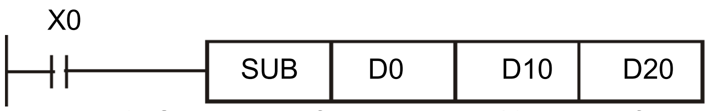
2. **32-bit operation (DSUB,DSUBP)**

The contents of **[S2.+1,S2.]** are subtracted from **[S1.+1,S1.]** in the binary format, and the subtraction result is transferred to **[D.+1,D.]**.

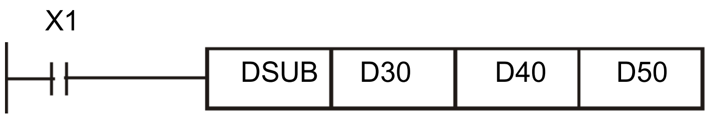
- When a constant (K) is specified in **S1.** and **S2 or [S1.+1,S1.]**and **[S2.+1,S2.]**, it is automatically converted into the binary format.
- The most significant bit of each data indicates the sign (positive: 0 or negative: 1), and data are subtracted algebraically.: 5-(-8)=13。
- Relationship between the flag operation and the sign (positive or negative) of a numeric value:

Device	Name	Description
M8020	Zero	ON : When the operation result is 0 OFF: When the operation result is other than 0
M8021	Borrow	ON : When the operation result is less than −32768 (in 16-bit operation) or −2,147,483,648 (in 32-bit operation) OFF: When the operation result is not less than −32768 (in 16-bit operation) or −2,147,483,648 (in 32-bit operation)
M8022	Carry	ON : When the operation result is more than 32767 (in 16-bit operation) or 2,147,483,647 (in 32-bit operation) OFF: When the operation result is not more than 32767 (in 16-bit operation) or 2,147,483,647 (in 32-bit operation)

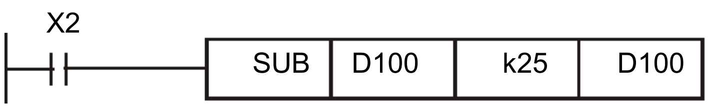
Program Example



- 16-bit BIN Subtraction: When X0=ON, Result of subtracting D10 content from D0 content will be restored in D20 content



- 32-bit BIN Subtraction: When X1=ON, Result of subtracting (D41,D40) content from (D31,D30) content will be restored in (D51、D50)content. (Among them, D30,D40,D50 is the low 16-bit data;D31,D41,D51 are the high 16-bit data)



- Source Operand number and Target Operand number can also be specified as the same device number
- In the above case,when Continuous Operation is used "SUB: X2=ON",each scan cycle D100 will subtract K25 to D100.

8.3 MUL/BIN Multiplication

This instruction executes multiplication by two values to obtain the result (A x B = C).

Instruction	Operand Type	Function					
D FNC 22 MUL P	S1. S2. D.	**16-bit Instruction**	**Mnemonic**	**Operation Condition**	**32-bit Instruction**	**Mnemonic**	**Operation Condition**
		7 steps	MUL	Continuous Operation	13 steps	DMUL	Continuous Operation
			MULP	Pulse (Single) Operation		DMULP	Pulse (Single) Operation

Operand number			
Operand number	**S1.**	Data for multiplication or word device number storing data Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify	BIN16/32-bit
	S2.	Data for multiplication or word device number storing data Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify	BIN16/32-bit
	D.	Head word device number storing the multiplication result Applicable devices: KnY, KnM, KnS, T, C, D, R, V, Z, modify	BIN32/64-bit

Instruction Explanation

1. 16-bit operation (MUL,MULP)

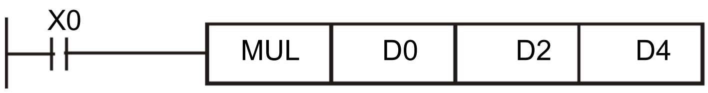
The contents of **S1.**are multiplied by **S2.** in the binary format, and the multiplication result is transferred to 32-bit **[D.+1,D.]**.

2. **32-bit operation (DMUL,DMULP)**

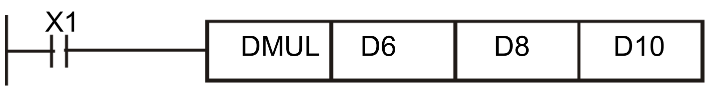
The contents of **[S1.+1, S1.]** are multiplied by**[S2.+1, S2.]** in the binary format, and the multiplication result is transferred to 64-bit **[D.+3, D.+2, D.+1, D.]** (four word devices).

- When a constant (K) is specified in **S1.** and **S2 or [S1.+1,S1.]**and **[S2.+1,S2.]**, it is automatically converted into the binary format.
- The most significant bit of each data indicates the sign (positive: 0 or negative: 1), and data are multiplied algebraically. : 5x(-8)=-40。
- When a digit (K1 to K8) is specified for **[D.+3, D.+2, D.+1, D.],occupy** 2 consecutive 32-bit data.
- In a 32-bit operation (DMUL ,DMULP), Z cannot be specified.

Program Example



- ◆ 16-bit BIN Multiplication: When X0=ON, Result of multiplying D0 and D2 content will be restored in [D5,D4] content



- ◆ 32-bit BIN Multiplication: When X1=ON, Result of multiplying (D7,D6) and (D9,D8) content will be restored in (D13,D12,D11,D10) content.

8.4 DIV/BIN Division

This instruction executes division by two values to obtain the result (A ÷ B = C ...).

<table>
<tr><th colspan="3">Instruction</th><th>Operand Type</th><th colspan="7">Function</th></tr>
<tr><td rowspan="2">D</td><td rowspan="2">FNC 23
DIV</td><td rowspan="2">P</td><td rowspan="2">S1.
S2.
D.</td><td>16-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td><td rowspan="2"></td><td>32-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td></tr>
<tr><td>7 steps</td><td>DIV
DIVP</td><td>Continuous Operation
Pulse (Single) Operation</td><td>13 steps</td><td>DDIV
DDIVP</td><td>Continuous Operation
Pulse (Single) Operation</td></tr>
<tr><td colspan="3" rowspan="3">Operand number</td><td>S1.</td><td colspan="6">Data for division or word device number storing data (dividend)
Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify</td><td>BIN16/32-bit</td></tr>
<tr><td>S2.</td><td colspan="6">Data for division or word device number storing data (divisor)
Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify</td><td>BIN16/32-bit</td></tr>
<tr><td>D.</td><td colspan="6">Head word device number storing the division result (quotient and remainder)
Applicable devices: KnY, KnM, KnS, T, C, D, R, V, Z*1, modify</td><td>BIN32/64-bit</td></tr>
</table>

Remarks: *1, Only supported in FX3U(C)

<table>
<tr><td>Instruction Explanation</td><td>
1. 16-bit operation（DIV,DIVP)

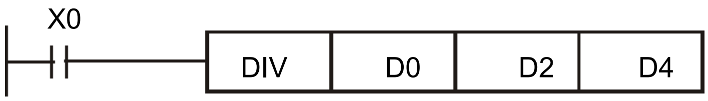
S1. indicates the dividend,S2. indicates the divisor, the quotient is transferred to D., and the remainder is transferred to D.+1.

2. 16-bit operation（DDIV,DDIVP)

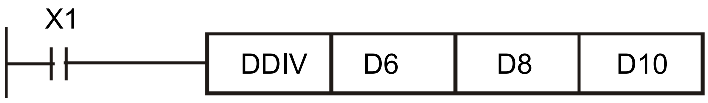
[S1.+1, S1.] indicates the dividend,[S2.+1, S2.] indicates the divisor, the quotient is transferred to [D.+1, D.]), and the remainder is transferred to [D.+3, D.+2]

● When a constant (K) is specified in S1. and S2 or [S1.+1,S1.]and [S2.+1,S2.],it is automatically converted into the binary format.
● The most significant bit of each data indicates the sign (positive: 0 or negative: 1), and data are divided algebraically. 36÷(-5)=-7…1.
● When the divisor is S2.or [S2.+1,S2.]"0", an operation error is caused and the instruction is not executed.
● The remainder is not obtained when a bit device is specified with digit specification.
● In a 32-bit operation （DDIV,DDIVP), Z cannot be specified.
</td></tr>
</table>

Program Example



- 16-bit BIN Division: When X0=ON, D0（dividend)÷D2（divisor)=D4（quotient)…D5（remainder).
- If D0=100, D2=33；Then D4=3, D5=1.



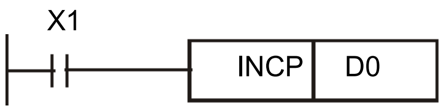
- 32-bit BIN Division: When X0=ON, [D7,D6]（dividend)÷[D9,D8]（divisor)=[D11,D10]（quotient)…[D13,D12] (remainder).
- If [D7,D6]=100000, [D9,D8]=3333；Then [D11,D10]=30, [D13,D12]=10.

8.5 INC/BIN Increment

This instruction increments the data of a specified device by "1".

<table>
<tr><th>Instruction</th><th>Operand Type</th><th colspan="7">Function</th></tr>
<tr><td rowspan="3">D | FNC 24 INC | P</td><td rowspan="3">D.</td><td>16-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td><td rowspan="3"></td><td>32-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td></tr>
<tr><td rowspan="2">3 steps</td><td>INC</td><td>Continuous Operation</td><td rowspan="2">5 steps</td><td>DINC</td><td>Continuous Operation</td></tr>
<tr><td>INCP</td><td>Pulse (Single) Operation</td><td>DINCP</td><td>Pulse (Single) Operation</td></tr>
<tr><td>Operand number</td><td>D.</td><td colspan="6">Word device number storing data to be incremented by "1"
Applicable devices: KnY, KnM, KnS, T, C, D, R, V, Z, modify</td><td>BIN16/32-bit</td></tr>
<tr><td>Instruction Explanation</td><td colspan="8">1. 16-bit operation (INC,INCP)
The contents of D. are incremented by "1", and the increment result is transferred to D..
2. 32-bit operation (DINC,DINCP)
The contents of [D.+1, D.] are incremented by "1", and the increment result is transferred to [D.+1, D.].
● Note that data is incremented in every operation cycle in a continuous operation type instruction.
● 16-bit BIN operation,When "+32767" is incremented by "1", the result is "−32768". Flags (zero, carry and borrow) are not activated at this time.
● 32-bit BIN operation,When "+2,147,483,647" is incremented by "1", the result is "−2,147,483,648". Flags (zero, carry and borrow) are not activated at this time.</td></tr>
</table>

Program Example



◆ When X1=OFF→ON 时, D0 content are incremented by 1 automatically.

8.6 DEC/BIN Decrement

This instruction decrements the data of a specified device by "1".

Instruction	Operand Type	Function					
		16-bit Instruction	**Mnemonic**	**Operation Condition**	**32-bit Instruction**	**Mnemonic**	**Operation Condition**
D **FNC 25 DEC** **P**	**D.**	3 steps	DEC	Continuous Operation	5 steps	DDEC	Continuous Operation
			DECP	Pulse (Single) Operation		DDECP	Pulse (Single) Operation
Operand number	**D.**	Word device number storing data to be decremented by "1" Applicable devices: KnY, KnM, KnS, T, C, D, R, V, Z, modify				BIN16/32-bit	

Instruction Explanation	**1. 16-bit operation (DEC,DECP)** The contents of **D.** are decremented by "1", and the decremented result is transferred to **D.**. 2. **32-bit operation (DDEC,DDECP)** The contents of **[D.+1, D.]** are decremented by "1", and the decremented result is transferred to **[D.+1, D.]**. ● Note that data is decremented in every operation cycle in a continuous operation type instruction. ● 16-bit BIN operation,When "−32768" is decremented by "1", the result is "+32767". Flags (zero, carry and borrow) are not activated at this time. ● 32-bit BIN operation,When "−2,147,483,648" is decremented by "1", the result is "+2,147,483,6478". Flags (zero, carry and borrow) are not activated at this time.

Program Example

◆ When X1=OFF→ON, D0 content are decremented by 1 automatically.

8.7 WAND/Logical Word AND

This instruction executes the logical product (AND) operation of two numeric values.

Instruction			Operand Type	Function						
W D	FNC 26 AND	P	S1. S2. D.	**16-bit Instruction**	**Mnemonic**	**Operation Condition**		**32-bit Instruction**	**Mnemonic**	**Operation Condition**
				7 steps	WAND	Continuous Operation		13 steps	DAND	Continuous Operation
					WANDP	Pulse (Single) Operation			DANDP	Pulse (Single) Operation

Operand number			
Operand number	**S1.**	Data used for logical product or word device number storing data Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify	BIN16/32-bit
	S2.	Data used for logical product or word device number storing data Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify	BIN16/32-bit
	D.	Word device number storing the logical product result Applicable devices: KnY, KnM, KnS, T, C, D, R, V, Z, modify	BIN16/32-bit

Instruction Explanation

1. 16-bit operation (WAND,WANDP)

The logical product (AND) operation is executed to the contents of **S1.** and **S2.** in units of bit, and the result is transferred to **D.**.

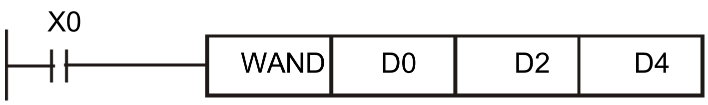
2. **32-bit operation (DAND,DANDP)**

The logical product (AND) operation is executed to the contents of **[S1.+1,S1.]** and **[S2.+1,S2.]** in units of bit, and the result is transferred to**[D.+1, D.]**.

- When a constant (K) is specified in the transfer source **S1.** and **S2 or [S1.+1,S1.]**and **[S2.+1,S2.]**, it is automatically converted into the binary format
- The logical product (AND) operation is executed in units of bit as shown in the table below (1 ∧ 1 = 1, 0 ∧ 1 = 0, 1∧ 0 = 0, 0 ∧ 0 = 0).

	S1. [S1.+1,S1.]	**S2. [S2.+1,S2.]**	**D. [D.+1, D.]**
Logical operation (unit: bit)	0	0	0
	1	0	0
	0	1	0
	1	1	1

Program Example



◆ When X0=ON, 16-bit D0 and D2 for WAND logical AND operation, the result will be restored in D4.

		b15															b0
Before execution	D0	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1
									WAND								
	D2	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0
									⇩								
After execution	D4	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0

8.8 WOR/Logical Word OR

This instruction executes the logical sum (OR) operation of two numeric values.

Instruction	Operand Type	Function					
		16-bit Instruction	**Mnemonic**	**Operation Condition**	**32-bit Instruction**	**Mnemonic**	**Operation Condition**
W **FNC 27 OR** **D** **P**	**S1.** **S2.** **D.**	7 steps	WOR	Continuous Operation	13 steps	DOR	Continuous Operation
			WORP	Pulse (Single) Operation		DORP	Pulse (Single) Operation

Operand number			
Operand number	**S1.**	Data used for logical sum or word device number storing data Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify	BIN16/32-bit
	S2.	Data used for logical sum or word device number storing data Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify	BIN16/32-bit
	D.	Word device number storing the logical sum result Applicable devices: KnY, KnM, KnS, T, C, D, R, V, Z, modify	BIN16/32-bit

Instruction Explanation

1. 16-bit operation (WOR,WORP)

The logical sum (OR) operation is executed to the contents of **S1.** and **S2.**in units of bit, and the result is transferred to **D.**.

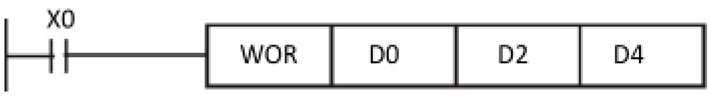
2. **32-bit operation (DOR,DORP)**

The logical sum (OR) operation is executed to the contents of **[S1.+1,S1.]** and**[S2.+1,S2.]** in units of bit, and the result is transferred to **[D.+1, D.]**.

- When a constant (K) is specified in the transfer source **S1.** and **S2 or [S1.+1,S1.]**and **[S2.+1,S2.]**, it is automatically converted into the binary format.
- The logical word OR operation is executed in units of bit as shown in the table below (1∨1=1 0∨1=1 0∨0=0 1∨0=1).

	S1. **[S1.+1,S1.]**	**S2.** **[S2.+1,S2.]**	**D.** **[D.+1, D.]**
Logical operation (unit: bit)	0	0	0
	1	0	1
	0	1	1
	1	1	1

Program Example



◆ When X0=ON, 16-bit D0 and D2 for WOR, logical word OR operation, the result will be restored in D4.

		b15															b0
Before execution	D0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
		WOR															
	D2	0	0	0	0	1	1	1	1	1	0	1	0	0	1	0	1
		⇩															
After execution	D4	0	1	0	1	1	1	1	1	1	1	1	1	0	1	0	1

8.9 WXOR/Logical Exclusive OR

This instruction executes the exclusive logical sum (XOR) operation of two numeric values.

Instruction	Operand Type	Function					
W / D / FNC 28 XOR / P	S1. S2. D.	**16-bit Instruction**	**Mnemonic**	**Operation Condition**	**32-bit Instruction**	**Mnemonic**	**Operation Condition**
		7 steps	WXOR	Continuous Operation	13 steps	DXOR	Continuous Operation
			WXORP	Pulse (Single) Operation		DXORP	Pulse (Single) Operation

Operand number			
Operand number	S1.	Data used for exclusive logical sum or word device number storing data **Applicable devices**: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify	BIN16/32-bit
	S2.	Data used for exclusive logical sum or word device number storing data **Applicable devices**: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify	BIN16/32-bit
	D.	Word device number storing the exclusive logical sum result **Applicable devices**: KnY, KnM, KnS, T, C, D, R, V, Z, modify	BIN16/32-bit

Instruction Explanation

1. 16-bit operation (WXOR,WXORP)

The exclusive logical sum (XOR) operation is executed to the contents of **S1.** and **S2.** in units of bit, and the result is transferred to **D.**.

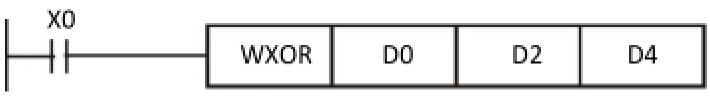
2. **32-bit operation (DXOR,DXORP)**

The exclusive logical sum (XOR) operation is executed to the contents of **[S1.+1,S1.]** and **[S2.+1,S2.]** in units of bit, and the result is transferred to **[D.+1, D.]**.

- When a constant (K) is specified in the transfer source **S1.** and **S2 or [S1.+1,S1.]**and **[S2.+1,S2.]**, it is automatically converted into the binary format.
- The logical exclusive OR operation is executed in units of bit as shown in the table below(1∀1=0 0∀0=0 1∀0=1 0∀1=1).

	S1. **[S1.+1,S1.]**	**S2.** **[S2.+1,S2.]**	**D.** **[D.+1, D.]**
Logical operation (unit: bit)	0	0	0
	1	0	1
	0	1	1
	1	1	0

Program Example



◆ When X0=ON, 16-bit D0 and D2 for WXOR, logical exclusive OR operation, the result will be restored in D4.

		b15															b0
Before execution	D0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
								WXOR									
	D2	0	0	0	0	1	1	1	1	1	0	1	0	0	1	0	1
								⇩									
After execution	D4	0	1	0	1	1	0	1	0	1	1	1	1	0	0	0	0

8.10 NEG/Negation

This instruction obtains the complement of a numeric value (by inverting each bit and adding "1").

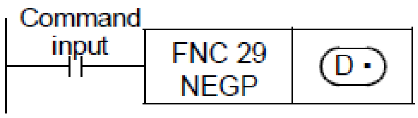
This instruction can be used to negate the sign of a numeric value.

Instruction	Operand Type	Function					
D FNC 29 NEG P	D.	**16-bit Instruction**	**Mnemonic**	**Operation Condition**	**32-bit Instruction**	**Mnemonic**	**Operation Condition**
		3 steps	NEG	Continuous Operation	13 steps	DNEG	Continuous Operation
			NEGP	Pulse (Single) Operation		DNEGP	Pulse (Single) Operation
Operand number	D.	Word device number which stores data for obtaining complement and will store the operation result (The operation result will be stored in the same word device number.) **Applicable devices**: KnY, KnM, KnS, T, C, D, R, V, Z, modify					BIN16/32-bit

Instruction Explanation

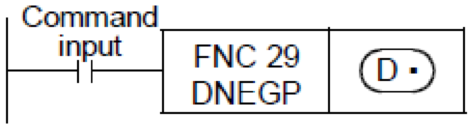
1. 16-bit operation (NEG,NEGP)

Each bit of **D.** is inverted (0 → 1, 1 → 0), "1" is added, and then the result is stored in the original device.

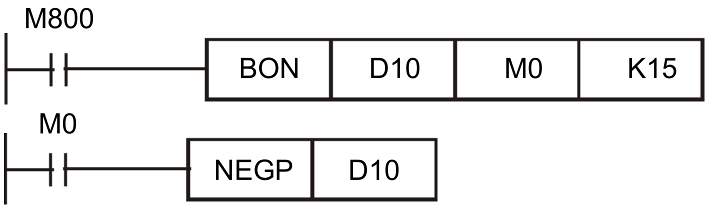


$(\overline{D\cdot})+1 \rightarrow D\cdot$

2. 32-bit operation (DNEG,DNEGP)

	Each bit of **[D.+1, D.]** is inverted (0 → 1, 1 → 0), "1" is added, and then the result is stored in the original device.  $(\overline{D\cdot + 1, D\cdot}) + 1 = (D\cdot + 1, D\cdot)$ ● When a constant (K) is specified in the transfer source **S1.** and **S2 or [S1.+1,S1.]**and **[S2.+1,S2.]**, it is automatically converted into the binary format ● Note that the complement is obtained in every operation cycle in a continuous operation type instruction

Program Example

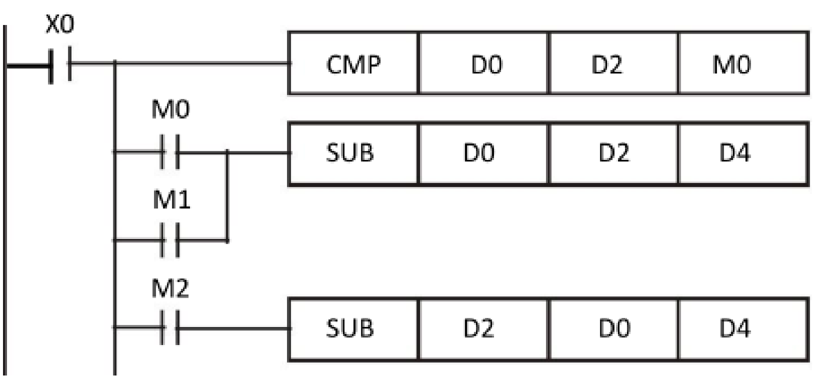


- **Obtaining the absolute value of a negative value using NEG instruction**

1) In BON (ON bit check) instruction, M0 turns ON when the bit 15 (b15 among b0 to b15) of D10 is "1".（D10 is negative value)

2) NEGP instruction is executed for D10 only when M0 turns ON.

Program Example



◆ **Obtaining the absolute value by SUB (subtraction) instruction,When X0=ON:**

1) If D0>D2, M0=ON。

2) If D0=D2, M1=ON。

3) If D0<D2, M2=ON。

4) Now D4 keep positive value.

More explanation

In PLCs, a negative value is expressed in 2’s complement.

When the most significant bit is "1", it is a negative value, and its absolute value can be obtained by NEG instruction.

(D 10)=2

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

(D 10)=1

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

(D 10)=0

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

(D 10)=−1 → $(\overline{D\ 10})+1=1$

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

(D 10)=−2 → $(\overline{D\ 10})+1=2$

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

(D 10)=−32,767 → $(\overline{D\ 10})+1=32,767$

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

(D 10)=−32,768 → $(\overline{D\ 10})+1=-32,768$

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

The absolute value can be obtained up to 32767.

9 Rotation and Shift Operation –FNC 30 to FNC 39

FNC NO.	Mnemonic	Function	Supported PLC series		
			3G PLC	2N PLC	MX2N PLC
30	ROR	Rotation Right	★	★	★
31	ROL	Rotation Left	★	★	★
32	RCR	Rotation Right with Carry	★	★	★
33	RCL	Rotation Left with Carry	★	★	★
34	SFTR	Bit Shift Right	★	★	★
35	SFTL	Bit Shift Left	★	★	★
36	WSFR	Word Shift Right	★	★	★
37	WSFL	Word Shift Left	★	★	★
38	SFWR	Shift write [FIFO/FILO control]	★	★	★
39	SFRD	Shift read [FIFO control]	★	★	★

9.1 ROR/Rotation Right

This instruction shifts and rotates the bit information rightward by the specified number of bits without the carry flag.

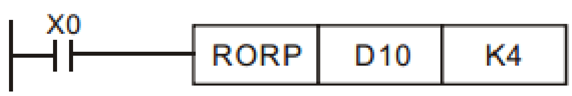
Instruction	Operand Type	Function						
D / FNC 30 ROR / P	D. n	**16-bit Instruction**	**Mnemonic**	**Operation Condition**		**32-bit Instruction**	**Mnemonic**	**Operation Condition**
		5 steps	ROR	Continuous Operation		9 steps	DROR	Continuous Operation
			RORP	Pulse (Single) Operation			DRORP	Pulse (Single) Operation

Operand number	D.	Word device number storing data to be rotated rightward **Applicable devices**: KnY, KnM, KnS, T, C, D, R, V, Z, modify	BIN16/32-bit
	n	Number of bits to be rotated [n ≤ 16 (16-bit instruction), n ≤ 32 (32-bit instruction)][*1] **Applicable devices**: D, R, K, H	BIN16/32-bit

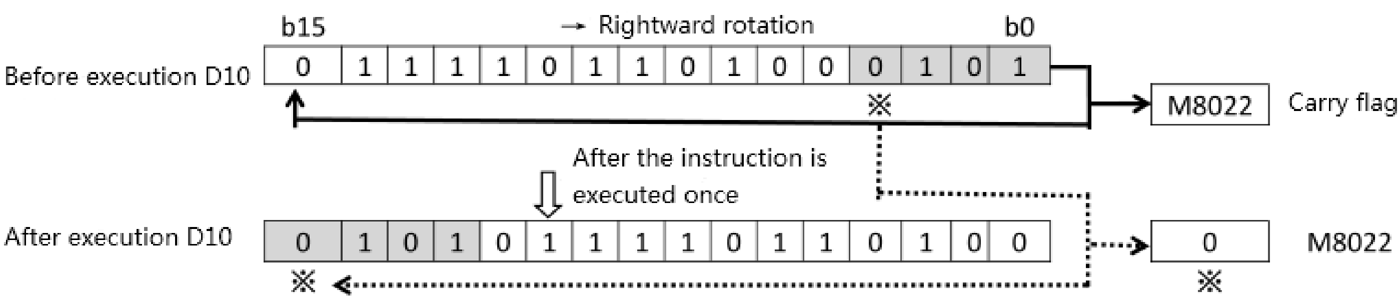
*1: Do not set the Bit number of rotations to a negative value.

Instruction Explanation	**1. 16-bit operation (ROR,RORP)** "n" bits out of 16 bits of **D.** are rotated rightward. 2. **32-bit operation (DROR,DRORP)** "n" bits out of 32 bits of **[D.+1, D.]** are rotated rightward. ● The final bit is stored in the carry flag (M8022). ● In a device with digit specification, K8 (32-bit instruction) is valid

Program Example



◆ When X0 change from "OFF" to "ON", 16bits of D10are rotated rightward as 4bits one group, bit content of ※ is transferred to carry flag M8022, as below picture.



9.2 ROL/Rotation Left

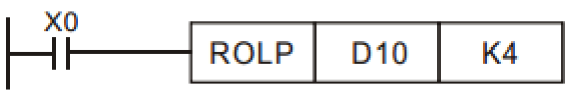
This instruction shifts and rotates the bit information leftward by the specified number of bits without the carry flag.

<table>
<tr><th>Instruction</th><th>Operand Type</th><th colspan="6">Function</th></tr>
<tr><td rowspan="3">D | FNC 31 ROL | P</td><td rowspan="3">D.
n</td><td>16-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td><td>32-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td></tr>
<tr><td rowspan="2">5 steps</td><td>ROL</td><td>Continuous Operation</td><td rowspan="2">9 steps</td><td>DROL</td><td>Continuous Operation</td></tr>
<tr><td>ROLP</td><td>Pulse (Single) Operation</td><td>DROLP</td><td>Pulse (Single) Operation</td></tr>
<tr><td rowspan="2">Operand number</td><td>D.</td><td colspan="5">Word device number storing data to be rotated leftward
Applicable devices: KnY, KnM, KnS, T, C, D, R, V, Z, modify</td><td>BIN16/32-bit</td></tr>
<tr><td>n</td><td colspan="5">Number of bits to be rotated [n ≤ 16 (16-bit instruction), n ≤ 32 (32-bit instruction)]*1
Applicable devices: D, R, K, H</td><td>BIN16/32-bit</td></tr>
</table>

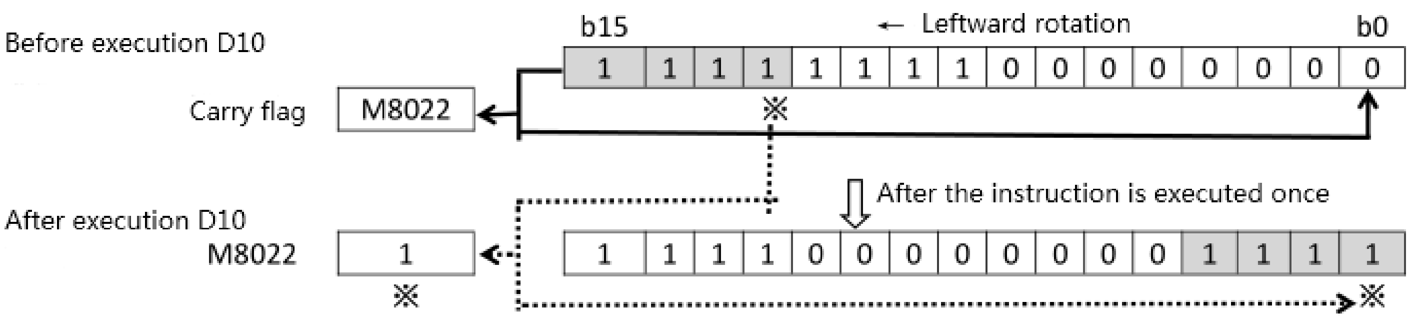
*1: Do not set the bit number of rotations to a negative value.

<table>
<tr><td>Instruction Explanation</td><td>1. 16-bit operation (ROL,ROLP)
"n" bits out of 16 bits of D. are rotated leftward.
2. 32-bit operation (DROL,DROLP)
"n" bits out of 32 bits of [D.+1, D.] are rotated leftward.
● The final bit is stored in the carry flag (M8022).
● In a device with digit specification, K4 (16-bit instruction) is valid.</td></tr>
</table>

Program Example



◆ When X0 change from “OFF” to “ON”, 16bits of D10are rotated leftward as 4bits one group, bit content of ※ is transferred to carry flag M8022, as below picture.



9.3 RCR/Rotation Right with Carry

This instruction shifts and rotates the bit information rightward by the specified number of bits together with the carry flag

Instruction	Operand Type	Function					
		16-bit Instruction	**Mnemonic**	**Operation Condition**	**32-bit Instruction**	**Mnemonic**	**Operation Condition**
D **FNC 32 RCR** P	**D.** **n**	5 steps	RCR	Continuous Operation	9 steps	DRCR	Continuous Operation
			RCRP	Pulse (Single) Operation		DRCRP	Pulse (Single) Operation

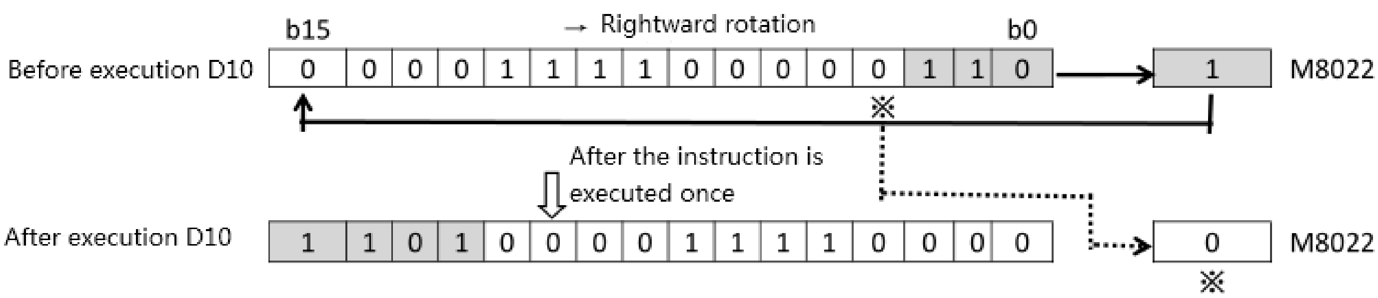
Operand number			
Operand number	**D.**	Word device number storing data to be rotated rightward **Applicable devices**: KnY, KnM, KnS, T, C, D, R, V, Z, modify	BIN16/32-bit
	n	Number of bits to be rotated [n ≤ 16 (16-bit instruction), n ≤ 32 (32-bit instruction)][*1] **Applicable devices**: D, R, K, H	BIN16/32-bit

*1: Do not set the bit number of rotations to a negative value.

Instruction Explanation	
Instruction Explanation	**1. 16-bit operation (RCR,RCRP)** "n" bits out of 16 bits of **D.** and 1 bit (carry flag M8022) are rotated rightward. **2. 32-bit operation (DRCR,DRCRP)** "n" bits out of 32 bits of **[D.+1, D.]** and 1 bit (carry flag M8022) are rotated rightward. ● The carry flag is intervened in the rotation loop. If M8022 has been set to ON or OFF before the rotation instruction, the carry flag is transferred to the destination Operand number. ● In a device with digit specification, K4 (16-bit instruction) / K8(32-bit instruction)is valid.

Program Example

◆ When X0 change from “OFF” to “ON”, Total 17bits that 16bits of D10 and carry flag M8022 are rotated rightward as 4bits one group, bit content of ※ is transferred to carry flag M8022, as below picture.



9.4 RCL/Rotation Left with Carry

This instruction shifts and rotates the bit information leftward by the specified number of bits together with the carry flag.

Instruction	Operand Type	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
D FNC 33 RCL P	D. n	5 steps	RCL	Continuous Operation	9 steps	DRCL	Continuous Operation
			RCLP	Pulse (Single) Operation		DRCLP	Pulse (Single) Operation

Operand number			
Operand number	D.	Word device number storing data to be rotated leftward **Applicable devices**: KnY, KnM, KnS, T, C, D, R, V, Z, modify	BIN16/32-bit
	n	Number of bits to be rotated [n ≤ 16 (16-bit instruction), n ≤ 32 (32-bit instruction)][*1] **Applicable devices**: D, R, K, H	BIN16/32-bit

*1: Do not set the bit number of rotations to a negative value.

Instruction Explanation

1. 16-bit operation (RCL,RCLP)

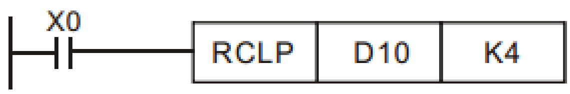
n" bits out of 16 bits of **D.** and 1 bit (carry flag M8022) are rotated leftward.

2. 32-bit operation (DRCL,DRCLP)

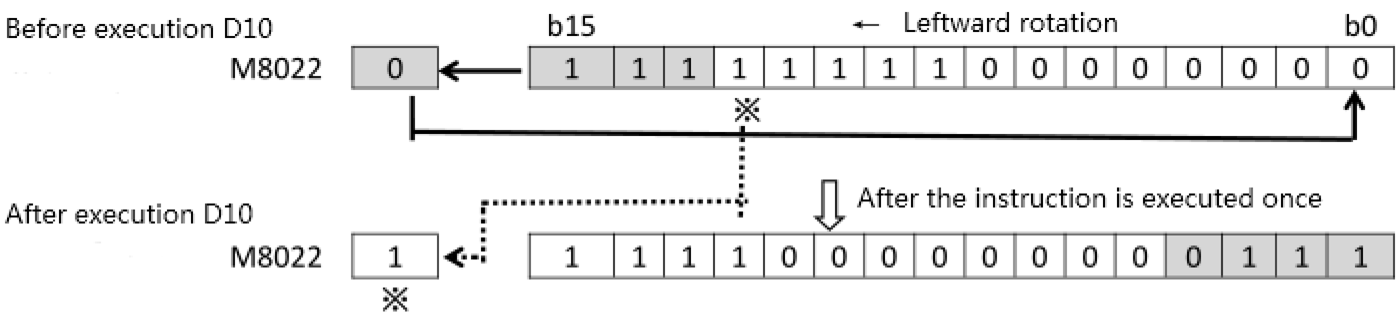
"n" bits out of 32 bits of **[D.+1, D.]** and 1 bit (carry flag M8022) are rotated leftward.

- The carry flag is intervened in the rotation loop. If M8022 has been set to ON or OFF before the rotation instruction, the carry flag is transferred to the destination Operand number.
- In a device with digit specification, K4 (16-bit instruction) / K8(32-bit instruction)is valid.

Program Example



◆ When X0 change from “OFF” to “ON”, Total 17bits that 16bits of D10 and carry flag M8022 are rotated leftward as 4bits one group, bit content of ※ is transferred to carry flag M8022, as below picture.



9.5 SFTR/Bit Shift Right

This instruction shifts bit devices of the specified bit length rightward by the specified number of bits. After shift, the bit device **S.** is transferred by "n2" bits from the most significant bit.

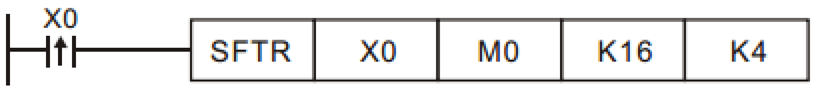
Instruction		Operand Type	Function					
			16-bit Instruction	**Mnemonic**	**Operation Condition**	**32-bit Instruction**	**Mnemonic**	**Operation Condition**
FNC 34 SFTR		**S. D. n1 n2**	9 steps	SFTR	Continuous Operation		--	
	P			SFTRP	Pulse (Single) Operation		--	

Operand number	**S.**	Head bit device number to be stored to the shift data after rightward shift **Applicable devices**: X, Y, M, S, D □.n, modify	bit
	D.	Head bit device number to be shifted rightward **Applicable devices**: Y, M, S, modify	bit
	n1	Bit length of the shift data n2 ≤ n1 ≤ 1024 **Applicable devices**: K, H	BIN16-bit
	n2	Number of bits to be shifted rightward n2 ≤ n1 ≤ 1024*1 **Applicable devices**: D, R, K, H	BIN16-bit

*1: Do not set the bit number of shift right to a negative value.

Instruction Explanation	**16-bit operation (SFTR,SFTRP)** For "n1" bits (shift register length) starting from **D.**, "n2" bits are shifted rightward ([1] and [2] shown below). After shift, "n2" bits from **S.**are transferred to "n2" bits from **D.**+n1-n2 ([3] shown below). ● 112FNote that "n2" bits are shifted every time the command input turns ON from OFF in SFTRP instruction, but that "n2" bits are shifted in each scan time (operation cycle) in SFTR instruction.

Program Example



- When X0 is in rising edge, 16bits (M0~M15) shift right as 4bits. Shift right of each scan acts as below ①~⑤.

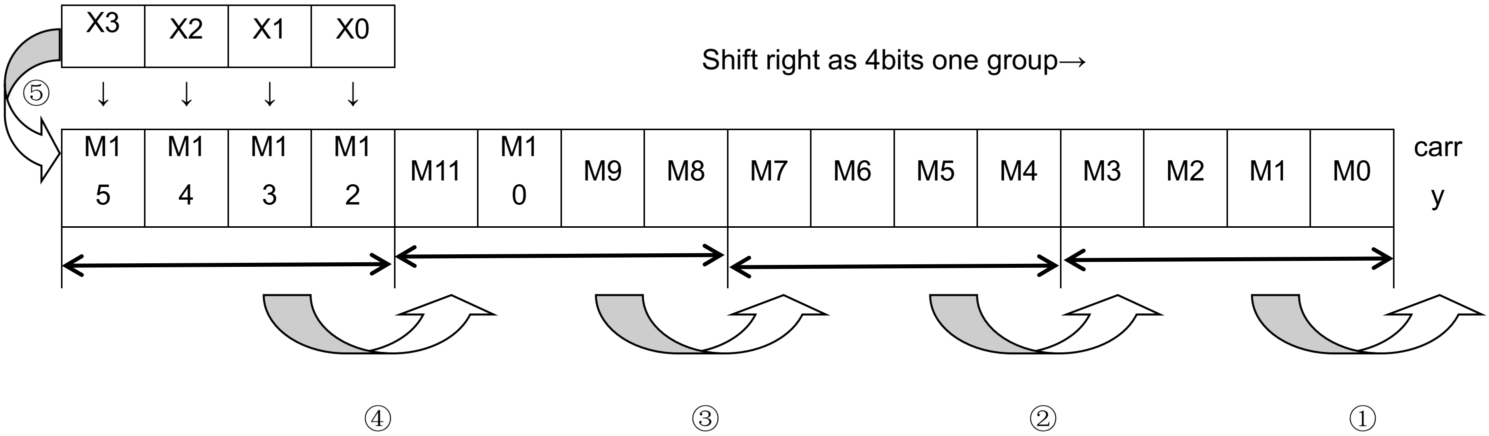
① M3~M0 → carry

② M7~M4 → M3~M0

③ M11~M8 → M7~M4

④ M15~M12 → M11~M8

⑤ X3~X0 → M15~M12 completed



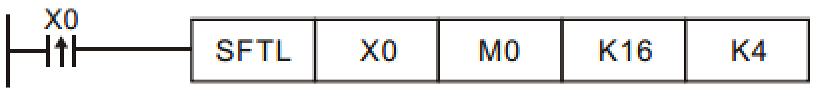
9.6 SFTL/Bit Shift Left

This instruction shifts bit devices of the specified bit length leftward by the specified number of bits. After shift, the bit device **S.** is transferred by "n2" bits from the least significant bit.

Instruction		Operand Type	Function					
			16-bit Instruction	**Mnemonic**	**Operation Condition**	**32-bit Instruction**	**Mnemonic**	**Operation Condition**
FNC 35 SFTL		**S.** **D.** **n1** **n2**	9 steps	SFTL	Continuous Operation		--	
	P			SFTLP	Pulse (Single) Operation		--	
Operand number		**S.**	Head bit device number to be stored to the shift data after leftward shift **Applicable devices**: X, Y, M, S, D □.n, modify				bit	
		D.	Head bit device number to be shifted leftward **Applicable devices**: Y, M, S, modify				bit	

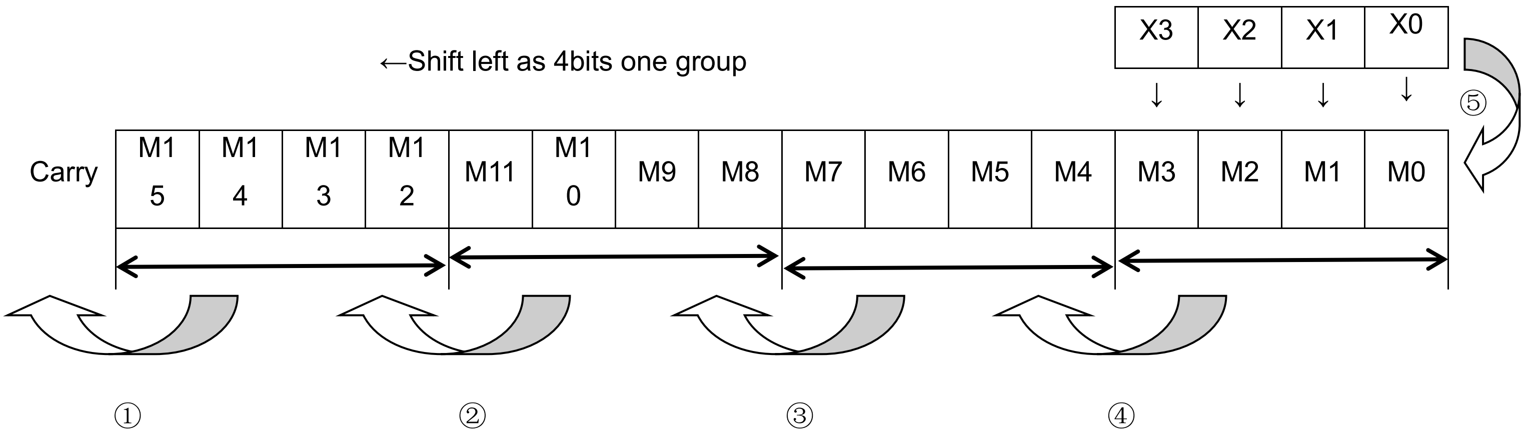
	n1	Bit length of the shift data n2 ≤ n1 ≤ 1024 **Applicable devices**: K, H	BIN16-bit
	n2	Number of bits to be shifted leftward n2≦n1≦1024 **Applicable devices**: D, R, K, H	BIN16-bit
	*1: Do not set the bit number of shift left to a negative value.		
Instruction Explanation	**16-bit operation (SFTL,SFTLP)** For "n1" bits (shift register length) starting from **D.** , "n2" bits are shifted leftward After shift, "n2" bits from **S.** are transferred to "n2" bits from **D.+n1-n2.** ● Note that "n2" bits are shifted every time the command input turns ON from OFF in SFTLP instruction, but that "n2" bits are shifted in each operation cycle in SFTL instruction.		

Program Example



◆ When X0 is in rising edge, 16bits (M0~M15) shift left as 4bits. Shift left of each scan acts as below ①~⑤.

①	M15~M12	→	Carry
②	M11~M8	→	M15~M12
③	M7~M4	→	M11~M8
④	M3~M0	→	M7~M4
⑤	X3~X0	→	M3~M0 Complete



9.7 WSFR/Word Shift Right

This instruction shifts word devices with "n1" data length rightward by "n2" words.

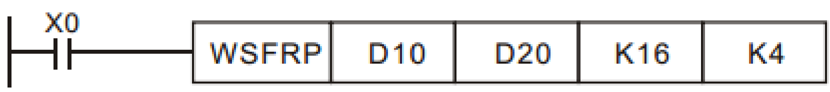
Instruction	Operand Type	Function					
		16-bit Instruction	**Mnemonic**	**Operation Condition**	**32-bit Instruction**	**Mnemonic**	**Operation Condition**
FNC 36 WSFR	**S.** **D.** **n1** **n2**	9 steps	WSFR	Continuous Operation		--	
P			WSFRP	Pulse (Single) Operation		--	

Operand number			
Operand number	**S.**	Head device number to be stored to the shift data after rightward shift **Applicable devices**: KnX, KnY, KnM, KnS, T, C, D, R, modify	BIN16-bit
	D.	Head word device number storing data to be shifted rightward **Applicable devices**: KnY, KnM, KnS, T, C, D, R, modify	BIN16-bit
	n1	Word data length of the shift data n2≦n1≦512 **Applicable devices**: K, H	BIN16-bit
	n2	Number of words to be shifted rightward n2≦n1≦512[*1] **Applicable devices**: D, R, K, H	BIN16-bit

*1: Do not set the number of word shift right to a negative value.

Instruction Explanation	
	16-bit operation (WSFR、WSFRP) For "n1" word devices starting from **D.**, "n2" words are shifted rightward After shift, "n2" words starting from **S.** are shifted to "n2" words starting from [**D.+n1-n2**] ● Note that "n2" words are shifted when the drive input turns ON in WSFRP instruction, but that "n2" words are shifted in each operation cycle in WSFR instruction.

Program Example



◆ When X0 change From “OFF” to “ON”, 16registers (D20-D35) are listed as the shift area, and shifted to the right by 4 registers. Shift right of each scan acts as below ①~⑤.

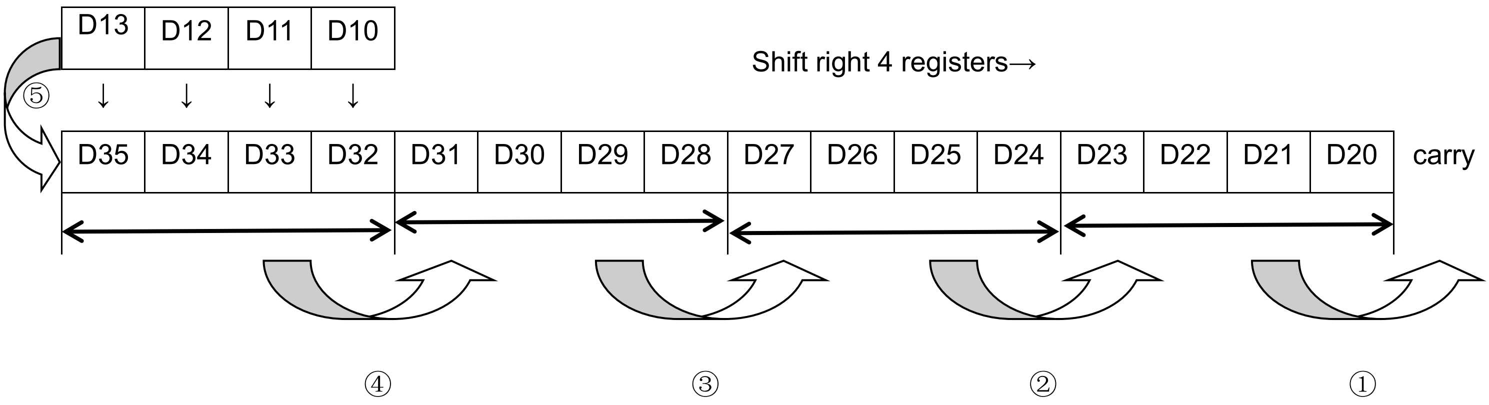
① D23~D20 → carry

② D27~D24 → D23~D20

③ D31~D28 → D27~D24

④ D35~D32 → D31~D28

⑤ D13~D10 → D35~D32 complete



9.8 WSFL/Word Shift Left

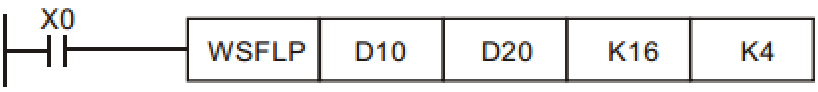
This instruction shifts the word data information leftward by the specified number of words.

<table>
<tr><th>Instruction</th><th>Operand Type</th><th colspan="7">Function</th></tr>
<tr><td rowspan="3">FNC 37
WSFL
P</td><td rowspan="3">S.
D.
n1
n2</td><td>16-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td><td></td><td>32-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td></tr>
<tr><td>9 steps</td><td>WSFL</td><td>Continuous Operation</td><td></td><td></td><td>--</td><td></td></tr>
<tr><td></td><td>WSFLP</td><td>Pulse (Single) Operation</td><td></td><td></td><td>--</td><td></td></tr>
<tr><td rowspan="4">Operand number</td><td>S.</td><td colspan="6">Head device number to be stored to the shift data after leftward shift
Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, modify</td><td>BIN16-bit</td></tr>
<tr><td>D.</td><td colspan="6">Head word device number storing data to be shifted leftward
Applicable devices: KnY, KnM, KnS, T, C, D, R, modify</td><td>BIN16-bit</td></tr>
<tr><td>n1</td><td colspan="6">Word data length of the shift data n2≦n1≦512
Applicable devices: K, H</td><td>BIN16-bit</td></tr>
<tr><td>n2</td><td colspan="6">Number of words to be shifted leftward n2≦n1≦512*1
Applicable devices: D, R, K, H</td><td>BIN16-bit</td></tr>
</table>

*1: Do not set the number of word shift left to a negative value.

Instruction Explanation	**16-bit operation (WSFL,WSFLP)** For "n1" word devices starting from **D.**, "n2" words are shifted leftward. After shift, "n2" words starting from **S.** are shifted to "n2" words starting from **D.** ● Note that "n2" words are shifted when the drive input turns ON from OFF in WSFLP instruction, but that "n2" words are shifted in each operation cycle in WSFL instruction.

Program Example



◆ When X0 change From "OFF" to "ON", 16registers (D20-D35) are listed as the shift area, and shifted to the left by 4 registers. Shift left of each scan acts as below ①~⑤.

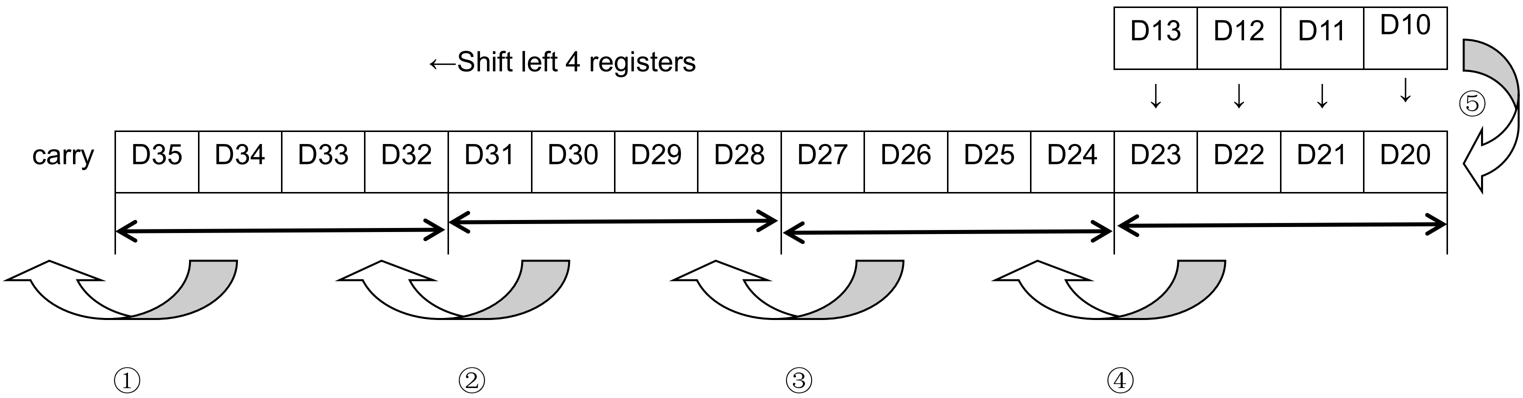
① D35~D32 → carry

② D31~D28 → D35~D32

③ D27~D24 → D31~D28

④ D23~D20 → D27~D24

⑤ D13~D10 → D23~D20 complete



9.9 SFWR/Shift Write [FIFO/FILO Control]

This instruction writes data for first-in first-out (FIFO) and last-in first-out (LIFO) control.

Instruction	Operand Type	Function					
FNC 38 SFWR P	S. D. n	**16-bit Instruction**	**Mnemonic**	**Operation Condition**	**32-bit Instruction**	**Mnemonic**	**Operation Condition**
		7steps	SFWR	Continuous Operation		--	
			SFWRP	Pulse (Single) Operation		--	

Operand number	**S.**	Word device number storing data to be put in first **Applicable devices**: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify	BIN16-bit
	D.	Head word device number storing data (The first word device works as the pointer, and data is stored in **D.+1** and later) **Applicable devices**: KnY, KnM, KnS, T, C, D, R, modify	BIN16-bit
	n	Number of store points +1 (pointer part) 2≦n≦512 **Applicable devices**: K, H	BIN16-bit

Instruction Explanation

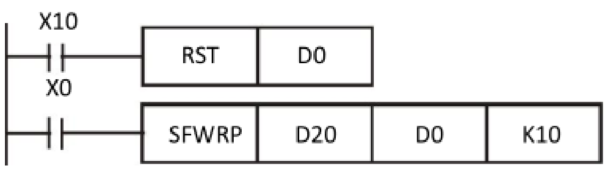
16-bit operation (SFWR,SFWRP)

The contents of **S.** are written to "n-1" devices from **D.+1**, and "1" is added to the number of data stored in **D.**.

For example: when D.=0, write D.+1；When D.=1, write D.+2。

- Note that data are stored (overwritten) in each scan time (operation cycle).
- When the contents of the pointer **D.** exceeds "n-1", no operation is executed (so data is not written) and the carry flag M8022 turns ON.
- the number of stored data is specified by the contents of the pointer
- SFWR and SFRD instruction can be used together, execute write/read control of FIFO/FILO data listed.

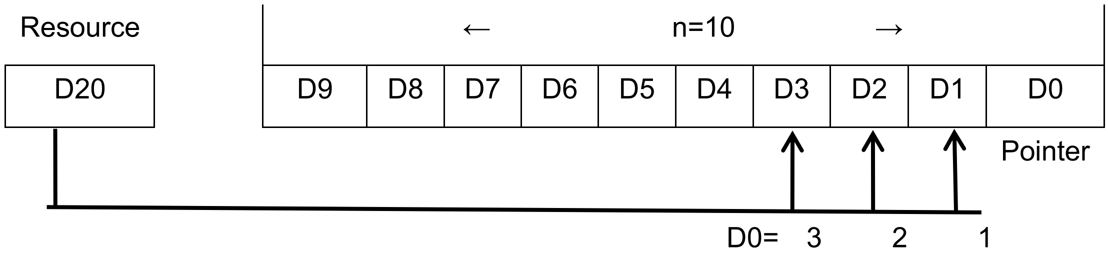
Program Example



◆ Clear the pointer D0, when X0 change from OFF to ON, content of D20 is transferred to D1, pointer D0 content becomes to 1. After changing D20 content, again change X0 from OFF to ON, then content of D20 is transferred to D2, D0 content becomes to 2.

◆ The instruction execute Shift Write once ,act as numbers 1.2.3...:

1. content of D20 is transferred to D1.
2. pointer D0 content becomes to 1.
3. Process below is same, act from right in orders, indicate the data storing numbers in content of Pointer D0.



9.10 SFRD/Shift Read [FIFO Control]

This instruction reads data for first-in first-out control.

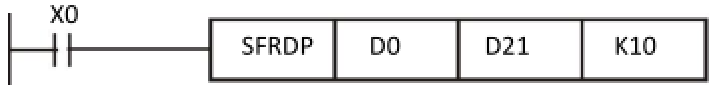
Instruction	Operand Type	Function					
		16-bit Instruction	**Mnemonic**	**Operation Condition**	**32-bit Instruction**	**Mnemonic**	**Operation Condition**
FNC 39 SFRD	**S.** **D.** **n**	7 steps	SFRD	Continuous Operation		--	
P			SFRDP	Pulse (Single) Operation		--	

Operand number			
Operand number	**S.**	Head word device number storing data (The first word device works as the pointer, and data is stored from **S.+1**) **Applicable devices**: KnY, KnM, KnS, T, C, D, R, modify	BIN16-bit
	D.	Word device number storing data taken out first **Applicable devices**: KnY, KnM, KnS, T, C, D, R, V, Z, modify	BIN16-bit
	n	Number of store points plus "1"* 2≦n≦512 **Applicable devices**: K, H	BIN16-bit

Instruction Explanation	
	16-bit operation (SFRD,SFRDP) **S.+1** written in turn by SFWR (FNC 38) instruction is transferred (read) to **D.**, and "n-1" words from **S.+1** are shifted rightward by 1 word. "1" is subtracted from the number of data stored in **S.**. ● When use SFRD Continuous Operation,Each scan circle(operation circle) will execute and read in turns, but content of **S.+n** do not change. ● When content of **S.** is 0,no operand, and the content of **D.** will not change,the zero flag M8020 turns **ON.** ● Data after reading was executed,The contents of **S.+n** do not change by reading.

	● SFWR and SFRD instruction can be used together, execute write/read control of FIFO/FILO data listed.

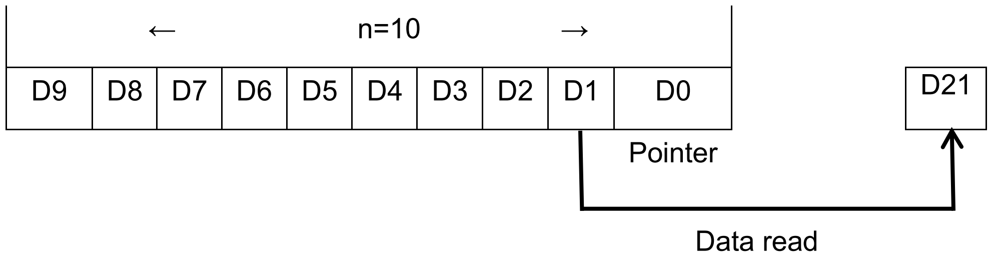
Program Example



◆ When X0=OFF→ON, the content of D1 is transferred to D21, D9~D2 are shifted right by 1 register(D9 keep same), Pointer D0 content decrease 1.

◆ The instruction execute Shift Read once act as numbers 1.2.3.

1. Content of D1 is read and transferred to D21.
2. D9~D2 all are shifted rightward by 1 register.
3. Content of Pointer D0 decrease 1.



10 Data Operation- FNC 40 to FNC 49

FNC NO.	Mnemonic	Function	Supported PLC series		
			3G PLC	2N PLC	MX2N PLC
40	ZRST	Zone Reset	★	★	★
41	DECO	Decode	★	★	★
42	ENCO	Encode	★	★	★
43	SUM	Sum of Active Bits	★	★	★
44	BON	Check Specified of ON bits	★	★	★
45	MEAN	Mean	★	★	★
46	ANS	Timed Annunciator Set	★		★
47	ANR	Annunciator Reset	★		★
48	SQR	BIN Square Root	★	★	★
49	FLT	BIN Conversion to Floating Point	★	★	★

10.1 ZRST/Zone Reset

This instruction resets devices located in a zone between two specified devices at one time.

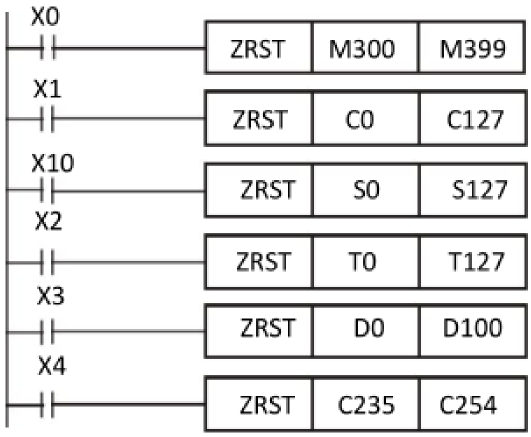
Use this instruction for restarting operation from the beginning after pause or after resetting control data.

<table>
<tr><td>Instruction</td><td>Operand Type</td><td colspan="6">Function</td></tr>
<tr><td rowspan="3">FNC 40
ZRST
P</td><td rowspan="3">D1.
D2.</td><td>16-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td><td>32-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td></tr>
<tr><td rowspan="2">5 steps</td><td>ZRST</td><td>Continuous Operation</td><td rowspan="2"></td><td>--</td><td></td></tr>
<tr><td>ZRSTP</td><td>Pulse (Single) Operation</td><td>--</td><td></td></tr>
<tr><td rowspan="2">Operand number</td><td>D1.</td><td colspan="3">Head bit or word device number to be reset at one time
Applicable devices: Y, M, S, T, C, D, R, modify</td><td colspan="2" rowspan="2">D1.≤D2.
Specify same type of devices.</td><td>BIN16-bit</td></tr>
<tr><td>D2.</td><td colspan="3">Last bit or word device number to be reset at one time
Applicable devices: Y, M, S, T, C, D, R, modify</td><td>BIN16-bit</td></tr>
</table>

<table>
<tr><td>Instruction Explanation</td><td>16-bit operation (ZRST,ZRSTP)

Same type of devices from to are reset at one time.
● Specify same type of devices in D1. and D2.. The device number of D1. should be smaller than or equal to the device number of D2.
● If the device number of D1. is larger than the device number of D2.,only one device specified in D1. is reset.
● ZRST instruction is handled as the 16-bit type, but 32-bit counters can be specified in D1. and D2. . However, it is not possible to specify a 16-bit counter in D1. and specify a 32-bit counter in D2.; D1. and D2. should be a same type.</td></tr>
</table>

Program Example



- When X0=ON, auxiliary relays M300 ~ M399 are cleared to OFF.
- When X1=ON, all 16-bit counters C0 ~ C127 are cleared. (Write 0, and clear contact and coil to OFF)
- When X10=ON, the step points S0 ~ S127 are cleared to OFF.
- When X2=ON, all timers T0 ~ T127 are cleared. (Write 0, and clear contact and coil to OFF)
- When X3=ON, the data in data registers D0 ~ D100 are cleared to 0.
- When X4=ON, all 32-bit counters C235 ~ C254 are cleared. (Write 0, and clear contact and coil to OFF)

Extended instruction

1. RST is an independent reset instruction for device,. RST instruction can be used for bit devices (Y,M,S) and word devices (T, C, D and R).
   For example: RST M0, RST T0.
2. FMOV (FNC 16) instruction is provided to write a constant (example: K0) at one time to devices (KnY, KnM, KnS, T, C, D and R) to clear to 0.
   For example: FMOV K0 D0 K100（write K0 to D0~D99)

10.2 DECO/Decode

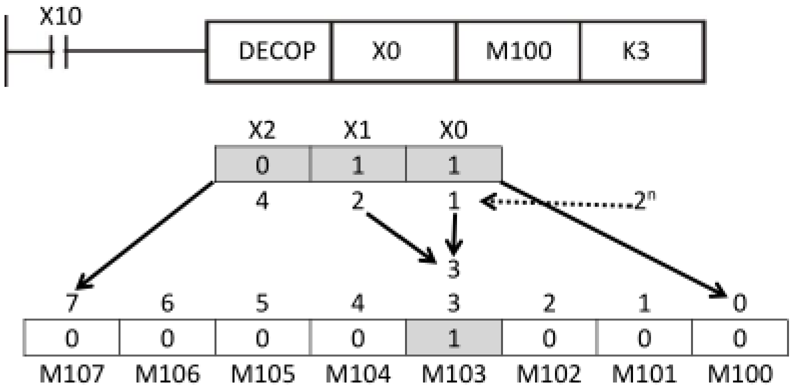
This instruction converts numeric data into ON bit.

A bit number which is set to ON by this instruction indicates a numeric value.

Instruction		Operand Type	Function					
			16-bit Instruction	**Mnemonic**	**Operation Condition**	**32-bit Instruction**	**Mnemonic**	**Operation Condition**
FNC 41 DECO		**S.** **D.** **n**	7 steps	DECO	Continuous Operation		--	
	P			DECOP	Pulse (Single) Operation		--	
Operand number		**S.**	Data to be decoded or word device number storing data **Applicable devices**: X, Y, M, S, T, C, D, R, V, Z, K, H, modify			BIN16-bit		
		D.	Bit or word device number storing the decoding result **Applicable devices**: Y, M, S, T, C, D, R, modify			BIN16-bit		
		n	Number of bits of device storing the decoding result (n = 1 to 8) (No processing is executed in the case of "n = 0".) **Applicable devices**: K, H			BIN16-bit		

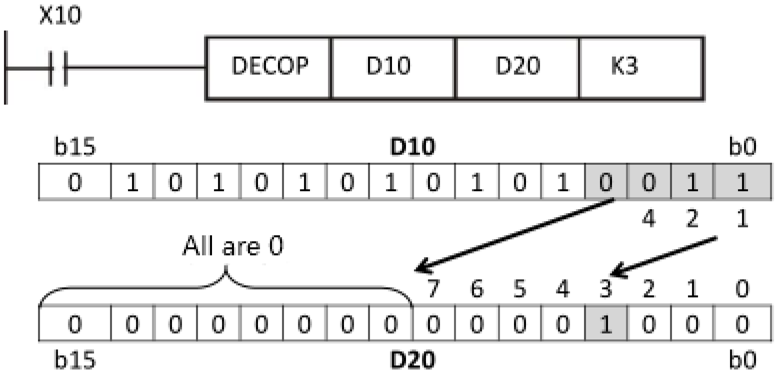
Instruction Explanation	**16-bit operation (DECO,DECOP)** One bit among **D.~D.+2ⁿ-1** is set to ON according to the **S.**value. ● When **D.** is a bit device (1 ≤ n ≤ 8),The numeric value (expressed in 2^n , 1 ≤ n ≤ 8) of a device specified by is decoded to **D.** . ——When all bits of **S.** are "0", the bit device turns ON. When "n" is "8", 2^8 points (= 256 bits which is the maximum value) are occupied.. ● When **D.** is a word device (1 ≤ n ≤ 4),The numeric value (expressed in 2^n , on the low-order side) of **S.** is decoded to **D.** . ——When all bits of **S.** are "0", b0 of the word device **D.** turns ON. In the case of "n ≤ 3", all of high-order bits of **D.** become "0" (turn OFF).

Program Example



- When X10=OFF→ON, DECO instruction decodes the content value of X0~X2 to M100~M107.
- When the value of X0~X3 is 3(1+2+0), the 3rd bit M103 from M100 is set to 1.
- When the DECO instruction is executed, and X10 becomes OFF, The decoder output acted as usual.

Program Example



- When X10=OFF→ON, DECO instruction decodes the content value of (b2~b0) in D10 to (b7~b0) of D20 All unused bits (b15~b8) in D20 become 0.
- The lower 3 bits of D10 are decoded and stored in the lower 8 bits of D20, and the upper 8 bits are all 0.
- When the DECO instruction is executed, and X10 becomes OFF, The decoder output acted as usual.

10.3 ENCO/Encode

This instruction obtains positions in which bits are ON in data.

Instruction		Operand Type	Function						
FNC 42 ENCO	P	S. D. n	**16-bit Instruction**	**Mnemonic**	**Operation Condition**		**32-bit Instruction**	**Mnemonic**	**Operation Condition**
			7 steps	ENCO	Continuous Operation			--	
				ENCOP	Pulse (Single) Operation			--	

Operand number	S.	Data to be encoded or word device number storing data **Applicable devices**: X, Y, M, S, T, C, D, R, V, Z, K, H, modify	BIN16-bit
	D.	Word device number storing the encoding result **Applicable devices**: T, C, D, R, V, Z, modify	BIN16-bit
	n	Number of bits of device storing the encoding result (n = 1 to 8) (When "n" is "0", no processing is executed.) **Applicable devices**: K, H	BIN16-bit

Instruction Explanation

16-bit operation (ENCO,ENCOP)

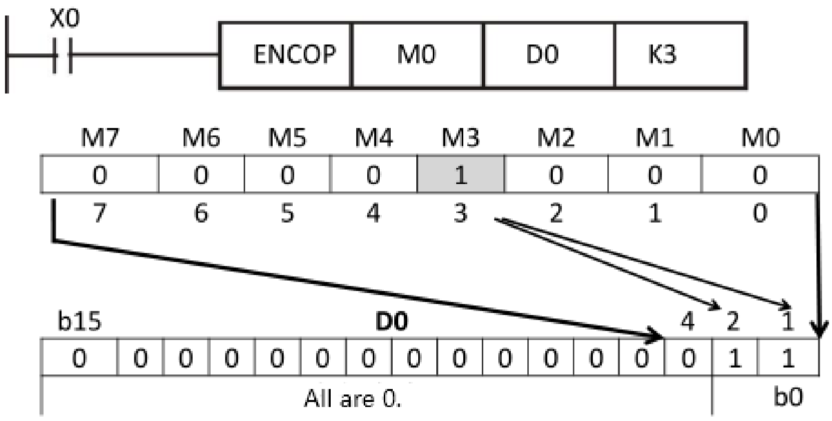
The 2^n bit of **S.** is encoded, and the result value is stored to **D.**.
This instruction converts data into BIN data according to a bit position in the ON status.

- When **S.** is a bit device ((1≦n≦8), ON bit positions among "2^n" bits (1 ≤ n ≤ 8) from **S.** are encoded to (1≦n≦8)**D.**.

  ——The encoding result of **D.** is "0" (OFF) from the most significant bit to the low-order bit "n".
  When "n" is "8", 2^8 = 256 bits (which is the maximum value) are occupied.
- When is a word device **S.** (1≦n≦4), ON bit positions among "2^n" bits (1 ≤ n ≤ 4) from a device specified in **S.** are encoded to **D.** .

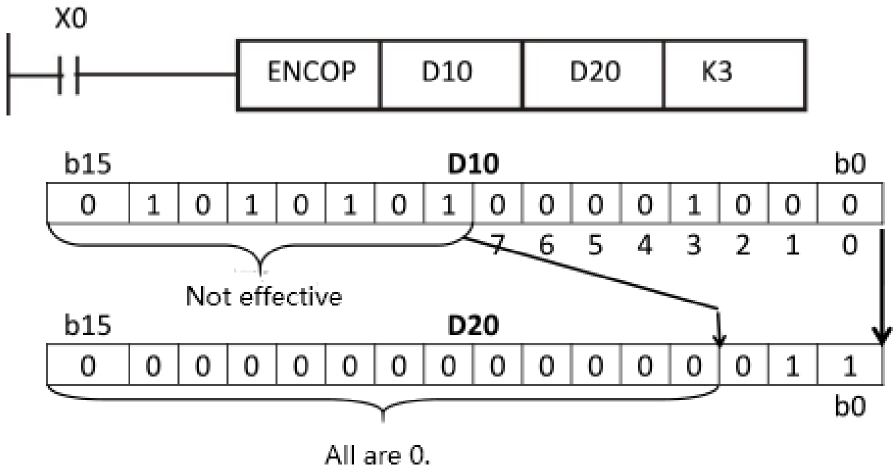
  ——The encoding result of **D.** is "0" (OFF) from the most significant bit to the low-order bit "n".

Program Example



- ◆ When X10=OFF→ON,2^3 bits（M0~ M7)of data is decoded by ENCO instruction and stored in the lower 3bits (b2~b0) of D0,All unused bits (b15~b3) in D0 become 0.
- ◆ When the ENCO instruction is executed, and X10 becomes OFF, data in D don’t change.

Program Example



- ◆ When X0=OFF→ON, the 2^3 bits of data (b0~b7) in D10 are stored in the lower 3 bits (b2~b0) of D20, and all unused bits (b15~b3) in D20 become 0 . (B8~b15 in D10 are invalid data)
- ◆ When the ENCO instruction is executed, X0 turns OFF, and the data in D don’t change.

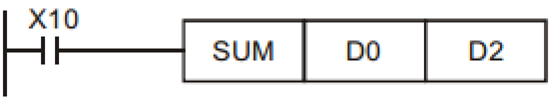
10.4 SUM/Sum of Active Bits

Calculate how many "1" (ON) instructions are in the data of the specified device.

Instruction	Operand Type	Function					
D FNC43 SUM P	S. D.	**16-bit Instruction**	**Mnemonic**	**Operation Condition**	**32-bit Instruction**	**Mnemonic**	**Operation Condition**
		5 steps	SUM	Continuous Operation	9 steps	DSUM	Continuous Operation
			SUMP	Pulse (Single) Operation		DSUMP	Pulse (Single) Operation
Operand number	S.	Word device number storing the source data **Applicable devices**: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify					BIN16/32-bit
	D.	Word device number storing the result data **Applicable devices**: KnY, KnM, KnS, T, C, D, R, V, Z, modify					BIN16/32-bit

Instruction Explanation	
	1. 16-bit operation (SUM,SUMP) The number of bits in the ON status in **S.** is counted, and stored to **D.**. **2. 32-bit operation (DSUN,DSUMP)** The number of bits in the ON status in **[S.+1, S.]** is counted, and stored to **D.**. ● When all bits are OFF in **S.** or **[S.+1, S.]**, the zero flag M8020 turns ON. ● The number of bits in the ON status are stored in **D.**, and K0 is stored in **D.+1**.

Program Example



b15 … b0

0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0		K3
D0																	**D2**

◆ When X10 is ON, Among 16bits of D0, the total number of bits with a content of "1" is stored in D2.

10.5 BON/Check Specified Bit Status

This instruction counts the number of "1" (ON) bits in the data of a specified device.

Instruction	Operand Type	Function						
D FNC44 BON P	S. D. n	**16-bit Instruction**	**Mnemonic**	**Operation Condition**		**32-bit Instruction**	**Mnemonic**	**Operation Condition**
		7 steps	BON	Continuous Operation		13 steps	DBON	Continuous Operation
			BONP	Pulse (Single) Operation			DBONP	Pulse (Single) Operation

Operand number			
Operand number	S.	Word device number storing the source data **Applicable devices**: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify	BIN16/32-bit
	D.	Bit device number to be driven **Applicable devices**: Y, M, S, D □.b, modify	BIN16/32-bit
	n	Bit position to be checked [n: 0 to 15 (16-bit instruction), 0 to 31 (32-bit instruction)] **Applicable devices**: D, R, K, H	BIN16/32-bit

Instruction Explanation

1. 16-bit operation (BON,BONP)

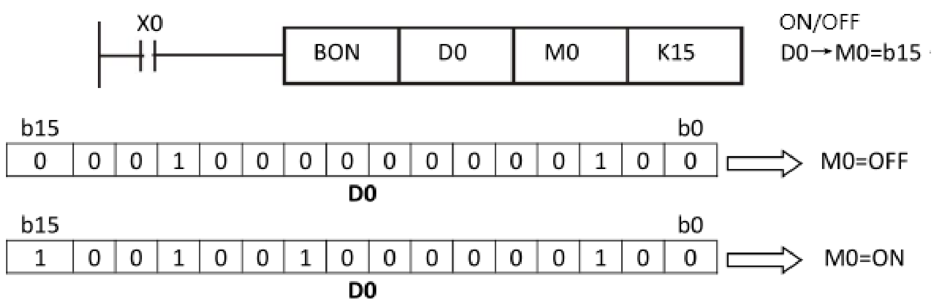
The status (ON or OFF) of the bit "n" in **S.** is output to **D.**.

2. 32-bit operation (DBON,DBONP)

The status (ON or OFF) of the bit "n" in **[S.+1, S.]** is output to **D.**

- When a constant (K) is specified as the transfer source **S.** or **[S.+1, S.]**, it is automatically converted into the BIN format.
- Note: When D and R are specified as n of 32-bit instruction, the 32-bit value of [n+1,n] will take effect. For example: When DBON D0 M0 R0, then n=[R1, R0]。

Program Example



- ◆ When X0=ON, if the 15th bit of D0 is "1", M0=ON; if it is "0", M0=OFF.
- ◆ When X0 =OFF, M0 still keep the previous state.

10.6 MEAN/Mean

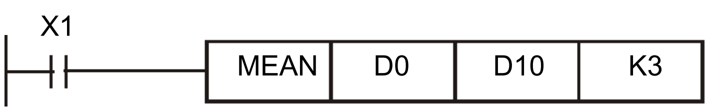
This instruction obtains the mean value of data.

Instruction	Operand Type	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
D FNC45 MEAN P	S. D. n	7 Steps	MEAN	Continuous Operation	13 steps	DMEAN	Continuous Operation
			MEANP	Pulse (Single) Operation		DMEANP	Pulse (Single) Operation

Operand number			
	S.	Head word device number storing data to be averaged **Applicable devices**: KnX, KnY, KnM, KnS, T, C, D, R, modify	BIN16/32-bit
	D.	Word device number storing the mean value result **Applicable devices**: KnY, KnM, KnS, T, C, D, R, V, Z, modify	BIN16/32-bit
	n	Number of data to be averaged (n=1~64) **Applicable devices**: D, R, K, H	BIN16/32-bit

Instruction Explanation	
	1. 16-bit operation (MEAN,MEANP) The mean value of "n" 16-bit data from **S.** is stored to **D.** . **2. 32-bit operation (DMEAN,DMEANP)** The mean value of "n" 32-bit data from **[S.+1, S.]** is stored to **[D.+1, D.]**. ● The sum is obtained as algebraic sum, and divided by "n". ● The remainder is ignored.

Program Example



(D0+D1+D2)/3→ D10

(K100+K113+K125)/3→ D10=K112.......remainder=2 (ignored)

◆ When X10=ON, The data of D0, D1 and D2 are summed, divided by "3", and then stored to D10.

10.7 ANS/Timed Annunciator Set

This instruction sets a state relay as an annunciator (S900 to S999).

Instruction	Operand Type	Function						
FNC46 ANS P	S. m D.	**16-bit Instruction**	**Mnemonic**	**Operation Condition**		**32-bit Instruction**	**Mnemonic**	**Operation Condition**
		7 steps	ANS	Continuous Operation				
Operand number	S.	Timer number for evaluation time **Applicable devices**: T[T0 ~ T199], modify						BIN16-bit
	m	Evaluation time data [m = 1 to 32767 (unit: 100 ms)] **Applicable devices**: D, R, K, H						BIN16-bit
	D.	Annunciator device to be set **Applicable devices**: S[S900 ~ S999], modify						BIN16-bit

Instruction Explanation

1. 16-bit operation (ANS)

When the command input remains ON for equivalent to or longer than the evaluation time [m*100 ms, timer **S.**], **D.**is set.

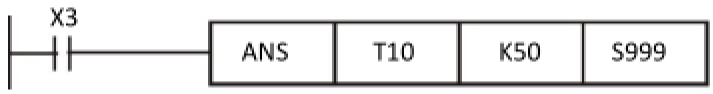
When the command input remains ON for less than the evaluation time [m*100 ms] and then turns OFF, the
current value of the timer for evaluation **S.** is reset and **D.** is not set.

Besides, When the command input turns OFF, the timer for evaluation is reset.

2. Related devices:

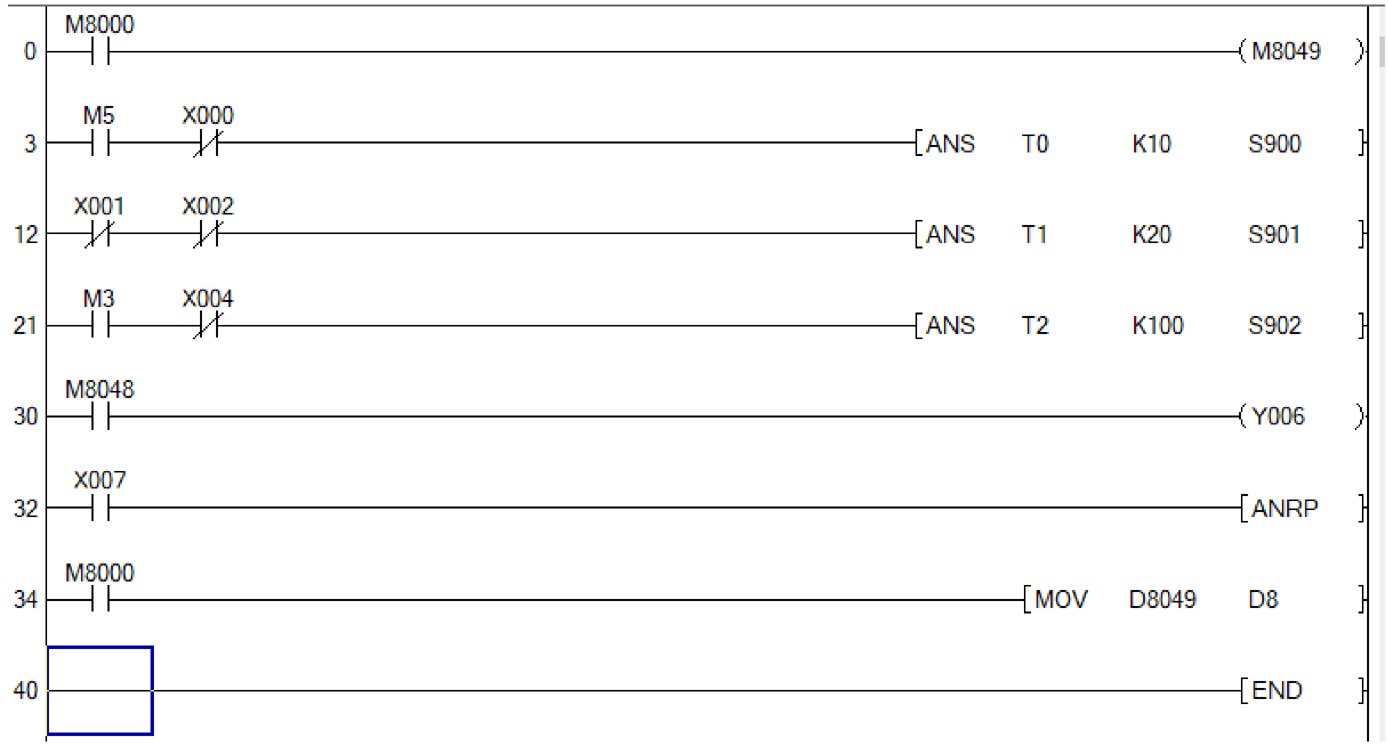
Device	Name	Description
M8049	Enable annunciator	When M8049 is set to ON, M8048 and D8049 are valid.
M8048	Annunciator ON	When M8049 is ON and one of the state relays S900 to S999 is ON, M8048 turns ON.
D8049	Smallest state relay number in ON status	Among S900 to S999, the smallest state relay number in the ON status is stored.

Program example (1)



◆ When X3=ON exceeds 5 seconds, the alarm point S999=ON, then even if X3 becomes OFF, S999 will remain ON. (But T10 will be reset to OFF, the current value = 0).

Program example (2)



◆ When M8049=ON, M8048 D8049 monitoring becomes valid
◆ When M5=ON exceeds 1second, and X0 does not turn ON within 1 second, then S900=ON.
◆ X1 and X2 do not turn ON within 2 seconds, then S901=ON.
◆ In devices with an interval of less than 10 seconds, M3=ON exceeds 10 seconds. When X4 does not turn ON during 1 cycle of the device, then S902=ON.
◆ When one among S900 to S999 turns ON, M8048 turns ON and the fault display output Y006 turns ON.
◆ Use the reset button X7 to turn off the activated state. Each time X7 turns ON, the operating state of the new number is reset in sequence, and the reset sequence starts with the smaller number.

10.8 ANR/Annunciator Reset

This instruction resets an annunciator (S900 to S999) in the ON status with the smallest number.

Instruction		Operand Type	Function						
FNC47 ANR	P	null	16-bit Instruction	Mnemonic	Operation Condition		32-bit Instruction	Mnemonic	Operation Condition
			1 step	ANR	Continuous Operation				
				ANRP	Pulse (Single) Operation				

Instruction Explanation	
	1. 16-bit operation (ANR,ANRP) When the command input turns ON, a state relay working as annunciator (S900 to S999) in the ON status is reset. If two or more state relays are ON, the state relay with the smallest number is reset. When the command input is set to ON again, the state relay with the next smallest number is reset among state relays working as annunciator (S900 to S999) in the ON status. 3. **Related devices:**

Device	Name	Description
M8049	Enable annunciator	When M8049 is set to ON, M8048 and D8049 are valid.
M8048	Annunciator ON	When M8049 is ON and one of the state relays S900 to S999 is ON, M8048 turns ON.
D8049	Smallest state relay number in ON status	Among S900 to S999, the smallest state relay number in the ON status is stored.

❖ Program example, refer to ANS(FNC 46).

10.9 SQR/BIN Square Root

This instruction obtains the square root.

The ESQR (FNC127) instruction obtains the square root in floating point operation.

Instruction	Operand Type	Function					
D **FNC48 SQR** P	**S.** **D.**	**16-bit Instruction**	**Mnemonic**	**Operation Condition**	**32-bit Instruction**	**Mnemonic**	**Operation Condition**
		5steps	SQR	Continuous Operation	9 steps	DSQR	Continuous Operation
			SQRP	Pulse (Single) Operation		DSQRP	Pulse (Single) Operation
Operand number	**S.**	Word device number storing data whose square root is obtained **Applicable devices**: D, R, K, H, modify					BIN16/32-bit
	D.	Data register number storing the square root operation result **Applicable devices**: D, R, modify					BIN16/32-bit

Instruction Explanation	**1. 16-bit operation (SQR,SQRP)** The square root of the data stored in **S.** is calculated, and stored to **D.**. **2. 32-bit operation (DSQR,DSQRP)** The square root of the data stored in **[S.+1, S.]** is calculated, and stored to **[D.+1, D.]**. ● **Operation result**: 1) The obtained square root is an integer because the decimal point is ignored. When the calculated value is ignored, M8021 (borrow flag) turns ON. 2) When the calculated value is true "0", M8020 (zero flag) turns ON.

Program Example

◆ When X10=ON, The square root of D10 is stored to D12.

such as, D0=16, Then $\sqrt{D0}$ →D12, namely D12=4.

10.10 FLT/BIN Conversion to Floating Point

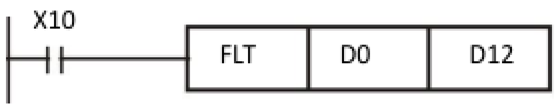
This instruction converts a binary integer into a binary floating point (real number).

Instruction	Operand Type	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
D FNC49 FLT P	S. D.	5steps	FLT	Continuous Operation	9steps	DFLT	Continuous Operation
			FLTP	Pulse (Single) Operation		DFLTP	Pulse (Single) Operation

Operand number			
Operand number	S.	Data register number storing binary integer **Applicable devices**: D, R, modify	BIN16/32-bit
	D.	Data register number storing binary floating point (real number) **Applicable devices**: D, R, modify	Real number (BIN)

Instruction Explanation	
	1. 16-bit operation (FLT,FLTP) The binary integer data of **S.** is converted into binary floating point (real number), and stored to **[D.+1, D.]**. **2. 32-bit operation (DFLT,DFLTP)** The binary integer data of **[S.+1, D.]** is converted into binary floating point (real number), and stored to**[D.+1, D.]**. ● The value of a K or H specified in each instruction for binary floating point (real number) operation is automatically converted into binary floating point (real number). It is not necessary to convert such a constant using by FLT instruction

Program Example



◆ When X10=ON, D0 (internal BIN integer) is converted into a binary floating point value and stored in D13 and D12.

11 High Speed Processing – FNC 50 to FNC 59

FNC NO.	Mnemonic	Function	Supported PLC series		
			3G PLC	2N PLC	MX2N PLC
50	REF	Refresh	★	★	★
51	REFF	Refresh and filter adjust	★		
52	MTR	Input Matrix	★		
53	HSCS	High speed counter set	★		
54	KSCR	High speed counter reset	★		
55	HSZ	High speed counter zone compare	★		
56	SPD	Speed Detection	★	★	★
57	PLSY	Pulse Y Output	★	★	★
58	PWM	Pulse Width Modulation	★	★	★
59	PLSR	Acceleration/deceleration setup	★	★	★

11.1 REF/Refresh

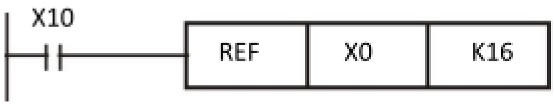
This instruction immediately outputs the latest input (X) information or the current output (Y) operation result in the middle of a sequence program.

Instruction	Operand Type	Function					
		16-bit Instruction	**Mnemonic**	**Operation Condition**	**32-bit Instruction**	**Mnemonic**	**Operation Condition**
FNC50 REF P	**D. n**	5 steps	REF REFP	Continuous Operation Pulse (Single) Operation			

Operand number	Operand	Description	Type
Operand number	**D.**	Head bit device (X ,Y) number to be refreshed **Applicable devices**: X, Y X000,X010,X020.....Up to the final input number (whose least significant digit number is "0") Y000,Y010,Y020.....Up to the final output number (whose least significant digit number is "0")	bit
	n	Number of bit devices to be refreshed (FX3U/FX3UC: multiple of 8 in the range from 8 to 256, FX3G: multiple of 8 in the range from 8 to 128) **Applicable devices**: K, H FX3U·FX3UC: K8 (H8) , K16 (H10)...K256 (H100) (which is a multiple of 8) FX3G: K8 (H8) , K16 (H10)...K256 (H100) (which is a multiple of 8)	BIN16-bit

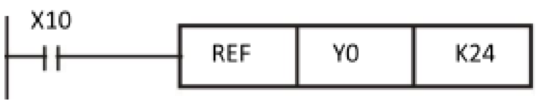
Instruction Explanation	**1. 16-bit operation (REF,REFP)** n" points are refreshed from the specified output device . ("n" must be a multiple of 8.) ● **D.**Operand number, must specify X0, X10, Y0, Y10... and other numbers with zero single digits. ● **n** is a multiple of 8 like K8 (H8) , K16 (H10)...K256 (H100);Any other numeric value causes an error.

Program Example1



◆ When X10=ON, X0~X7,X10~X17 The status of these 16 input points is refreshed immediately.

Program Example2



◆ When X10=ON, Y0~Y7,Y10~Y17,Y20~Y27 The status of these 24 input points is refreshed immediately.

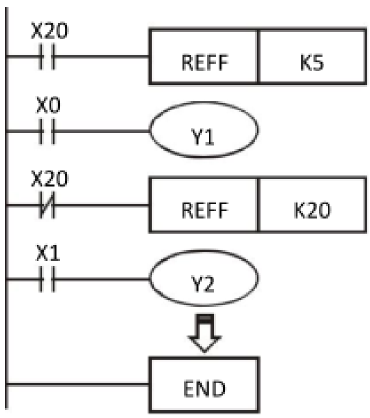
11.2 REFF/Refresh and Filter Adjust

The digital input filter time of the inputs X000 to X017*1 can be changed using this instruction or D8020.
Using this instruction, the status of inputs X000 to X017*1 can be refreshed at an arbitrary step in the program for the specified input filter time, and then transferred to the image memory.

Instruction		Operand Type	Function						
FNC51 REFF	P	n	**16-bit Instruction**	**Mnemonic**	**Operation Condition**		**32-bit Instruction**	**Mnemonic**	**Operation Condition**
			3 steps	REFF	Continuous Operation				
				REFFP	Pulse (Single) Operation				
Operand number		n	Digital input filter time [K0～K60(H0～H3C)×1ms] **Applicable devices**: D, R, K, H				BIN16-bit		

Instruction Explanation	**16-bit operation（REFF,REFFP)** 16 inputs from X000 to X017*1 in the image memory are refreshed at the digital input filter time [n*1 ms]. ● The value of the input filter, changes according to the design content of D8020 (initial value: 10ms). ● When the input turns ON "n × 1 ms" before the instruction is executed, the image memory is set to ON. When the input turns OFF "n × 1 ms" before the instruction is executed, the image memory is set to OFF. ● When the command input is ON, the REFF instruction is executed in each operation cycle. ● When the command input is OFF, the REFF instruction is not executed, and the input filter of X000 to X017*1 uses the set value of D8020 (which is the value used during input processing).

Program Example



- ◆ When PLC power =OFF→ON, When PLC power is turned from OFF→ON, The response time of the input terminals X0~X17 is determined by the content value of D8020 (default 10ms).
- ◆ When X20=ON, REFF K5 instruction is executed,
  The response time was changed to 5 ms, and it was adjusted at the next scan.
- ◆ X20=OFF 时, REFF K20 instruction is executed,
  The response time was changed to 20 ms, and it was adjusted at the next scan.

11.3 MTR/Input Matrix

This instruction reads matrix input as 8-point input* "n"-point output (transistor) in the time division method

Instruction	Operand Type	Function							
FNC52 MTR	**S.** **D1.** **D2.** **n**	**16-bit Instruction**	**Mnemonic**	**Operation Condition**		**32-bit Instruction**	**Mnemonic**	**Operation Condition**	
		9 steps	MTR	Continuous Operation					
Operand number	**S.**	Input device (X) number of matrix signal input X0,X10,X20…final input device number (Only "0" is allowed in the least significant digit of device numbers.) **Applicable devices**: X							bit
	D1.	Head device (Y) number of matrix signal output Y0,Y10,Y20…final output device number (Only "0" is allowed in the least significant digit of device numbers.) **Applicable devices**: Y							bit
	D2.	Head bit device (Y, M or S) number of ON output destination Y0,Y10,Y20…,M0,M10,M20…,S0,10,S20…final Y,M,S number (Only "0" is allowed in the least significant digit of device numbers.) **Applicable devices**: Y, M, S							bit
	n	Number of columns in matrix input (K2～K8/H2～H8) **Applicable devices**: K, H							BIN16-bit

Instruction Explanation	**16-bit operation (MTR)** An input signal of 8 points* "n" columns is controlled in the time division method using 8 inputs **S.** and "n" **D1.** transistor outputs . Each column is read in turn, and then output to **D2.** . ● For each output, the I/O processing is executed immediately in turn in interrupt at every 20 ms under consideration of the input filter response delay of 10 ms. ● 8 input points are occupied from the input device number specified in **S.** "n" output points are occupied from the output device number specified in **D1.**. When specifying the output in **D2.**, make sure that "n" output numbers specified in **D1.** does not overlap the output specified in **D2.**. ● Use the transistor output format.

Program Example

◆ When PLC RUN, The MTR instruction starts to be executed, and the status of 16 switches in the external 2 lines is read sequentially and stored in the internal relays M10~M17, M20~M27.

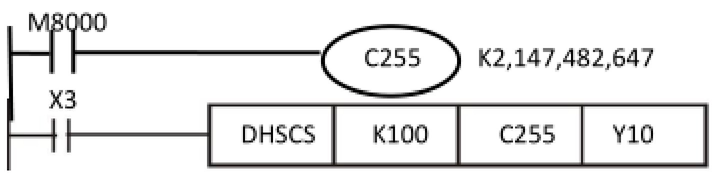
11.4 HSCS/High Speed Counter Set

This instruction compares a value counted by a high speed counter with a specified value, and immediately sets an external output (Y) if the two values are equivalent each other.

<table>
<tr><th>Instruction</th><th>Operand Type</th><th colspan="7">Function</th></tr>
<tr><td rowspan="2">D
FNC53
HSCS</td><td rowspan="2">S1.
S2.
D.</td><td>16-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td><td rowspan="2"></td><td>32-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td></tr>
<tr><td></td><td></td><td></td><td>13 steps</td><td>DHSCS</td><td>Continuous Operation</td></tr>
<tr><td rowspan="3">Operand number</td><td>S1.</td><td colspan="5">Data to be compared with the current data value of a high-speed counter or word device number.
Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, Z, K, H, modify</td><td colspan="2">BIN32-bit</td></tr>
<tr><td>S2.</td><td colspan="5">Device number of a high speed counter [C235～C255]
Applicable devices: C, modify</td><td colspan="2">BIN32-bit</td></tr>
<tr><td>D.</td><td colspan="5">Bit device number to be set to ON when the compared two values are equivalent to each other
Applicable devices: Y, M, S, D □.b[FX3U support], P[counter interrupt 6points: I010~I060]</td><td colspan="2">bit</td></tr>
</table>

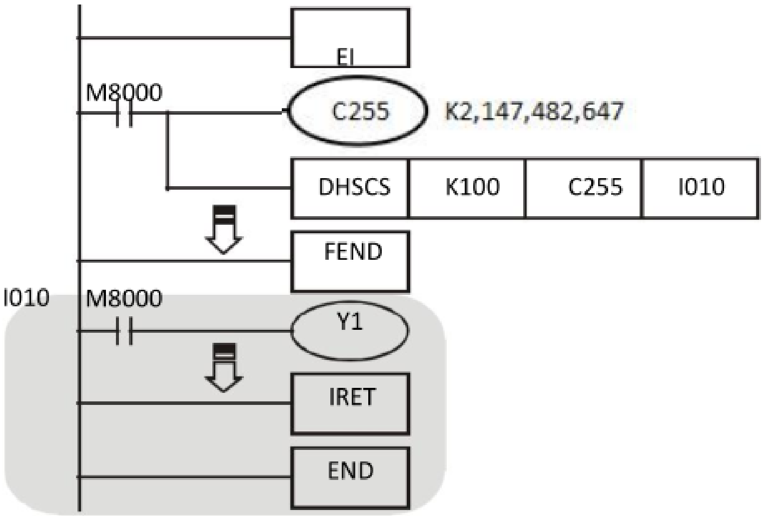
Instruction Explanation	**32-bit operation（DHSCS)** When the current value of a high speed counter (C235-C255) specified in **S2.** becomes the comparison Value **[S1.+1, S1.]**,(Comparison Value K200=199→200 or 201→200), is set to ON without regard to the operation cycle. This instruction is executed after the counting processing in the high speed counter.

Program Example



- When X3=ON, DHSCS instruction execute, the current value of the high speed counter C255 changes from "99" to "100" or from "101" to "100",Y010 is set to ON (output refresh).

Program Example



- High speed counter （only 3G series PLC support, refer to **3.11.3**):

  D. Operand number range of the DHSCS instruction can also be specified, I0□0, □=1~6, When the counter reaches the count, an interrupt occurs and the interrupt service routine is executed.
- When the current value of C255 changes from 99→100 and 101→100, the program jumps to the interrupt pointer I010 to execute the interrupt service subroutine.

11.5 HSCR/High Speed Counter Reset

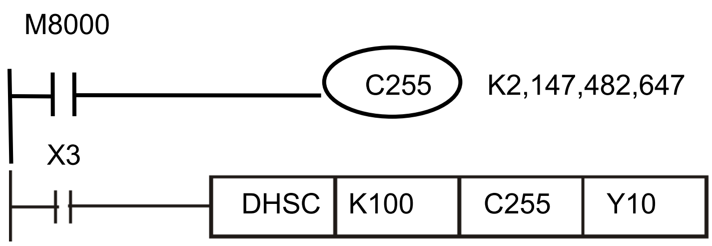
This instruction compares the value counted by a high speed counter with a specified value at each count,and immediately resets an external output (Y) when both values become equivalent to each other.

Instruction	Operand Type	Function						
D FNC54 HSCR	S1. S2. D.	**16-bit Instruction**	**Mnemonic**	**Operation Condition**		**32-bit Instruction**	**Mnemonic**	**Operation Condition**
						13 步	DHSCR	Continuous Operation

Operand number			
Operand number	**S1.**	Data to be compared with the current value of a high speed counter or word device number storing the data to be compared **Applicable devices**: KnX, KnY, KnM, KnS, T, C, D, R, Z, K, H, modify	BIN32-bit
	S2.	Device number of a high speed counter [C235～C255] **Applicable devices**: C, modify	BIN32-bit
	D.	Bit device number to be reset (set to OFF) when both values become equivalent each other. **Applicable devices**: Y, M, S, D 口.b[FX3U support], C [The same counter as can be specified**S2.**]	-bit

Instruction Explanation	**32-bit operation (DHSCR)** When the current value of a high speed counter (C235-C255) specified in **S2.** becomes the comparison value **[S1.+1, S1.],** (for example, when the current value K200 changes from 199→200 or from 201→200), the bit device is reset (set to OFF) regardless of the operation cycle. In this instruction, the comparison processing is executed after the counting processing in the high speed counter. ● 8 input points are occupied from the input device number specified in **S.** ● "n" output points are occupied from the output device number specified in **D1.**. When specifying the output in **D2.**, make sure that "n" output numbers specified in **D1.** ● Use the transistor output format.

Program Example



◆ When X3=ON, DHSCS instruction execute, the current value of the high speed counter C255 changes from "99" to "100" or from "101" to "100",Y010 is set to ON (output refresh).

11.6 HSZ/High Speed Counter Zone Compare

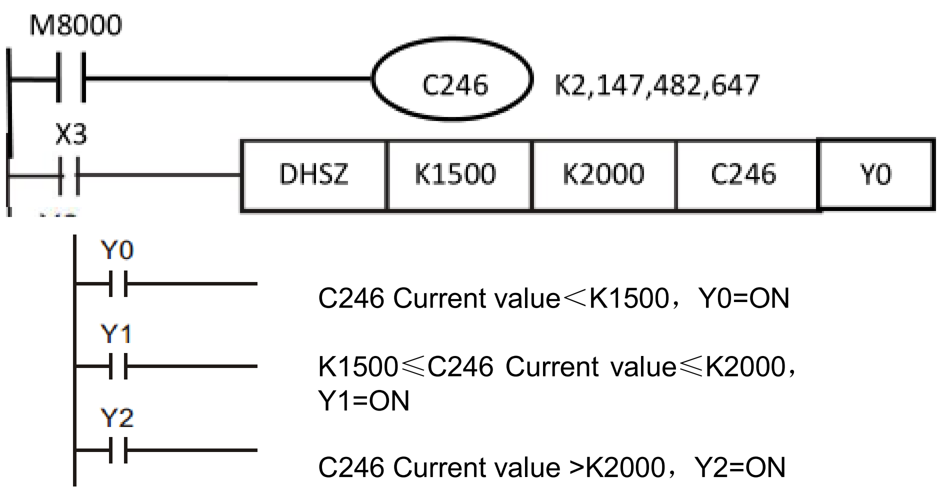
This instruction compares the current value of a high speed counter with two values (one zone), and outputs the comparison result to three bit devices (refresh).

Instruction	Operand Type	Function						
D **FNC55 HSZ**	**S1. S2. S. D.**	**16-bit Instruction**	**Mnemonic**	**Operation Condition**		**32-bit Instruction**	**Mnemonic**	**Operation Condition**
						17 steps	DHSZ	Continuous Operation
Operand number	**S1.**	Data to be compared with the current value of a high speed counter or word device number storing data to be compared (comparison value 1) **Applicable devices**: KnX, KnY, KnM, KnS, T, C, D, R, Z, K, H, modify					BIN32-bit	
	S2.	Data to be compared with the current value of a high speed counter or word device number storing data to be compared (comparison value 2) **Applicable devices**: KnX, KnY, KnM, KnS, T, C, D, R, Z, K, H, modify					BIN32-bit	

	S.	Device number of a high speed counter [C235～C255] **Applicable devices**: C, modify	BIN32-bit
	D.	Head bit device number to which the comparison result is output based on upper and lower comparison values **Applicable devices**: Y, M, S, D □.b[FX3U support], modify	-bit

Instruction Explanation	**32-bit operation（DHSZ)** The current value of a high speed counter (C235～C255) specified in is compared with two comparison points (comparison value 1 **S1**.and comparison value 2 **S2.**). Based on the comparison result, "smaller than the lower comparison value", "inside the comparison zone" or "larger than the upper comparison value",one among , **D.**,**D.+1**,**D.+2** is set to ON regardless of the operation cycle. In this instruction, the comparison processing is executed after the count processing in the high speed counter. ● comparison value 1 **S1** ≦ comparison value 2 **S2.**。 ● DHSZ instruction outputs the comparison result only when the count pulse is input. Even if the current value of C23 is 0, Y010 will remain OFF when start.

Program Example



- ◆ If the designated device is Y0, it will automatically occupy Y0~Y2.
- ◆ When the DHSZ instruction is executed, when the high-speed counter C246 has a count input, one of the upper and lower limits Y0~Y2 is On.

11.7 SPD/Speed Detection

This instruction counts the input pulse for a specified period of time as interrupt input.

Instruction	Operand Type	Function					
D FNC56 SPD	S1. S2. D.	**16-bit Instruction**	**Mnemonic**	**Operation Condition**	**32-bit Instruction**	**Mnemonic**	**Operation Condition**
		7 steps	SPD	Continuous Operation	13 steps	DSPD	Continuous Operation
Operand number	S1.	Device number of pulse input (X) **Applicable devices**: X0~X7, modify					bit
	S2.	Time data (ms) or word device number storing the data **Applicable devices**: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify					BIN16/32-bit
	D.	Head word device number storing the pulse density data **Applicable devices**: T, C, D, R, V, Z, modify					BIN16/32-bit

Instruction Explanation

1. 16-bit operation (SPD)

The input pulse **S1.** is counted only for **S2.*** 1 ms. The measured value is stored in **D.**, the present value is stored in **D.+1**, and the remaining time is stored in **D.+2**(ms).

By repeating this operation, the measured value **D.** will store the pulse density (which is proportional to the rotation speed).

2. 32-bit operation (DSPD)

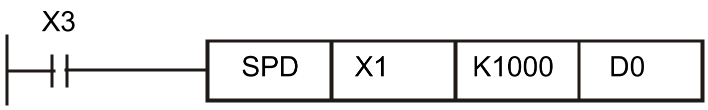
The input pulse **S1.** is counted only for **[S2.+1, S.]*** 1 ms. The measured value is stored in **[D.+1, D.]**, the present value is stored in **[D.+3, D.+2]**, and the remaining time is stored in **[D.+5, D.+4]**(ms).

By repeating this operation, the measured value [+1,] will store the pulse density (which is proportional to the rotation speed).

- An input device X000 to X007 specified as cannot overlap the following functions or instructions:
  - High speed counter
  - Input interrupt
  - Pulse catch
  - DSZR instruction
  - DVIT instruction
- - ZRN instruction ,Only one instruction can be used per input point.
- **Occupied devices**
  1) 16-bit operation: 3devices are occupied from a device specified in **D.**
  2) 32-bit operation: 6devices are occupied from a device specified in **D.**
- The value of the measured value **D.** is proportional to the speed as follows

	$N=\dfrac{60(\mathbf{D.})}{nt}\times 10^3\ (r/min)$ N: Rotating speed. n: The number of pulses produced by one rotation of the rotating equipment. t: The detection time specified for S2.(ms)

Program Example



- WhenX3=ON, D1 calculates the high-speed pulse input by X1. After 1000ms, the calculation stops automatically, and the result is stored in D0.
- 1000ms timing, is over, the content of D1 is cleared to 0, and when X7 turns ON again, D1 accepts the count again.

11.8 PLSY/Pulse Y Output

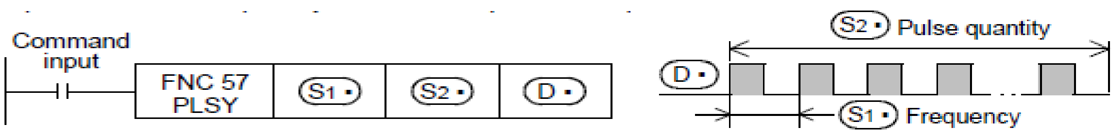
This instruction generates a pulse signal.

Instruction	Operand Type	Function						
D FNC57 PLSY	S1. S2. D.	**16-bit Instruction**	**Mnemonic**	**Operation Condition**		**32-bit Instruction**	**Mnemonic**	**Operation Condition**
		7 steps	PLSY	Continuous Operation		13 steps	DPLSY	Continuous Operation
Operand number	**S1.**	Output pulse frequency **Applicable devices**: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify						BIN16/32-bit
	S2.	Number of output pulses **Applicable devices**: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify						BIN16/32-bit
	D.	Device number (Y) from which pulses are output **Applicable devices**: Y[transistor pulse output, FX3G: Y0~Y7；FX2N: Y0,Y1,Y6,Y7,Y10(optional)], modify						bit

Instruction Explanation

1. 16-bit operation(PLSY)

A pulse train of frequency **S1.** is output in the quantity **S2.** from the output Y **D.**



2. 32-bit operation(DPLSY)

A pulse train at the frequency **[S1+1., S1]** is output by the quantity **[S2+1., S2]** from the output Y **D.**

- **S1.,[S1.+1, S1.]**: Specify frequency (Hz), Allowable setting range: 10 ~ 200,000(Hz)
- **S2.,[S2.+1, S2.]**: Specify the generated pulse quantity(PLS), Allowable setting range: 1 ~ 2,147,483,647(PLS)
- **D.:** Specify the output (Y) number from which pulses are output, Allowable setting range: 3G: Y0~Y7；2N: Y0~Y3,Y10.
- Other special soft component, please refer to
- 2N series:

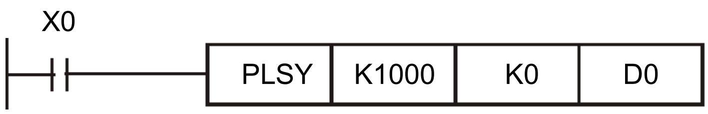
	Y0	Y1	Y6	Y7	Y10
Send end flag	M8029	M8029	M8029	M8029	M8029
Position pulse(32bit)	D8140 D8141	D8142 D8143	D8150 D8151	D8152 D8153	D8154 D8155
accelerate / decelerate time	D8148	D8148	D8148	D8148	D8148

Pulse stop bit	M8145	M8146	M8155	M8156	M8159
Pulse output busy flag	M8147	M8148	M8157	M8158	M8161

- 3G series:

Pulse point / Function Description	Y0	Y1	Y2	Y3	Y4	**Y5**	**Y6**	**Y7**
Send end flag	M8029	M8029	M8029	M8029	M8029	M8029	M8029	M8029
Pulse operation monitoring	M8340	M8350	M8360	M8370	M8151	M8152	M8153	M8154
Position pulse(32bit)	D8340 D8341	D8350 D8351	D8360 D8361	D8370 D8371	D8140 D8141	D8142 D8143	D8144 D8145	D8160 D8161
accelerate / decelerate time	D8348 D8349	D8358、 D8359	D8368、 D8369	D8378、 D8379	D8148	D8148	D8148	D8148
Pulse stop bit	M8349	M8359	M8369	M8379	M8450	M8451	M8452	M8453
Maximum speed	D8343 D8344	D8353 D8354	D8363 D8364	D8373 D8374	D8146 D8147	D8146 D8147	D8146 D8147	D8146 D8147

Program Example



◆ If X0=ON, generate 1KHz frequency pulse without any limitation(**S2.** is set to K0, pulses are output without any limitation.)output from Y0.

11.9 PWM/Pulse Width Modulation

This instruction outputs pulses with a specified period and ON duration.

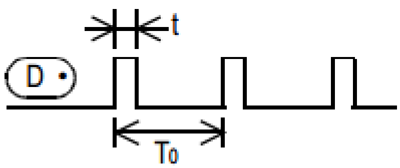
Instruction	Operand Type	Function					
FNC58 PWM	S1. S2. D.	**16-bit Instruction**	**Mnemonic**	**Operation Condition**	**32-bit Instruction**	**Mnemonic**	**Operation Condition**
		7 steps	PWM	Continuous Operation	——	——	

Operand number			
Operand number	**S1.**	Output pulse width (ms) **Applicable devices**: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify	BIN16-bit
	S2.	Period (ms) **Applicable devices**: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify	BIN16-bit
	D.	Device number (Y) from which pulses are to be output **Applicable devices**: Y[Transistor pulse output, FX3G: Y0~Y7；FX2N: Y0~Y3,Y10(Optional)], modify	-bit

Instruction Explanation

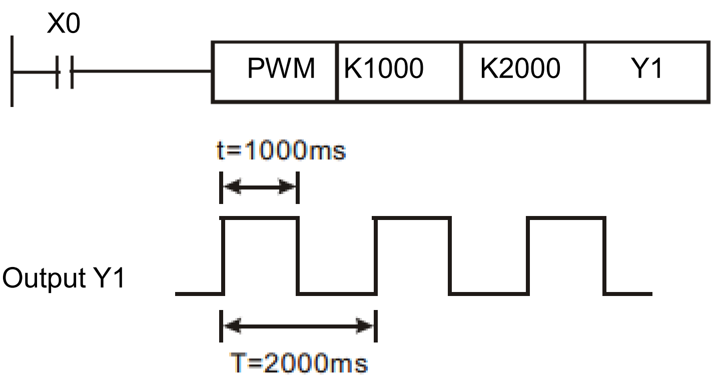
1. 16-bit operation(PWM)

Pulses whose ON pulse width is **S1.** ms are output in periods of **S2.** ms



- Specify the pulse width "t" in **S1.**,Allowable setting range:: 0~32767ms
- Specify the period "T0" in **S2.**,Allowable setting range:: 1~32767ms
- Specify the output (Y) number from which pulses are to be output in **D.**.
- Please note, pulse width **S1.**≤period **S2.**。

Program Example



- ◆ If X0=ON, Y1 output above pulse, When X0=OFF, Y1 output also change to OFF.

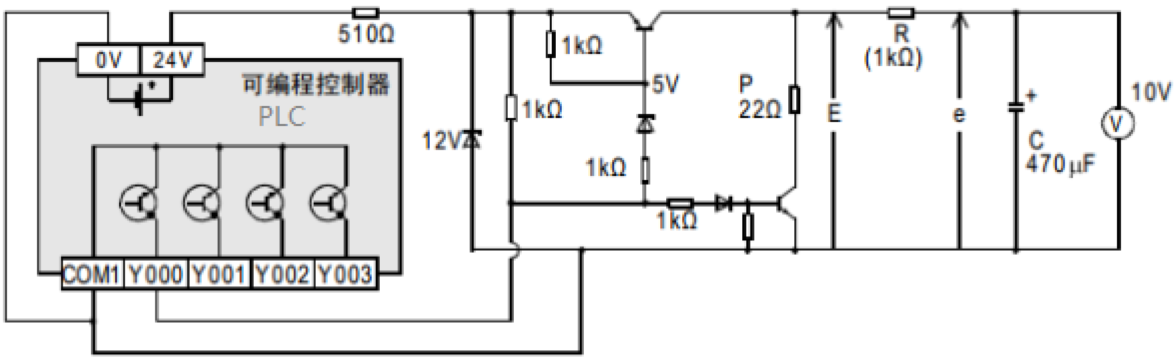
Examples of smoothing circuits::

R >> P

t = R(KΩ)* C(μF) = 470ms >>T0

The time constant τ of the filter is extremely large ,compared to the pulse period T0.

The fluctuation value Δe in the average output current e is approximately $\frac{\Delta e}{e} \leq \frac{T0}{\tau}$



Extra Description

1. For Coolmay PLC, PWM pulse width usage pls refer to below:

Conventional PWM:

1) Only Support 2channels Y0,Y1(please select transistor MT output);

2) There is no limit to the pulse width and pulse period, both in milliseconds (ms).

Special customized PWM——as Analog output

The following parameters are required for model selection:

1) the controller type and the output voltage of the required PWM;

2) the output frequency of the required PWM;

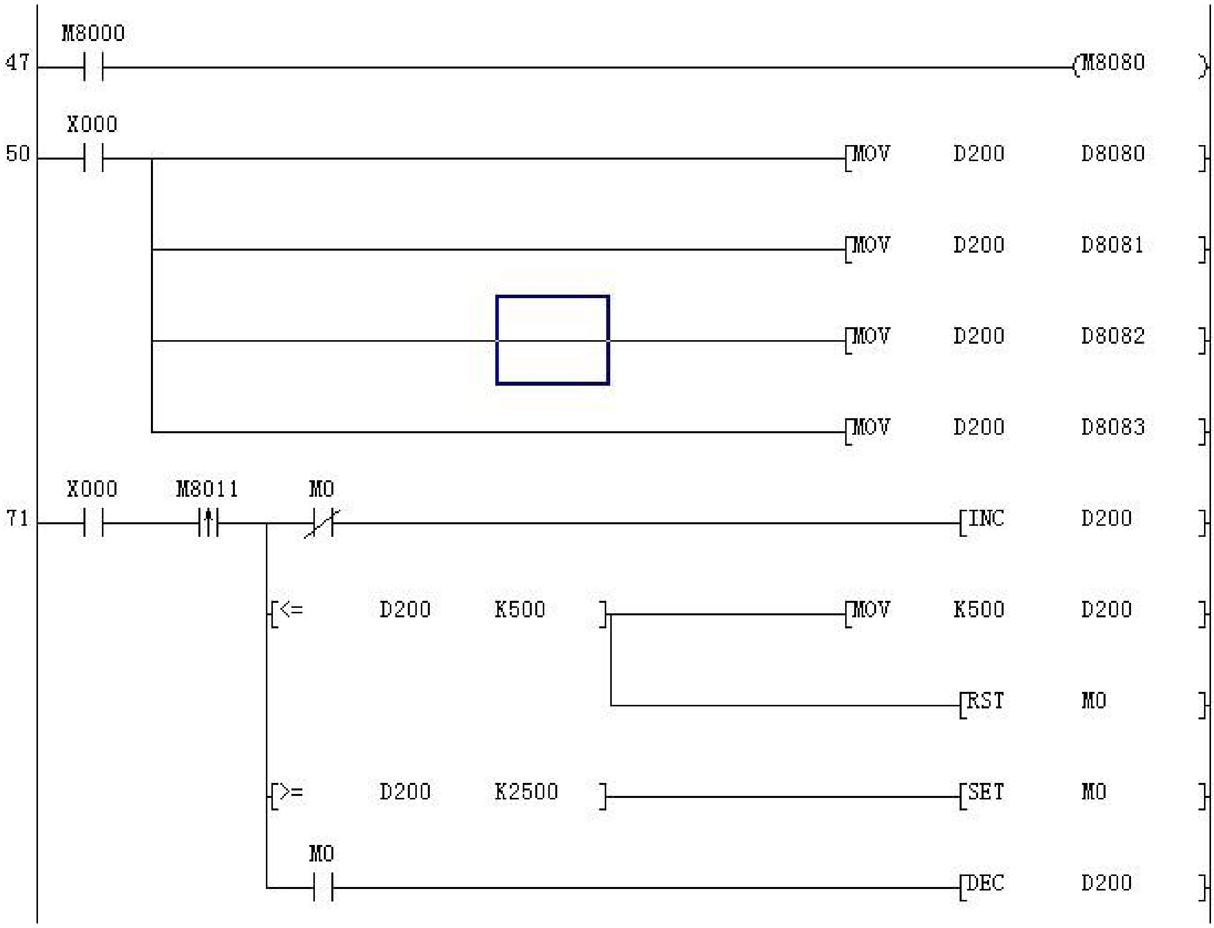
3) pulse width range;

4) to customize to microsecond (μs) level or millisecond (ms) level? [default is millisecond (ms) level];

5) the digital range corresponding to the controller action range;

6) Confirm the numbers of customized PWM, up to 8 PWM. (depending on analog outputs that customer make).

Program, refer to analog output.



1) M8080 , the start contact of the analog DA0-DA3 output function. When it is set to ON, the analog DA0-DA3 can be output.
2) M8084 is the start contact for the analog DA4-DA7 output function. When it is set to ON, the analog DA4-DA7 can be output.

2. For Coolmay PLC, PWM pulse width usage in 3G series pls refer to below:

Conventional PWM:

1) Support 8channels Y0-Y7 (please select transistor MT output);

2) There is no limit to the pulse width and pulse period, both in milliseconds (ms).

Special customized PWM——as Analog output

The following parameters are required for model selection:

1) the output voltage of the required PWM;

2) the output frequency of the required PWM;

3) Confirm the numbers of customized PWM, up to 8 PWM. (depending on analog outputs that customer make).

4) Whether the customized PWM coexists with other analog. (If the product is separately equipped with analog, the analog output terminals DA0~DA3 are a group, and DA4~DA7 are a group. When custom PWM of 3G series products, Only when the output frequency is 21KHz,it can be used with other analog group.).

Special customized PWM -- Output frequency setting

When special customize PWM,don't need to use the PWM instruction. You only need to set the special register and then turn on the hardware.

The special registers used for each analog, check below table:

Analog output address	DA0	DA1	DA2	DA3	DA4	DA5	DA6	DA7
Duty cycle setting	D8050	D8051	D8052	D8053	D8054	D8055	D8056	D8057
PWM division factor setting	D8268	D8268	D8268	D8268	D8278	D8278	D8278	D8278

D8050-D8057: 0~32760

D8268-D8278: 840~16800000 (32 bits)；

D8050-D8057 ≦ D8268 and D8278

When D8268 and D8278are powered on, the default setting is 4000, No data keeping when power off, and program assignment is required when using.

PWM output frequency setting, For example:

Main frequency 84MHz, namely 84000000

When D8268=4000, PWM output frequency=84MHz/4000=21KHz, D8050 adjustable range is 0-4000, output duty cycle 0~100%.

When D8268=8000, PWM output frequency=84MHz/8000=10.5KHz, D8050 adjustable range is 0-8000, output duty cycle 0~100%.

If D8268=16800000, then PWM minimum output frequency=84MHz/16800000=5Hz

If D8268=840, the PWM maximum output frequency=84MHz/840=100KHz

That is, the lower the frequency, the larger the adjustable range; the higher the frequency, the smaller the adjustable range.

11.10 PLSR/Acceleration/Deceleration Setup

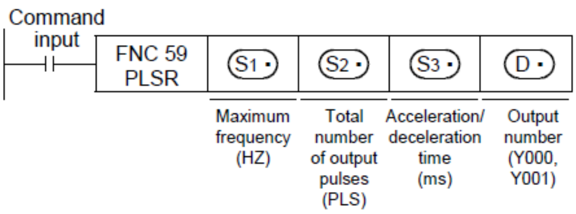
This pulse output instruction has the acceleration/deceleration function.

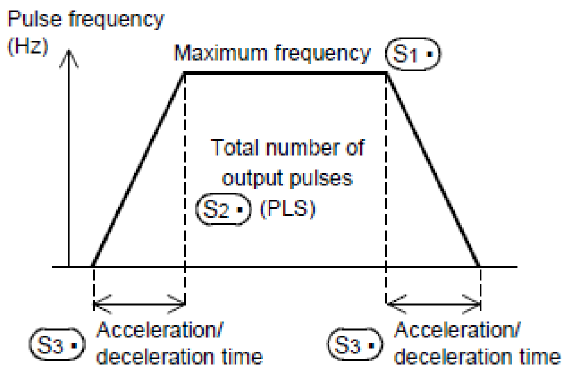
Instruction	Operand Type	Function					
D FNC59 PLSR	S1. S2. S3. D.	**16-bit Instruction**	**Mnemonic**	**Operation Condition**	**32-bit Instruction**	**Mnemonic**	**Operation Condition**
		9 steps	PLSR	Continuous Operation	17 steps	DPLSR	Continuous Operation
Operand number	**S1.**	Maximum frequency (Hz) **Applicable devices**: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify					BIN16/32-bit
	S2.	Total number of output pulses (PLS) **Applicable devices**: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify					BIN16/32-bit
	S3.	Acceleration/deceleration time (ms) **Applicable devices**: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify					BIN16/32-bit
	D.	Device number (Y) from which pulses are to be output **Applicable devices**: Y[Transistor pulse output, 3G: Y0~Y7 ; 2N: Y0,Y1,Y6,Y7,Y10(Optional)], modify					-bit

Instruction Explanation	

1. 16-bit operation(PLSR)

Pulses are output from output (Y **D.**) by the specified number **S2.** with acceleration/deceleration to the maximum frequency **S1.** over the time **S3.**(ms).





2. 32-bit operation(DPLSR)

Pulses are output from the output (Y **D.**) by the specified number **[S2.+1, S2.]** with acceleration/deceleration to the maximum frequency **[S1.+1, S1.]** for the time**[S3.+1, S3.]**(ms).

- **S1.,[S1.+1, S1.]**: Maximum frequency(Hz), Allowable setting range: 10 ~ 200,000(Hz)
- **S2.,[S2.+1, S2.]**: Total number of output pulses(PLS), Allowable setting range: 1 ~ 2,147,483,647(PLS)
- **S3.,[S3.+1, S3.]**: Acceleration/deceleration time(ms), Allowable setting range: 1 ~ 2,147,483,647(PLS)

- **D.:** Pulse output signal, Allowable setting range: 3G: Y0~Y7；2N: Y0~Y3,Y10。
- 2N series:

	Y0	Y1	Y6	Y7	Y10
Send end flag	M8029	M8029	M8029	M8029	M8029
Position pulse(32bit)	D8140 D8141	D8142 D8143	D8150 D8151	D8152 D8153	D8154 D8155
accelerate / decelerate time	D8148	D8148	D8148	D8148	D8148
Pulse stop bit	M8145	M8146	M8155	M8156	M8159
Pulse output busy flag	M8147	M8148	M8157	M8158	M8161

- 3G series:

Pulse point / Function Description	Y0	Y1	Y2	Y3	Y4	**Y5**	**Y6**	**Y7**
Send end flag	M8029	M8029	M8029	M8029	M8029	M8029	M8029	M8029
Pulse operation monitoring	M8340	M8350	M8360	M8370	M8151	M8152	M8153	M8154
Position pulse(32bit)	D8340 D8341	D8350 D8351	D8360 D8361	D8370 D8371	D8140 D8141	D8142 D8143	D8144 D8145	D8160 D8161
accelerate / decelerate time	D8348 D8349	D8358 、 D8359	D8368 、 D8369	D8378 、 D8379	D8148	D8148	D8148	D8148
Pulse stop bit	M8349	M8359	M8369	M8379	M8450	M8451	M8452	M8453
Maximum speed	D8343 D8344	D8353 D8354	D8363 D8364	D8373 D8374	D8146 D8147	D8146 D8147	D8146 D8147	D8146 D8147

Program Example

- ◆ If X0=ON, the PLSR instruction executes the maximum frequency value of pulse output of 1,000Hz, the total number of pulses output by all pulses D10 and the acceleration and deceleration time of 3,000ms, and then outputs pulses from Y0. Start to output pulses every time the frequency increases by 1,000/10 Hz. The output time of each frequency pulse is fixed at 3,000/9.
- ◆ The output is interrupted when X0 turns OFF, and the pulse count starts from 0 when it turns ON again.

12 Handy Instruction – FNC 60 to FNC 69

FNC NO.	Mnemonic	Function	Supported PLC series		
			3G PLC	2N PLC	MX2N PLC
60	IST	Initial State	★		
61	SER	Search a Data Stack	★		★
62	ABSD	Absolute drum sequencer	★		
63	INCD	Incremental drum sequencer	★		
64	TTMR	Teaching Timer	★		
65	STMR	Special Timer	★		
66	ALT	Alternate State	★	★	★
67	RAMP	Ramp Variable Value	★	★	★
68	ROTC	Rotary Table Control	★		
69	SORT	SORT Tabulated Data	★		

12.1 IST/Initial State

This instruction automatically controls the initial state and special auxiliary relays in a step ladder program.

Instruction	Operand Type	Function						
FNC 60 IST	**S. D1. D2.**	**16-bit Instruction**	**Mnemonic**	**Operation Condition**		**32-bit Instruction**	**Mnemonic**	**Operation Condition**
		7 steps	IST	Continuous Operation		——	——	

Operand number			
Operand number	**S.**	Head bit device number of the selector switch in the operation mode **Applicable devices**: X, Y, M, D □.b, modify	bit
	D1.	Smallest state relay number of practical state relays in the automatic mode[**D1.** < **D2.**] **Applicable devices**: S[S20 ~ S899、S1000 ~ S4095], modify	bit
	D2.	Largest state relay number of practical state relays in the automatic mode[**D1.** < **D2.**] **Applicable devices**: S[S20 ~ S899、S1000 ~ S4095], modify	bit

Instruction Explanation	1. 16-bit operation(IST)

1. 16-bit operation(IST)

Specify the head input in the operation mode in **S.**.

Selector switches in the operation mode occupy eight devices from the head device , and the switch functions shown in the table below are assigned to each of them.

When X020 is assigned as shown below, it is necessary to set X020 to X024 as rotary switches so that they do not turn ON at the same time.

It is not necessary to wire unused switches, but they cannot be used for any other purpose because they are occupied by IST instruction.

Source	Device number (example)	Switch function
S•	X020	Individual operation
S• + 1	X021	Return to zero point
S• + 2	X022	Stepping
S• + 3	X023	Cycle operation
S• + 4	X024	Continuous operation
S• + 5	X025	Zero return start
S• + 6	X026	Automatic start
S• + 7	X027	Stop

- While the command input is ON, the following devices are automatically switched and controlled. While the command input is OFF, the devices are not switched.

Device number	Operation function
M8040	STL transfer disable
M8041*1	Transfer start
M8042	Start pulse
M8043*1	Zero return complete
M8045	All output reset disable
M8047*2	Enable STL monitoring

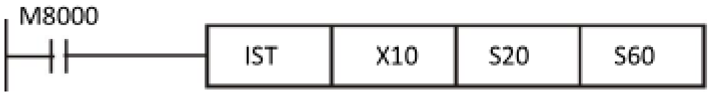
Device number	Operation function
S0	Individual operation initial state
S1	Zero return initial state
S2	Automatic operation initial state

*1. Cleared when the PLC mode is changed from RUN to STOP.

*2. Set to ON when END instruction is executed

- It is not necessary to use all switches for mode selection.
  When some switches are not used, leave the corresponding numbers in the unused status. Such numbers cannot be used for any other purpose.
- IST instruction should be programmed earlier than a series of STL circuit such as state relays S0 to S2.
- Use the state relays S10 to S19 for the zero return operation
  In the final state in the zero return operation, set M8043 to ON, and then let it be reset to OFF by itself.
- IST instruction can be used only once in a program.

Program Example



X10: Manual operation	X14: Continuous operation
X11: Return to origin	X15: Start to return to origin
X12: Step	X16: Start continuous operation
X13: One cycle	X17: Stop continuous operation

- When the IST instruction is executed, the following special auxiliary relays will be switched automatically.

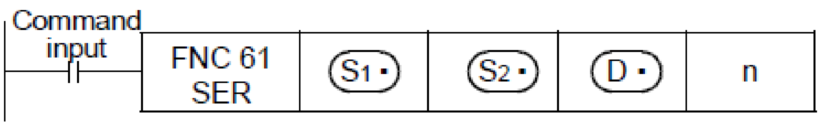
M8040: STL transfer disable	S0: Manual operation initial state step point
M8041: Transfer start	S1: Return to origin initial state step point
M8042: Start pulse	S2: Automatic operation initial state step point
M8047: Enable STL monitoring	

- When using the IST instruction, S10~S19 are used for “Return to origin”, and the step point in this state cannot be used as a general step point. When the step points of S0~S9 are used, the actions of the three state points of S0~S2 are manual use, return-to-origin use, and automatic operation. Therefore, in the program, you must write the circuit of the three state step points firstly.
- When switching to S1 (Return to origin) mode, if any point between S10~S19 is ON, there will be no action in “Return to origin”.
- When switching to S2 (automatic operation) mode, if any point of S between D1~D2 is ON, or M8043 is ON, there will be no action in automatic operation.

12.2 SER/Search a Data Stack

This instruction searches for the same data, maximum value and minimum value in a data table.

<table>
<tr><th colspan="3">Instruction</th><th>Operand Type</th><th colspan="7">Function</th></tr>
<tr><td rowspan="3">D</td><td rowspan="3">FNC61
SER</td><td rowspan="3">P</td><td rowspan="3">S1.
S2.
D.
n</td><td>16-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td><td rowspan="3"></td><td>32-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td></tr>
<tr><td rowspan="2">9 steps</td><td>SER</td><td>Continuous Operation</td><td rowspan="2">17 steps</td><td>DSER</td><td>Continuous Operation</td></tr>
<tr><td>SERP</td><td>Pulse (Single) Operation</td><td>DSERP</td><td>Pulse (Single) Operation</td></tr>
<tr><td colspan="3" rowspan="4">Operand number</td><td>S1.</td><td colspan="6">Head device number in which same data, maximum value and minimum value are searched
Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, modify</td><td>BIN16/32-bit</td></tr>
<tr><td>S2.</td><td colspan="6">Data to be searched for or device number storing data
Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, modify</td><td>BIN16/32-bit</td></tr>
<tr><td>D.</td><td colspan="6">Head device number storing number of same data, maximum value and minimum value detected by search
Applicable devices: KnY, KnM, KnS, T, C, D, R, modify</td><td>BIN16/32-bit</td></tr>
<tr><td>n</td><td colspan="6">Number of data in which same data, maximum value and minimum value are searched
[16-bit instruction: 1 to 256; 32-bit instruction: 1 to 128]
[17-Applicable devices: D, R, K, H</td><td>BIN16/32-bit</td></tr>
</table>

<table>
<tr><td rowspan="1">Instruction Explanation</td><td>
1. 16-bit operation(SER,SERP)
In "n" data starting from S1., same data as S2. is searched, and the search result is stored to D. to D.+4.


2. 32-bit operation(DSER,DSERP)
In "n" data starting from [S1.+1, S1.], same data as [S2.+1, S2.] is searched, and the search result is stored to[D.+1, D.] to [D.+9, D.+8].

● Contents of searched data and the search result
a) When same data was detected
In 5 32-bit devices starting from D. or [D.+1, D.] store the number of same data, first position, last position, maximum value position and minimum value position.
</td></tr>
</table>

b) When same data was not detected

In 5 32-bit devices starting from **D.** or **[D.+1, D.]** store the number of same data, first position, last position, maximum value position and minimum value position.

However, "0" is stored in 3 devices starting from **[D.+1, D.]** (which store the number of same data, first position and last position).

- Comparison of values,It is executed algebraically. (-10<2)
- When there are two or more maximum or minimum values in the searched data, the last position of the max/min is stored respectively.
- When this instruction is driven, the following number of devices are occupied for storing the search result **D.**:

1) 16-bit operation: occupy **[D.,D.+1,D.+2,D.+3,D.+4]** 5 points.

2) 32-bit operation: occupy **[D.+1, D.],[D.+3, D.+2],[D.+5, D.+4],[D.+7, D.+6],[D.+9, D.+8]** 10 points.

- Note: When D and R are designated as n of a 32-bit operation, the 32-bit value of [n+1,n] becomes effective.

For example: DSER D0 D100 D200 R0, then n=[R1, R0]。

Program Example

n=K1

S1.	**S1.** value	Comparison data **S2.**	**Data position**	Result
D10	K88	D0=K100	0	
D11	K100		1	equal
D12	K110		2	
D13	K150		3	
D14	K100		4	equal
D15	K300		5	
D16	K100		6	equal
D17	K5		7	Minimum
D18	K100		8	equal
D19	K500		9	Maximum

D.	D. Content value	Description
D50	4	Number of same data
D51	1	Same data position (first position)
D52	8	Same data position (last position)
D53	7	Minimum value position
D54	9	Maximum value position

- If X0=ON, the data block composed of D10~D19 is compared with D0, and the result is stored in D50~D52. When the equal value does not exist, the contents of D50~D52 are all 0.
- The minimum number of all comparison data is recorded in D53, and the maximum value is recorded in D54. When there is more than one minimum and maximum value, the one with the largest number is recorded.

12.3 ABSD/Absolute Drum Sequencer

This instruction creates many output patterns corresponding to the current value of a counter

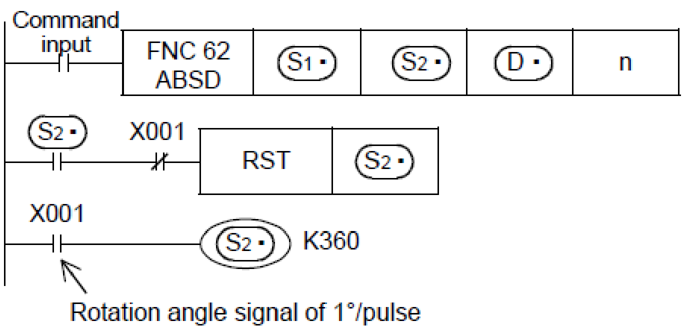
Instruction	Operand Type	Function					
D FNC62 ABSD P	S1. S2. D. n	**16-bit Instruction**	**Mnemonic**	**Operation Condition**	**32-bit Instruction**	**Mnemonic**	**Operation Condition**
		9 steps	ABSD	Continuous Operation	17 steps	DABSD	Continuous Operation
Operand number	**S1.**	Head device number storing the data table (with rising and faling point data) **Applicable devices**: KnX, KnY, KnM, KnS, T, C, D, R, modify	BIN16/32-bit				
	S2.	Counter number for monitoring the current value compared with the data table **Applicable devices**: C, modify	BIN16/32-bit				
	D.	Head bit device number to be output **Applicable devices**: Y, M, S, D □.b, modify	-bit				
	n	Number of lines in the table and the number of output bit devices [1≤n≤64] **Applicable devices**: K, H	BIN16-bit				

Instruction Explanation

1. 16-bit operation(ABSD)

In this example, outputs are controlled to ON or OFF by one rotation (0 to 360°) (using the rotation angle signal of 1°/pulse).

The current value **S2.** of the counter is compared with the data table with "n" lines starting from **S1.** (which occupies "n" lines* 2points), and consecutive "n" outputs starting from are controlled to ON or OFF during one rotation.



2. 32-bit operation(DABSD)

In this example, outputs are controlled to ON or OFF by one rotation (0 to 360°) (using the rotation angle signal of 1°/pulse).

The present value **S2.** of the counter is compared with the data table having "n" lines starting from **[S1.+1, S1.]** (which occupies "n" lines*2 points), and consecutive "n" outputs starting from **D.** are controlled to ON or OFF during one rotation.

- Specifying a high speed counter (C235 ~ C255)

In DABSD instruction, a high seed counter can be specified as **S2.** , however, the output pattern contains response delay caused by the scan cycle with regard to the current value of a counter.

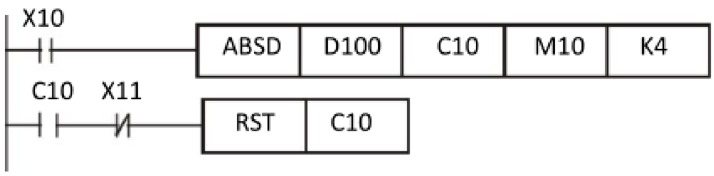
- When specifying digits of a bit device as **S1.,**

  1) Device number: Specify a multiple of 16 (0, 16, 32, 64 ...).

  2) Digital number: - ABSD(16-bit operation),Only K4 is available.

  - DABSD(32-bit operation),Only K8 is available.

Program Example

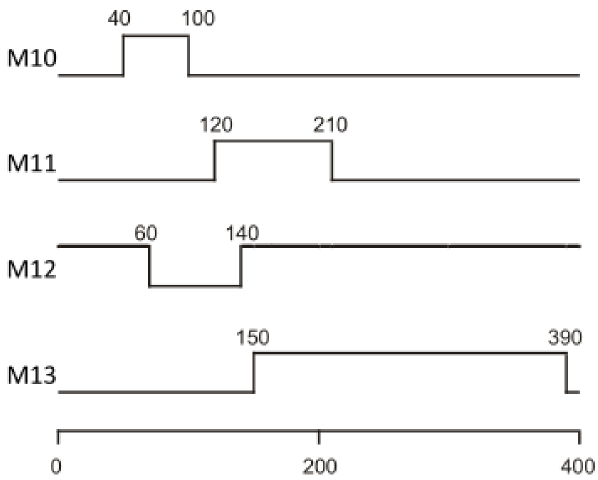


- M10~M13. When X10=ON, the current value of the counter C10 is compared with the upper and lower limits of 4 groups such as D100~D107, and the results are reflected in M10~M13 respectively.
- When X10=OFF, the ON/OFF status of the original M10~M13 will not change.
- The corresponding M10~M13 within the range of greater than or equal to the lower limit and less than or equal to the upper limit will be ON.

lower limit value	upper limit value	C10 current value	Output
D100=40	D101=100	40≤C10≤100	M10=ON
D102=120	D103=210	120≤C10≤210	M11=ON
D104=140	D105=170	140≤C10≤170	M12=ON
D106=150	D107=390	150≤C10≤390	M13=ON

- If the lower limit value is greater than the upper limit value, Then when it is less than the upper limit value (C10<60) or greater than the lower limit value (C10>140), M12=On.

lower limit value	upper limit value	C10 current value	Output
D100=40	D101=100	40≤C10≤100	M10=ON
D102=120	D103=210	120≤C10≤210	M11=ON
D104=140	D105=170	140≤C10≤170	M12=OFF
D106=150	D107=390	150≤C10≤390	M13=ON



12.4 INCD/ Incremental Drum Sequencer

This instruction creates many output patterns using a pair of counters.

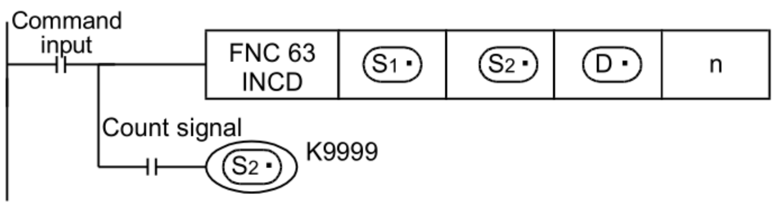
instruction	Operand type	Function					
FNC63 INCD	**S1. S2. D. n**	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
		9 steps	INCD	Continuous Operation	——		

Operand	**S1.**	Head word device number storing the set value Target device: KnX, KnY, KnM, KnS, T, C, D, R, retouch	16-bit binary
	S2.	Head number of counters whose current value is monitored Target device: C, retouch	16-bit binary
	D.	Head bit device number to be output Target device: Y, M, S, D □.b, tetouch	bit
	n	Number of output bit devices [1≤n≤64] Target device: K, H	16-bit binary

Explanation of function and operation

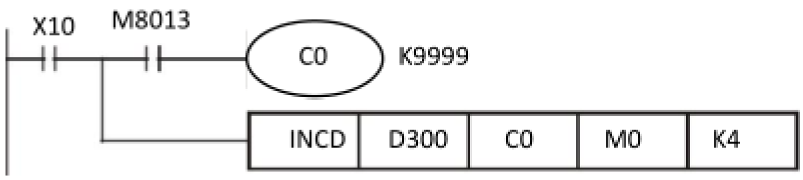
1. **16-bit operation(INCD)**

   The current value of a counter is compared with the data table having "n" lines starting from (which occupies "n" lines x 1 device). When is equivalent to the table data, the current output is reset, and the next output is set to ON. In this way, the ON/OFF status of specified outputs is controlled in turn.

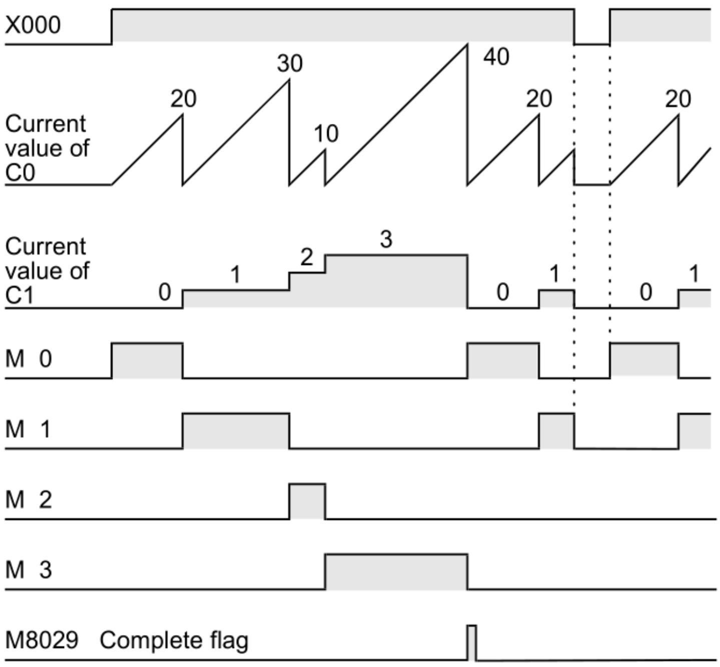


- When specifying digits of a bit device as S1 specify a multiple of 16 (0, 16, 32, 64 ...).

Program example



S1. Data value (example)	Output result**D.**(example)
D300=20	M0
D301=30	M1
D302=10	M2
D303=40	M3
...	...



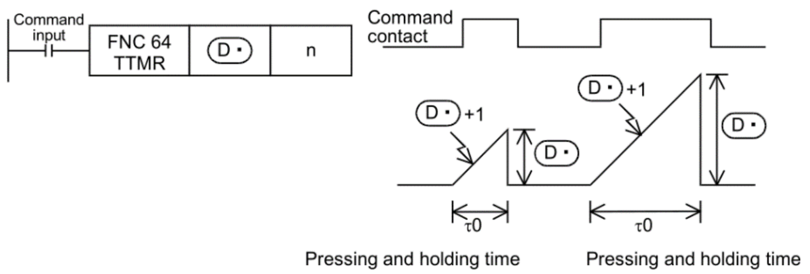
- When the command contact turns ON, the output M0 turns ON.
- When the current value of C0 reaches the comparison value D300, the output M0 is reset. "1" is added to the count value of the process counter C1, and the current value of the counter C0 is reset.
- The next output M1 turns ON.
- When the current value of C0 reaches the comparison value D301, the output M1 is reset. "1" is added to the count value of the process counter C1, and the current value of the counter C0 is reset.
- The current value is compared for up to "n (K4)" outputs in the same way (1 ≤ n ≤ 64).
- When the final process specified by "n" is finished, the execution complete flag M8029 turns ON and remains ON for one operation cycle.M8029 is used for many instructions as the instruction execution complete flag. Use M8029 as a contact just after a corresponding instruction.
- The program execution returns to the beginning, and outputs are repeated.

12.5 TTMR/ Teaching Timer

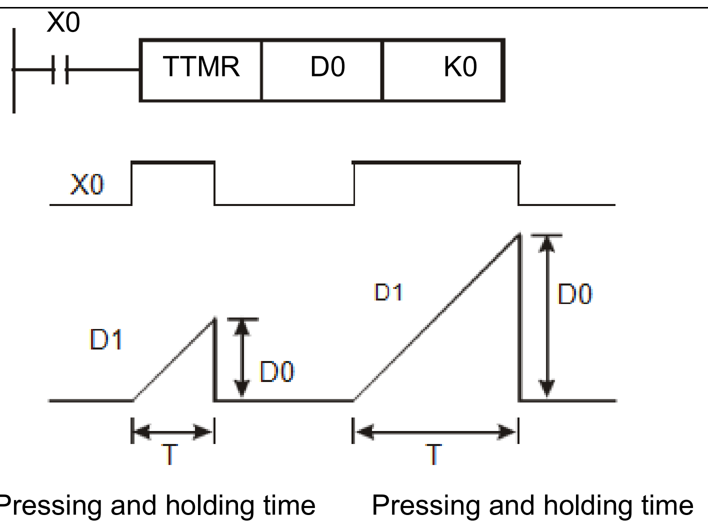
This instruction measures the period of time in which TTMR instruction is ON.

Use this instruction to adjust the set value of a timer by a pushbutton switch.

Instruction	Operand type	Function						
FNC64 TTMR	D. n	16-bit Instruction	Mnemonic	Operation Condition		32-bit Instruction	Mnemonic	Operation Condition
		5 step	TTMR	Continuous Operation		——		
Operand	D.	Device number storing the teaching data **Target device**: D, R, retouch					16-bit binary	
	n	Magnification by which the teaching data is multiplied [K0 to K2/H0 to H2] **Target device**: D, R, K, H					16-bit binary	

Explanation of function and operation	
	16-bit operation (TTMR) The period of time to press and hold the command input (pushbutton switch) is measured in 1-second units,multiplied by the magnification (10 n), and then transferred to **D.**  ● The current value **[D. +1]** of the pressing and holding time is reset, and the teaching time will not change any more. ● Two devices are occupied from a device specified as the teaching time. Make sure that these devices are not used in other controls for the machine. D. : Teaching time **[D. +1]**: Current value of the pressing and holding time

Program example



- The time the button switch X0 is held down (the ON time of X0) is stored in D1, the multiple of that time is specified by n, and the number of digits of time is stored in D0. In this way, you can use the button switch to adjust the timer setting value.
- When X0 turns OFF, the content of D1 is reset to 0, but the content of D0 has not changed.
- Assuming the ON time of X0 is T seconds, and the relationship between D0, D1 and n is shown in the following table:

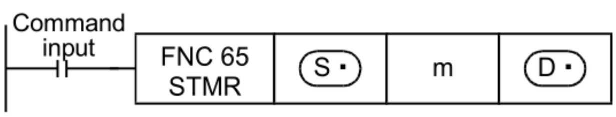
n	D0	D1(Unit: 100ms)
K0 (Unit: s)	1xT	D1=D0x10
K1 (Unit: 100ms)	10xT	D1=D0
K2 (Unit: 10ms)	100xT	D1=D0/10

12.6 STMR/Special Timer

This instruction can easily make off-delay timers, one-shot timers and flicker timers.

Instruction	Operand type	Function					
		16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
FNC65 STMR	**S. m D.**	7step	STMR	Continuous Operation	——		16-bit binary

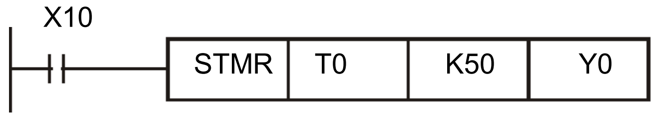
Operand	**S.**	Used timer number [T0 to T199 (100 ms timer)] **Target soft device**: D, R, retouch	16-bit binary
	m	Set value of the timer[1～32,767] **Target soft device**: D, R, K, H	16-bit binary
	D.	Head bit number to which the set value is output (Four devices are occupied.) Target device: D, R, K, H, retouch	bit

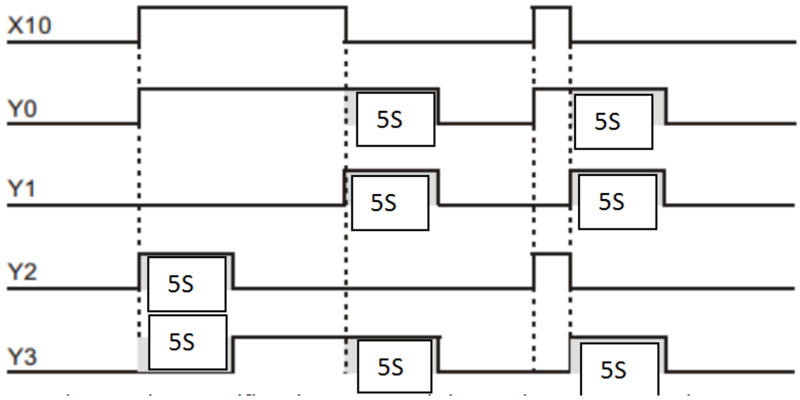
Explanation of function and operation	**16-bit operation (STMR)** The value specified in **m** is handled as the set value of a timer specified in **S.** , and output to four devices starting from **D.**.  ● The timer number specified in this instruction cannot be used in other general circuits (such as OUT instruction). ● Four devices are occupied from a device specified in **D.**

Device	Function	
		Flicker
D.	Off-delay timer	Occupied
D.+1	One-shot timer	Flicker(a contact)
D.+2	Occupied	Flicker(b contact)
D.+3	Occupied	flicker(b contact)

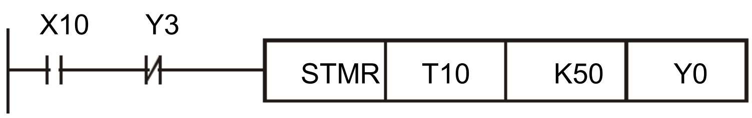
- **D.、D.+1、D.+3 will turn OFF after the set time. D.+2 and the timer S. are immediately reset.**

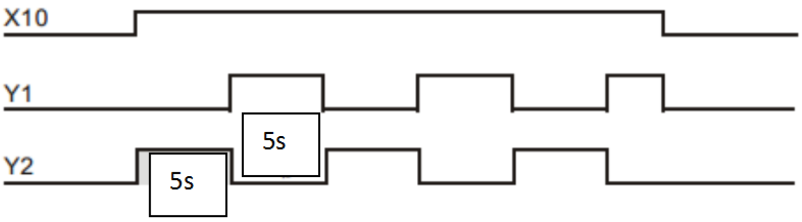
Program example





- When X10 = ON, the STMR instruction specifies timer T0, and the setting value of T0 is 5 seconds.
- Y0 is OFF Delay contact: When X10 changes from OFF to ON, Y0 = ON, and when X10 changes from ON to OFF, Y0 = OFF after a delay of 5 seconds.
- When Y1 turns from ON to OFF, Y1 = ON output for 5 seconds.
- When Y2 changes from OFF to ON, Y2 will be output once for 5 seconds.
- When Y3 changes from OFF to ON, Y3 = ON after a delay of 5 seconds, and when X10 changes from ON to OFF, Y3 = OFF after a delay of 5 seconds.
- Add a b contact of Y3 after conditional contact X10, then Y1 and Y2 can be output as a flashing circuit. When X10 turns OFF, Y0, Y1 and Y3 turn OFF, and the contents of T10 are reset to 0.





12.7 ALT/Alternate State

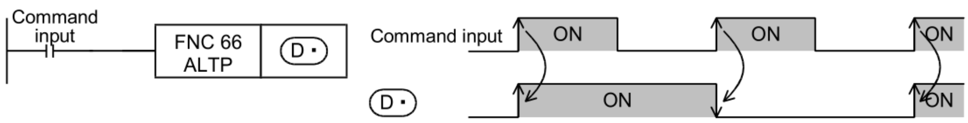
This instruction alternates a bit device (from ON to OFF or from OFF to ON) when the input turns ON.

instruction	Operand type	Function					
		16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
FNC66 ALT	**D.**	3 step	ALT	Continuous Operation	——		
P			ALTP	Pulse Operation			
Operand	**D.**	Bit device number whose output is alternated **Target device**: Y, M, S, D □.,b, retouch					bit

Explanation of function and operation

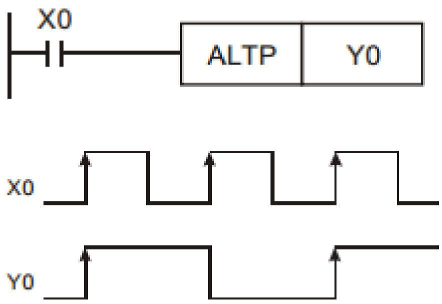
16-bit operation (ALT、ALTP)

Every time the command input turns ON from OFF, a bit device specified in **D.** is alternated (from ON to OFF or from OFF to ON).



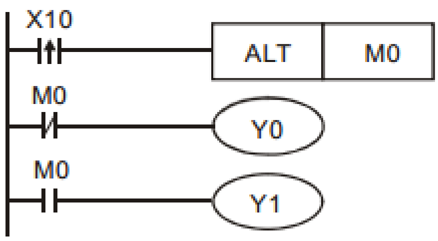
- When ALT instruction is used, a specified bit device is alternated in every operation cycle.
  To alternate a specified device by turning the command ON or OFF, please use the (pulse operation type) ALTP instruction

Program example 1



◆ When X0 goes from OFF → ON for the first time, Y0 = ON. The second time X0 goes from OFF → ON, Y0 = OFF.

Program example 2



◆ Use a single switch to control start and stop. At the beginning, M0 = OFF, so Y0 = ON, Y1 = OFF, when X10 is the first ON / OFF, M0 = ON, so Y1 = ON, Y0 = OFF, when the second ON / OFF, M0 = OFF Therefore, Y0 = ON and Y1 = OFF.

12.8 RAMP/Ramp Variable Value

This instruction obtains the data which changes between the start value (initial value) and the end value (target value) over the specified "n" times.

instruction	Operand type	Function					
		16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
FNC67 RAMP	**S1. S2. D. n**	9 Step	RAMP	Continuous Operation	——		

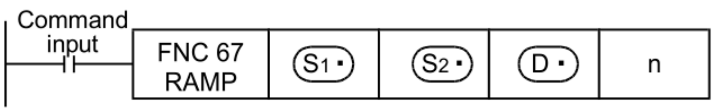
Operand	**S1.**	Device number storing the initial value of ramp Target device: D, R, retouch	16-bit binary
	S2.	Device number storing the target value of ramp Target device: D, R, retouch	16-bit binary
	D.	Device number storing the current value of ramp Target device: D, R, retouch	16-bit binary
	n	Ramp transfer time (scan) [1 to 32, 767] Target device: K, H	16-bit binary

Explanation of function and operation

16-bit operation (RAMP)

When the start value **S1.** and the end value S2. have been specified and the command input is set to ON, the value obtained by adding a value divided equally by "n" times to in every operation cycle is stored to D. .

By combining this instruction and an analog output, the cushion start/stop command can be output.



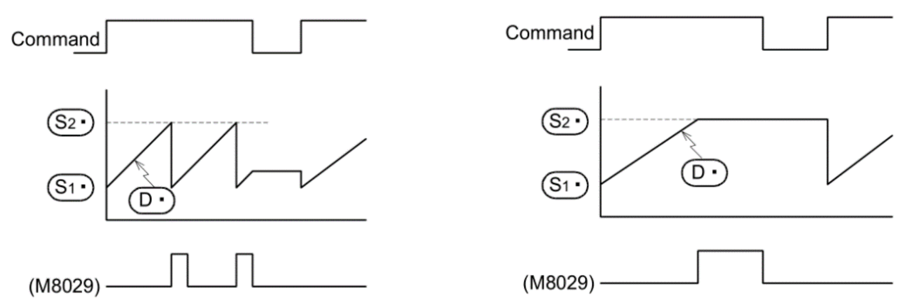
- When the power failure holding device (holding area) is specified in **D.**, the command input turns ON as it is, and when the programmable controller is set to RUN (start), first clear **D.**
- This is an instruction to find the slope. The slope is linear and has an absolute relationship with the scanning cycle. Therefore, when using this instruction, the scanning cycle must usually be fixed in advance.
- Write a prescribed scan time (which is longer than the actual scan time) to D8039 and set M8039

to ON to select the constant scan mode in the PLC.

For example, when "20 ms" is written to D8039 and "n" is set to 100, the D. value will change from **S1.** to **S2.** in 20 seconds.

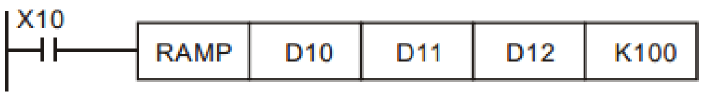
- Action of mode flag bit (M8026):

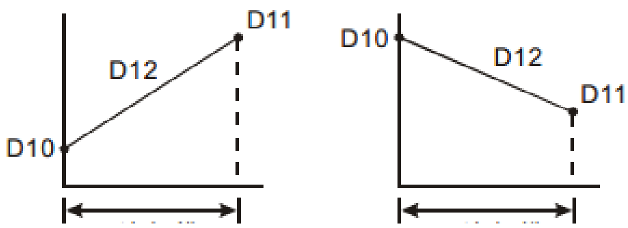
1) When M8026 is OFF   2) When M8026 is ON



Device	Name	Content
M8029	Instruction execution complete	Turns ON when **D.**=**S2.** after "n" operation cycles.
M8026 **(From RUN→STOP Clear)**	RAMP Model	Refer to the operation of the mode flag M8026 described above.

Program example





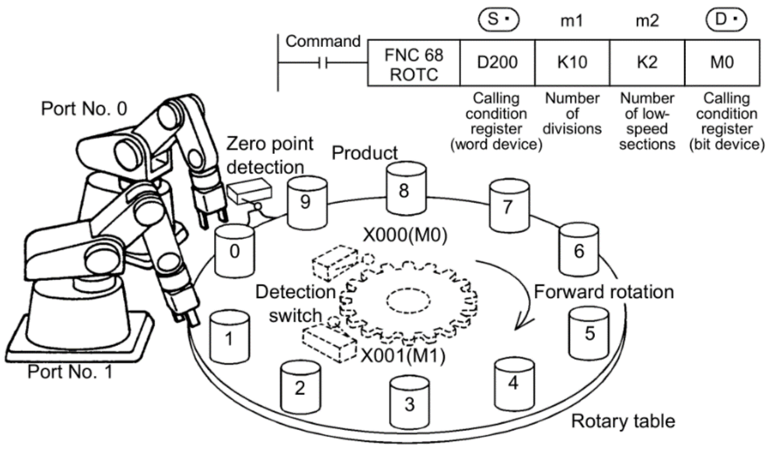
Scan n times   Scan n times
D10<D11   D10>D11
Scan times are stored in D13

- ◆ Write the setting value of the start point of the tilt signal to D10 and the setting value of the end point of the tilt signal to D11 in advance. When X10 = ON, the setting value of D10 moves towards D11 (increases), and the elapsed time (n = 100 scans) It is stored in D12, and the number of scans is stored in D13.
- ◆ In the program, first drive M1039 to ON to fix the scan cycle, and then use the MOV instruction to write the fixed scan cycle setting value to the special data register D1039. Assuming that the value is 30ms, taking the above program as an example, n = K100, the time from D10 to D11 is 3 seconds (30ms × 100).
- ◆ During the execution of the instruction, when the start signal X10 turns OFF, the execution of the instruction is stopped. When X10 turns ON again, the content of D12 is reset to 0 and recalculated.
- ◆ When M8026 = OFF, M1029 = ON, the content of D12 is reset to the setting value of D10.

12.9 ROTC/Rotary Table Control

This instruction is suitable for efficient control of the rotary table for putting/taking a product into/out of the rotary table.

<table>
<tr><th>instruction</th><th>Operand type</th><th colspan="7">Function</th></tr>
<tr><td rowspan="2">FNC68
ROTC</td><td rowspan="2">S.
m1
m2
D.</td><td>16-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td><td rowspan="2"></td><td>32-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td></tr>
<tr><td>9 steps</td><td>ROTC</td><td>Continuous Operation</td><td>——</td><td></td><td></td></tr>
<tr><td rowspan="4">Operand</td><td>S.</td><td colspan="5">Data register for counting
Target device: D, R, retouch</td><td colspan="2">16-bit binary</td></tr>
<tr><td>m1</td><td colspan="5">Number of divisions
Target device: K, H</td><td colspan="2">16-bit binary</td></tr>
<tr><td>m2</td><td colspan="5">Number of low-speed sections
Target device: K, H</td><td colspan="2">16-bit binary</td></tr>
<tr><td>D.</td><td colspan="5">Head bit device number to be driven
Target device: Y, M, S, D □.b, retouch</td><td colspan="2">16-bit binary</td></tr>
</table>

<table>
<tr><td rowspan="3">Explanation of function and operation</td><td colspan="3">16-bit operation (ROTC)
The table rotation is controlled by "m2", S.and D. so that a product can be efficiently put into or taken out of the rotary table divided into "m1" (=10) sections as shown in the figure below.


1) Register (word device) specifying the calling condition S.</td></tr>
<tr><td colspan="3">
<table>
<tr><td>S.</td><td>Sets the port No. to be called.</td><td rowspan="3">Set them in advance using a transfer instruction.</td></tr>
<tr><td>S.+1</td><td>Sets the port No. to be called.</td></tr>
<tr><td>S.+2</td><td>Sets the product No. to be called.</td></tr>
</table>

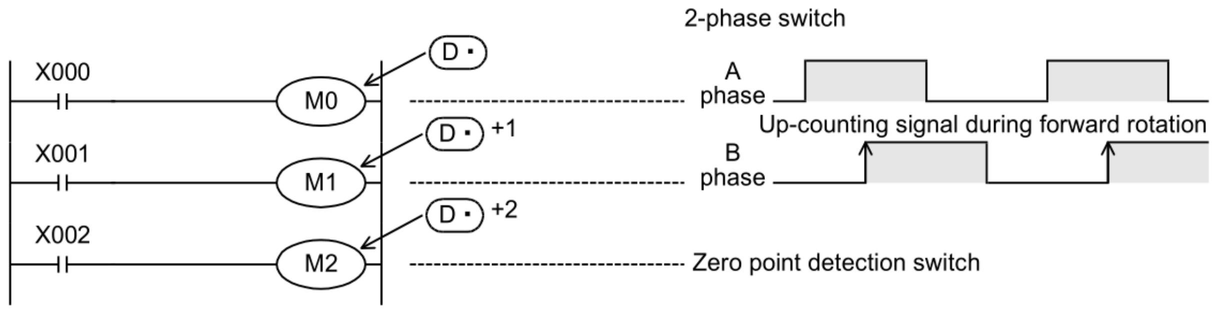
2) Register (bit device) specifying the calling condition D.</td></tr>
<tr><td colspan="3">
<table>
<tr><td>D.</td><td>A phase signal</td><td rowspan="2">Construct an internal contact circuit in advance which is driven by the input signal (X)</td></tr>
<tr><td>D.+1</td><td>B phase signal</td></tr>
</table></td></tr>
</table>

D.+2	Zero point detection signal
D.+3	Forward rotation at high speed
D.+4	Forward rotation at low speed
D.+5	Stop
D.+6	Backward rotation at low speed
D.+7	Backward rotation at high speed

Operation conditions

1) Rotation detection signal: X→**D.**

- Provide a 2-phase switch (X000 and X001) for detecting the rotation direction (forward or backward) of the table and the switch X002 which turns ON when the product No. 0 reaches the port No. 0.
- Create the sequence program shown below



2) The counter **S.** detects which number of product is located at the port No. 0.
3) Registers specifying the calling condition: +1, +2
   a) Set the port No. to be called in **S.**+1
   b) Set the product No. to be called in
4) Number of divisions m1 and number of low-speed sections m2
   Specify the number of divisions m1 of the table, and number of low-speed sections m2.

Cautions

- When the command input is set to ON and this instruction is executed, the result will be automatically output to **D.+3 ~ D.+7**

  When the command input is set to OFF, **D.+3 ~ D.+7** are set to OFF accordingly.

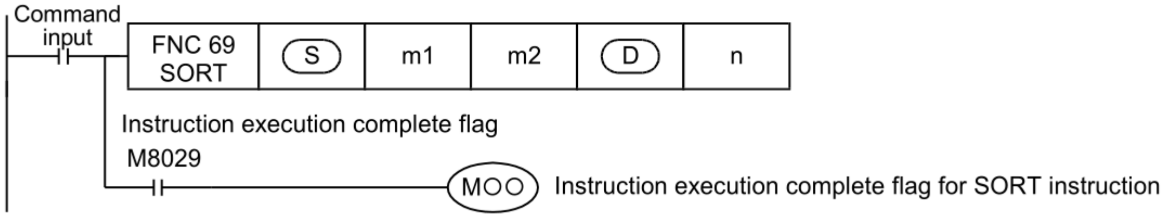
- Multiple activation of the rotation detection signal **(D. ~ D.+2)** in one division

  For example, when the rotation detection signal (**D. ~ D.+2**) is activated 10 times in one division, set a value multiplied by "10" to each division, port No. to be called and product No. to be called.
  As a result, an intermediate value of the division number can be set to a low-speed section.

12.10 SORT/ Tabulated Data

This instruction sorts a data table consisting of data (lines) and group data (columns) based on a specified group data (column) sorted by line in ascending order. This instruction stores the group data (columns) in serial devices.On the other hand, SORT2 (FNC149) instruction stores the data (lines) in serial devices facilitating the addition of data (lines), and sorts a table in either ascending or descending order.

instruction	Operand type	Function							
FNC69 SORT	**S.** **m1** **m2** **D.** **n**	16-bit Instruction	Mnemonic	Operation Condition		32-bit Instruction	Mnemonic	Operation Condition	
		11steps	SORT	Continuous Operation		——			
Operand	**S.**	Head device number storing the data table [which occupies m1 × m2 points] **Target Device**: D, R							16-bit binary
	m1	Number of data (lines) [1 ~ 32] **Target Device**: K, H							
	m2	Number of group data (columns)[1 ~ 6] **Target Device**: K, H							
	D.	Head device number storing the operation result [which occupies m1 × m2 points] **Target Device**: D, R							
	n	Column number of the group data (column) used as the basis of sorting [1 to m2] **Target Device**: D, R, K, H							

	17-bit operation (SORT) In the data table (sorting source) having (m1 × m2) points from **S.** , data lines are sorted in the ascending order based on the group data in the column No. "n", and the result is stored in the data table (sorting result) having (m1 × m2) points from **D.**  **Cautions** ● Do not change the contents of operands and data while the instruction is executed. ● Before executing the instruction again, set the command input to OFF. ● Only one instruction can be used in a program.

	● When the same device is specified in **S.** and **D.** The source data is overwritten by the data acquired by sorting. Take special care so that the contents of **S.** are not changed until execution is completed.

Program example

X0	SORT	D0	K5	K5	D50	D100

◆ When X0 = ON, specify to execute the data sorting job. When sorting is completed, M8029 = ON. Please do not change the contents of the sorted data during the execution of the instruction. If you want to reorder the data, please turn off and turn on X0 again.

◆ Example of sorted data structure

		← Number of data: m2 →				
		Data field bit				
		1	2	3	4	5
		Student ID	Chinese	English	mathematics	Physicochemical
↑: m1 ↓	1	(D0)1	(D5)90	(D10)75	(D15)66	(D20)79
	2	(D1)2	(D6)55	(D11)65	(D16)54	(D21)63
	3	(D2)3	(D7)80	(D12)98	(D17)89	(D22)90
	4	(D3)4	(D8)70	(D13)60	(D18)99	(D23)50
	5	(D4)5	(D9)95	(D14)79	(D19)75	(D24)69

◆ The sorted data when D100 = K3:

		← Number of data: m2 →				
		Data field bit				
		1	2	3	4	5
		Student ID	Chinese	English	Math	Physicoch e-mical
↑Date number: m1↓	1	(D50)4	(D55)70	(D60)60	(D65)99	(D70)50
	2	(D51)2	(D56)55	(D61)65	(D66)54	(D71)63
	3	(D52)1	(D57)90	(D62)75	(D67)66	(D72)79
	4	(D53)5	(D58)95	(D63)79	(D68)75	(D73)69
	5	(D54)3	(D59)80	(D64)98	(D69)89	(D74)90

◆ The sorted data when D100 = K5:

		← Number of data: m2 →				
		Data field bit				
		1	2	3	4	5
		Student ID	Chinese	English	Math	Physicochemical
↑Data number: m1↓	1	(D50)4	(D55)70	(D60)60	(D65)99	(D70)50
	2	(D51)2	(D56)55	(D61)65	(D66)54	(D71)63
	3	(D52)5	(D57)95	(D62)79	(D67)75	(D72)69
	4	(D53)1	(D58)90	(D63)75	(D68)66	(D73)79
	5	(D54)3	(D59)80	(D64)98	(D69)89	(D74)90

13 External I/O

FNC NO.	Mnemonic	Function	Support Model		
			3G series PLC	2N series PLC	MX2N series PLC
70	TKY	Number key input	★		
71	KHY	Hexadecimal numeric key input	★		
72	DSW	Digital switch	★		
73	SEGD	7-segment decoder	★	★	★
74	SEGL	7SEG hour and minute display	★		
75	ARWS	Arrow switch	★		
76	ASC	ASCII data input	★		
77	PR	ASCII print	★		
78	FROM	BFM read	★		Install RS485 / RS232 as MODBUS-RTU master station function, used to read / write slave station data
79	TO	BFM input	★		

13.1 TKY/Ten Key Input

This instruction sets data to timers and counters through inputs of the ten keys from "0" to "9".

instruction	Operand type	Function						
D FNC70 TKY	S. D1. D2.	16-bit Instruction	Mnemonic	Operation Condition		32-bit Instruction	Mnemonic	Operation Condition
		7steps	TKY	Continuous Operation		13 steps	DTKY	Continuous Operation

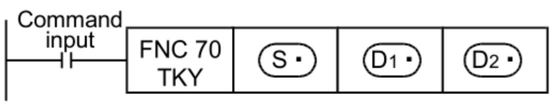
Operand	S.	Head bit device number from which one of the ten keys is input [10 devices are occupied] **Target Device**: X, Y, M, S, D. b, retouch	bit
	D1.	Word device number storing the data **Target Device**: KnY, KnM, KnS, T, C, D, R, V, Z, retouch	BIN16/32bit
	D2.	Head bit device number storing the key pressing information [11 devices are occupied] **Target Device**: X, Y, M, S, D. b, retouch	bit

Explanation of function and operation

1. 16-bit operation(TKY)

D1. stores a numeric value input **[S.～S.+9]** +9 connected to the ten keys. Output informations for key pressing and key sensing are output to **D2.～D2.**+10.

1) Numeric value **D1.**when an input value is larger than "9999", it overflows from the most significant digit.



2. 32-bit operation(DTKY)

[D1.+1, D.] store a numeric value input from**[S.～S.+9]** connected to the ten keys. Output informations for key pressing and key sensing are output to **D2.～D2. +10**

1) Numeric value **D1.**when an input value is larger than "9999", it overflows from the most significant digit.

- An input numeric **D1.** value is stored in the binary format.
- Key pressing information **[D2.～D2.+10]**
  - For the key pressing information, **D2.～D2.+9** turn ON or OFF according to the pressed keys.
  - **D2.+10** is ON while either one among "0" to "9" keys is pressed (key sensing output).
- When two or more keys are pressed at the same time, only the first key pressed is valid.
- Though the contents of **D1.** do not change, all of **D2.～D2.+10** turn OFF.

- Number of occupied device

  1) Ten bit devices are occupied from **S.** for connecting the ten keys.

  Because these devices are occupied even if the ten keys are not connected, they cannot be used for any other purpose.

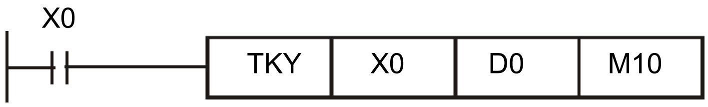
  2) Eleven bit devices are occupied from **D2.** for outputting the key pressing information.

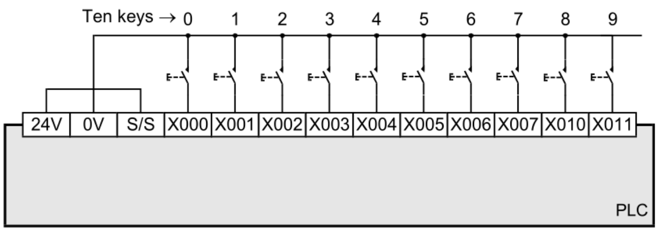
  Please be careful not to duplicate with the device used in other control of the machine.

  - **D2. ~ D2.+9**: Turn ON or OFF according to input of the ten keys "0" to "9".
  - **D2.+10**: Is ON while either one among "0" to "9" keys is pressed (key sensing output).

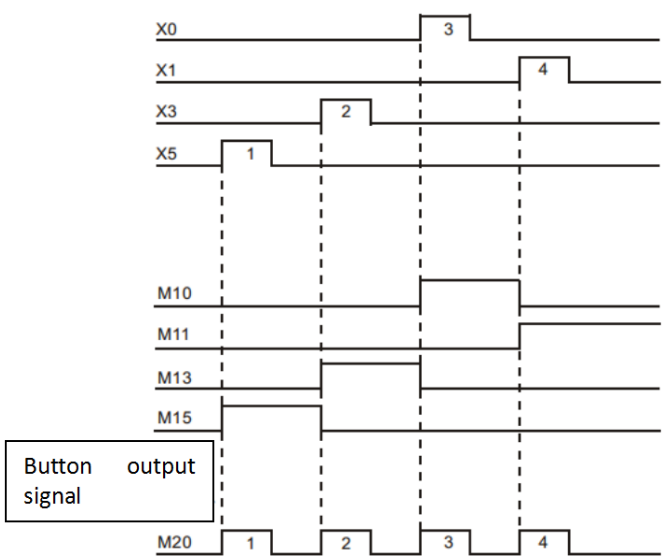
- TKY or DTKY instruction can be used only once in a program.

Program example





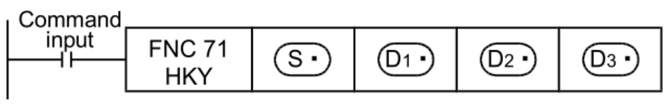
- The command specifies that the 10 input terminals starting with X0 are connected to 10 keys from 0 to 9. When X20=ON, the command is executed, and the value entered by the keyboard is stored in D0 in the form of BIN value, and the key is placed in M10~M19.
- When the ten keys are pressed in the order "[1] → [2] →[3] → [4]" shown in the figure, "5301" is stored in (D0). When an input value is larger than "9999",when the number exceeds 4 digits, it overflows from the most significant digit.
- When X2 is pressed, M12 turns ON and remains ON until another key is pressed. Other keys work in the same way.
- When any key among X0~X11 is pressed, one of M10~M19 corresponds to ON.
- When pressing a key, M20=ON.
- When conditional contact X20 turns OFF, the value before D0 does not change, but all M10~M20 turn OFF.



13.2 HKY/Hexadecimal Input

This instruction allow 16 key (0 to F) 4-digit (byte) input. Keys 0 to 9 stores numerical values, and keys A to F represent function keys. When the extension function is set to ON, hexadecimal keys 0 to F all store their corresponding numerical values.

instruction	Operand type	Function					
FNC71 HKY D	S. D1. D2. D3.	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
		9 steps	HKY	Continuous Operation	17 steps	DHKY	Continuous Operation
Operand	S.	**Head X device number to be used (Four devices occupied.)** **Target Device**: X, retouch					bit
	D1.	**Head Y device number to be used (Four devices occupied.)** **Target Device**: Y, retouch					bit
	D2.	**Device number storing the numerical input from the 16 keys** **Target Device**: T, C, D, R, V, Z, retouch					BIN16/32bit
	D3.	**Head bit device number storing the key pressing ON information(Eight devices are occupied.)** **Target Device**: Y, M, S, D. b, retouch					bit

Explanation of function and operation	**1. 16-bit operation(HKY)** Signals **[S. ~ S.+3]** and **[D1. ~ D1.+3]** connected to the 16 key input (0 to F) are scanned. When a key 0 to 9 is pressed, the corresponding numeric value is shifted into **D2.** from the least significant byte, and **D3.+7** turns ON. When a key A to F is pressed, the corresponding key press information bit **[D3. ~ D3.+5]** turns ON, and **D3.+6** turns ON. 1) When an input value **D2.、D3.+7** is larger than "9999", it overflows from the most significant digit.  **2. 32-bit operation(DHKY)** Signals **[S. ~ S.+3]** and **[D1. ~ D1.+3]** connected to the 16 key input (0 to F) are scanned. When a key 0 to 9 is pressed, the corresponding numeric value is shifted into **[D2.+1, D2.]** from the least significant byte, and **D3.+7** turns ON. When a key A to F is pressed, the corresponding key press information bit**[D3. ~ D3.+5]** turns ON. And **D3.+6** turns ON. 1) When an input value **[D2.+1、D2.]、D3.+7** is larger than "99,999,999", it overflows from the most

significant digit.

- About using the keys from **0 ~ 9** to enter values **D2.**、**D3.+7** or **[D2.+1**、**D2.]**、**D3.+7**

  - The entered value is stored in **D2.**or **[D2.+1**、**D2.]** as BIN (binary number).

  - When any key among **0~9** is pressed, **D3.+7** corresponds to ON.

- **Key pressing information for the keys A to F:**
  - Six devices starting from **D3.** corresponding to keys A to F turn ON.
  - The key sensing output **D3.+6** turns ON when any key A to F is pressed.

Key	Key pressing information	Key	Key pressing information
A	D3•	D	D3• +3
B	D3• +1	E	D3• +4
C	D3• +2	F	D3• +5

- When two or more keys are pressed at the same time, the first key pressed is valid.
- Though the contents of **D2.** do not change, to **D3.** ~ **D3.+7** turn OFF.
- Number of devices occupied

  1) Four devices are occupied from the head X device **S.** for connecting 16 keys.
  2) Four devices are occupied from the head Y device **D1.** for connecting 16 keys.
  3) Eight devices are occupied from the head device **D3.** for outputting the key pressing information.

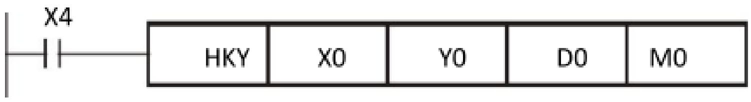
  Make sure that these devices are not used by other machine controls.

  - **D3.** ~ **D3.+5:** Key pressing information for the keys A to F
  - **D3.+6** : Key sensing output for the keys A to F
  - **D3.+7** : Key sensing output for the keys 0 to 9

- HKY or DHKY instruction can be used only once in a program.
- HKY and DHKY instructions are executed in synchronization with the operation cycle of the PLC. 8 scan cycles are required to finish reading the keys.
- Related devices:

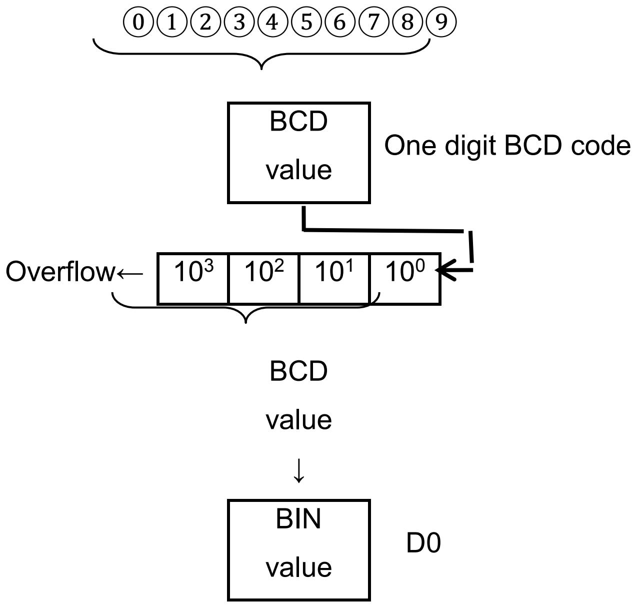
Device	Name	Content
M8029	Extension function flag	OFF: Data is being output to D1. ~ D1.+3 or the instruction is not executed yet. ON : A cycle operation of outputting data to D1. ~ D1.+3 (scan of the keys 0 to F) is completed.
M8167	nstruction execution complete flag	HKY(FNC 71)Instruction HEX data processing function OFF: Ten-keys and function keys ON: Hexadecimal keys

Program example



- The instruction specifies 4 input terminals such as X0~X3 and 4 input terminals such as Y0~Y3 to form a keyboard for scanning 16 keys. When X4=On, the instruction is executed, and the value input by the keyboard is stored in D0 in the form of BIN value, and the key is placed in M0~M7.

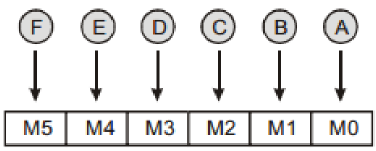
- Digital input:



- Function key input:

  1. When you press the A key, M0=On and hold, then when you press the D key again, M0 becomes OFF, M3=ON and hold.

  2. Press multiple buttons at the same time, whichever comes first is preferred.



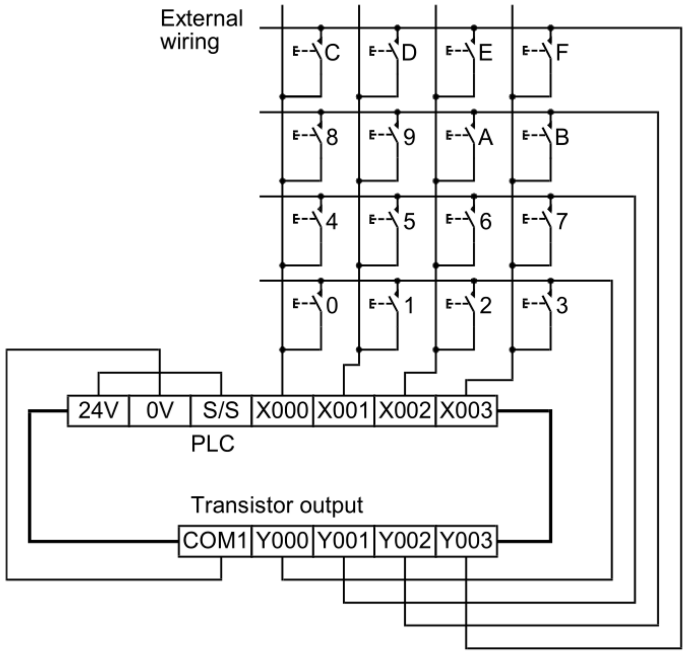
- Button output signal:

  1. When any key of A~F is pressed, M6=ON once.

  2. When any key of 0~9 is pressed, M7=ON once.

- When conditional contact X4 turns OFF, the input value before D0 remains unchanged, but all M0~M7 turn OFF.

◆ External wiring:



13.3 DSW/Digital Switch

This instruction reads the set value of digital switches.

This instruction can read a set of 4 digits (n = K1) or two sets of 4 digits (n = K2).

<table>
<tr><th>Instruction</th><th>Operand Type</th><th colspan="7">Function</th></tr>
<tr><td rowspan="2">FNC72
DSW</td><td rowspan="2">S.
D1.
D2.
n</td><td>16-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td><td rowspan="2"></td><td>32-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td></tr>
<tr><td>9 steps</td><td>DSW</td><td>Continuous Operation</td><td>——</td><td></td><td></td></tr>
<tr><td rowspan="4">Operand</td><td>S.</td><td colspan="6">Head device (X) number connected to a digital switch(Four devices are occupied.)
Target Device: X, retouch</td><td>bit</td></tr>
<tr><td>D1.</td><td colspan="6">Head device (Y) number to which the strobe signal is output(Four devices are occupied.)
Target Device: Y, retouch</td><td>bit</td></tr>
<tr><td>D2.</td><td colspan="6">Device number storing the numeric value of a digital switch("n" devices are occupied.)
Target Device: T, C, D, R, V, Z, retouch</td><td>BIN16 bit</td></tr>
<tr><td>n</td><td colspan="6">Total number of 4-digit switch sets (4 digits/set) (n = 1 or 2)
Target Device: K, H</td><td>BIN16 bit</td></tr>
</table>

Explanation of function and operation

16-bit operation(DSW)

The value of each digital switch connected to **S.** is input in the time division method (in which the value is input from the 1st digit in turn by the output signal at the interval of 100 ms), and stored to **D2.**

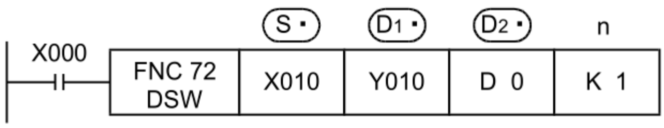
1) Data **D1.**

- A numeric value from 0 to 9999 (up to 4 digits) can be read.
- A numeric value is stored in the binary format.
- The first set is stored to **D2.** , and the second set is stored to **D2.+1**.

2) Specification of the number of sets ("n")

- When using one set of 4 digits [n = k1] A 4-digit BCD digital switch connected to **S. ~ [S.+3]** is read in turn by the strobe signal **D1. ~ [D1.+3]**, and stored in the binary format to **D2.**
- When using two sets of 4 digits [n = k2],A 4-digit BCD digital switch connected to **S. ~ [S.+3]** is read in turn by the strobe signal **D1. ~ [D1.+3]**, and stored in the binary format to **D2.**

  A 4-digit BCD digital switch connected to **S.+4 ~ [S.+7]** is read in turn by the strobe signal **D1. ~ [D1.+3]**, and stored in the binary format to **D2.+1**.



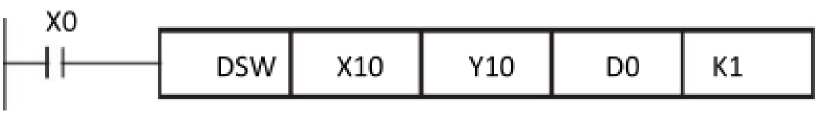
- When the command contact turns OFF. Though the contents of **D2.** do not change, all of **D1. ~ D1.+3** turn OFF.
- Number of occupied devices

  1) When two sets of 4 digits (n = K2) are used, two devices are occupied starting from **D2.**

  2) When one set of 4 digits is used, four devices are occupied starting from **S.** When two sets of 4 digits is used, eight devices are occupied starting from **S.**
- Related device:

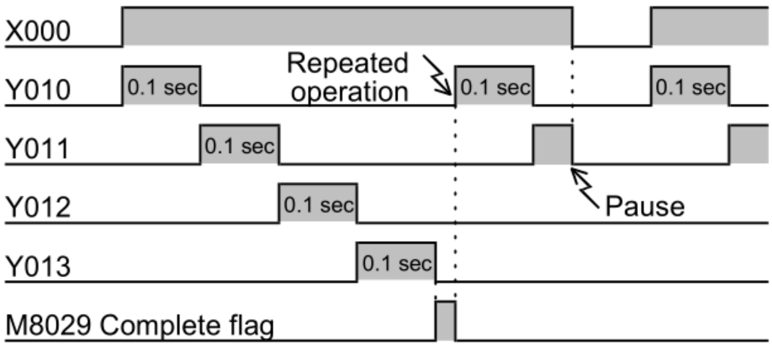
Device	Name	Description
M8029	Instruction execution	OFF: Data is being output to **D1. ~ D1.+3** or the instruction is not executed yet. ON : A cycle operation of outputting data to **D1. ~ D1.+3** (scan of the 1st to 4th digits) is completed.

Program example

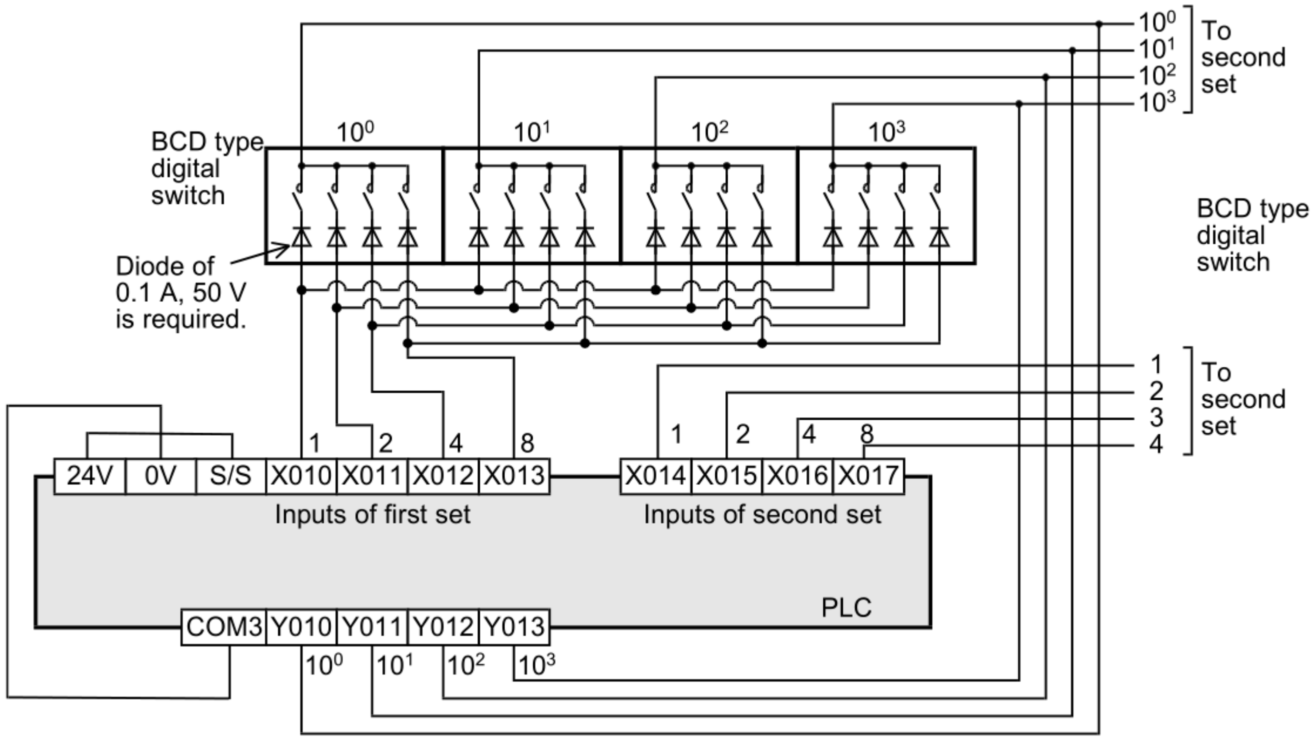


- The first group of DIP switch circuits is composed of X10~X13 and Y10~Y13, and the second group of DIP switch circuits is composed of X14~X17 and Y10~Y13. When X0=On, the instruction starts to execute, the setting value of the first group of dial switches is read and converted into BIN value and stored in D0, and the setting value of the second group of digital switches is read and converted into BIN value and stored To D1.

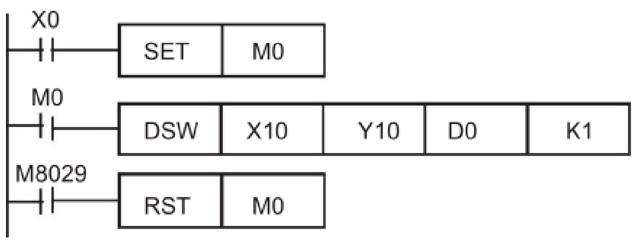
- ◆ While X000 is ON, Y010 to Y013 turn ON in turn at every 100 ms. After one cycle is finished, the execution complete flag M8029 turns ON.
- ◆ During the period when X0 is ON, Y10～Y13 will turn ON in sequence every 100ms. After the loop operates once, the end flag M8029 will be executed.



- ◆ Connection diagram



- ◆ How to use this instruction in a relay output type PLC



1) While M0 (digital switch read input) is ON, DSW (FNC 72) is driven.

2) DSW (FNC 72) completes one cycle of operation, and remains driven until the execution complete flag (M8029) turns ON.

13.4 SEGD/Seven Segment Decoder

This instruction decodes data, and turns the seven-segment display unit (1 digit) ON.

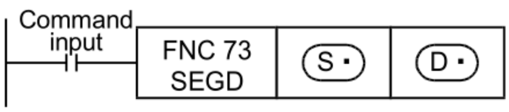
Instruction	Operand Type	Function					
		16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
FNC73 SEGD	**S.** **D.**	5 steps	SEGD	Continuous Operation	——		
P			SEGDP	Pulse (Single) Operation			

Operand Type	**S.**	Head word device to be decoded **Target Device**: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, retouch	BIN16 bit
	D.	Word device number storing the data to be displayed in the seven-segment display unit **Target Device**: KnY, KnM, KnS, T, C, D, R, V, Z, retouch	BIN16 bit

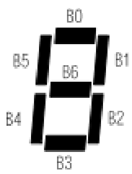
Explanation of function and operation

16-bit operation(DSW)

"0" to "F" (hexadecimal numbers) in low-order 4 bits (1 digit) of **S.** are decoded to data for the seven-segment display unit, and stored the low-order 8 bits of **D.**

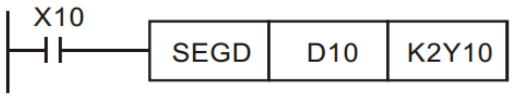


- The number of occupied points of the device, low-order 8 bits of D. are occupied, and high-order 8 bits do not change.Seven-segment decoding table:

S.		Seven segment Configuration	D.									Display data
Hexadecimal number	b3~b0		...	B7	B6	B5	B4	B3	B2	B1	B0	
0	0000		...	0	0	1	1	1	1	1	1	0
1	0001		...	0	0	0	0	0	1	1	0	1
2	0010		...	0	1	0	1	1	0	1	1	2
3	0011		...	0	1	0	0	1	1	1	1	3
4	0100		...	0	1	1	0	0	1	1	0	4
5	0101		...	0	1	1	0	1	1	0	1	5
6	0110		...	0	1	1	1	1	1	0	1	6
7	0111		...	0	0	1	0	0	1	1	1	7

	8	1000		...	0	1	1	1	1	1	1	1	8
	9	1001		...	0	1	1	0	1	1	1	1	9
	A	1010		...	0	1	1	1	0	1	1	1	A
	B	1011		...	0	1	1	1	1	1	0	0	b
	C	1100		...	0	0	1	1	1	0	0	1	C
	D	1101		...	0	1	0	1	1	1	1	0	d
	E	1110		...	0	1	1	1	1	0	0	1	E
	F	1111		...	0	1	1	1	0	0	0	1	F

Program example

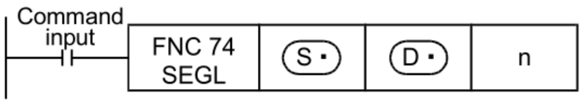


- When X10=ON, the contents (0~F: hexadecimal) of the lower 4 bits (b0~b3) of D10 are decoded into 7-segment display output, and the decoding result is temporarily stored in Y10~Y17. If the specified data exceeds 4 bits, the contents of the 4 bits are still removed for decoding.

13.5 SEGL/Seven Segment With Latch

This instruction controls one or two sets of 4-digit seven-segment display units having the latch function.

Instruction	16-bit Instruction	Function						
FNC74 SEGL	**S. D. n**	16-bit Instruction	Mnemonic	Operation Condition		32-bit Instruction	Mnemonic	Operation Condition
		7 steps	SEGL	Continuous Operation		——		
Operand	**S.**	Head word device converted into the BCD format **Target Device**: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, retouch					BIN16 bit	
	D.	Head Y number to be output **Target Device**: Y, retouch					bit	
	n	Parameter number [setting range: : K0(H0) ~ K7(H7)] **Target Device**: K, H					BIN16 bit	
Explanation of function and operation	**16-bit operation(SEGL)** The 4-digit numeric value stored in **S.** is converted into BCD data, and each digit is output to the seven- segment display unit with the BCD decoder in the time division method.							



When using one set of 4 digits (n = K0 to K3)

1) Data and strobe signal

A 4-digit numeric value stored in **S.** is converted from binary into BCD, and each digit is output in turn from **D. ~ [D.+3]** in the time division method. The strobe signal is output in turn from **[D.+4] ~ [D.+7]**in the time division method also to latch one set of 4-digit seven-segment display unit.

2) For **S.** , binary data in the range from 0 to 9999 is valid.

When using two sets of 4 digits (n = K4 to K7)

1) Data and strobe signal

a) 1st set of 4 digits

A 4-digit numeric value stored in **S.** is converted from binary into BCD, and its each digit is output in turn from **D. ~ [D.+3]** in the time division method.The strobe signal is output in turn from **[D.+4] ~ [D.+7]** in the time division method also to latch the first set of 4-digit seven-segment display unit.

b) 2nd set of 4 digits

A 4-digit numeric value stored in **S.+1** is converted from binary into BCD, and its each digit is output in turn from **D.+10 ~ D.+13** in the time division method.The strobe signal is output in turn from**D.+4 ~ D.+7** in the time division method also to latch the second set of 4-digit seven-segment display unit. (The strobe signal outputs **D.+4 ~ D.+7** are shared by the 1st and 2nd sets.)

2) For **S.** and **S.+1**, binary data in the range from 0 to 9999 is valid.

- It is determined by **n** to scan and output 4-digit seven-segment display with 1 set or 2 sets, and **n** is also used to specify the positive and negative logic output of PLC output.
- The scan time (operation cycle) multiplied by 12 is required to update (one or two sets of) the 4-digit display.
- When the command contact is set to OFF in the middle of an operation, the operation is paused. When the command contact is set to ON again, the operation is started from the beginning.
- Number of occupied devices

  When one set of 4 digits is used: 1 device is occupied from the head device specified in **S.**

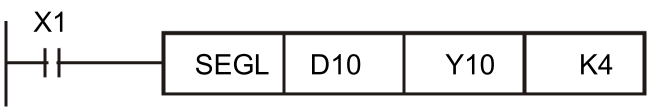
  8 devices are occupied from the head device specified in **D.**

  When two sets of 4 digits are used: 2 devices are occupied from the head device specified in S.

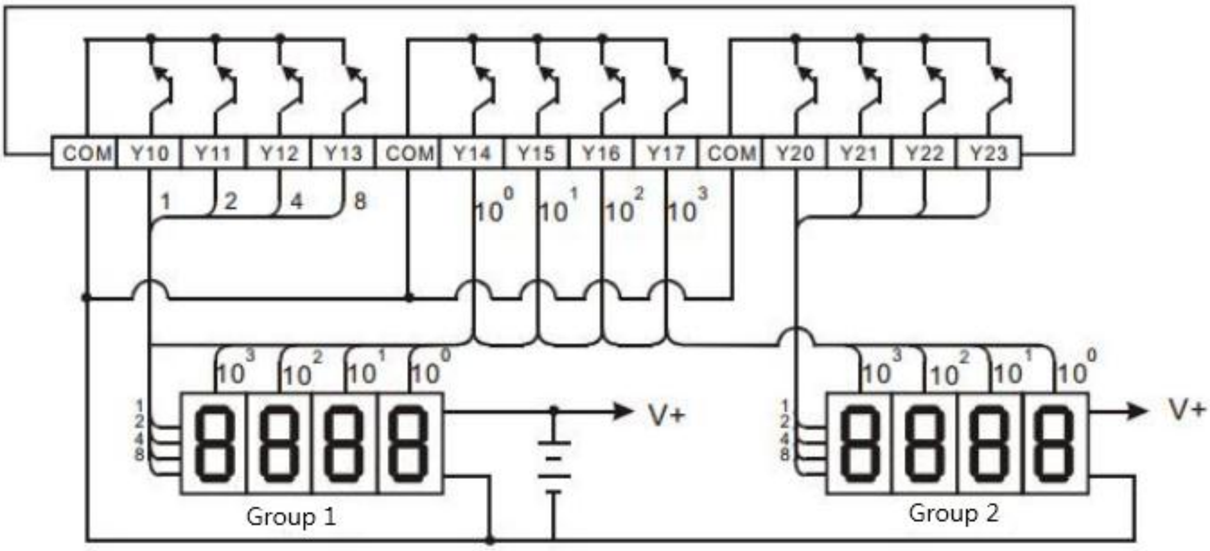
  12 devices are occupied from the head device specified in D.

  Note: Even if the number of digits is small, the occupied points cannot be used for other purposes.

Program example



- When X10=ON, the instruction starts to be executed, and Y10~Y17 form a seven-segment display scanning circuit. The value in D10 is converted into BCD code and sent to the first group of seven-segment display. The value in D11 is converted into BCD The code is sent to the second group of seven-segment displays and displayed. If the value in D10 or D11 exceeds 9,999, an operation error will occur.
- When X10=ON, Y14~Y17 will automatically scan cyclically. Each scan cycle needs 12 scan cycles, and the scan completion flag signal M9029=ON for one scan cycle.
- Seven-segment display units wiring details



13.6 ARWS/Arrow Switch

This instruction inputs data through arrow switches used for shifting the digit and incrementing/decrementing the numeric value in each digit.

Instruction	Operand type	Function					
FNC75 ARWS	**S.** **D1.** **D2.** **n**	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
		9 steps	ARWS	Continuous Operation	——		

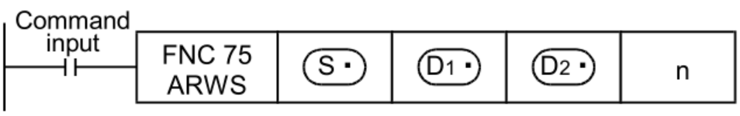
Operand	**S.**	Head bit device number to be input **Target Device**: X, Y, M, S, D .b	16-bit binary
	D1.	Word device number storing data converted into BCD **Target Device**: T, C, D, R, V, Z	16-bit binary
	D2.	Head bit device (Y) number connected to seven-segment display unit **Target Device**: Y	16-bit binary
	n	Number of digits of seven-segment display unit [setting range: K0 to K3] **Target Device**: K, H	16-bit binary

Explanation of function and operation

16-bit operation(ARWS)

Four arrow switches are connected to the inputs **S. ~ S.+3**, a seven-segment display unit having the BCD decoder is connected to the outputs **D2. ~ D2.+7**, and a numeric value is input to **D1.**

D1. actually stores a 16-bit binary value in the range from 0 to 9999



1) Specifying the number of digits of the seven-segment display unit having the BCD decoder n
   In the explanation below, "n" is set to "4" (up to the 10^3 digit).
2) Operation of the digit selection switches(S.+2、S.+3)

   - Operation when the lower digit input **S.+2** turns ON

     Every time the lower digit switch is pressed, the digit specification changes in the way $10^3 \to 10^2 \to 10^1 \to 10^0 \to 10^3$

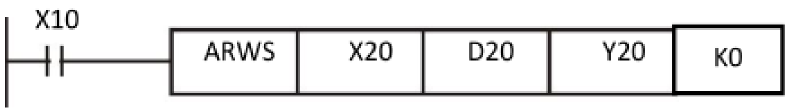
   - Operation when the higher digit input **S.+3** turns ON

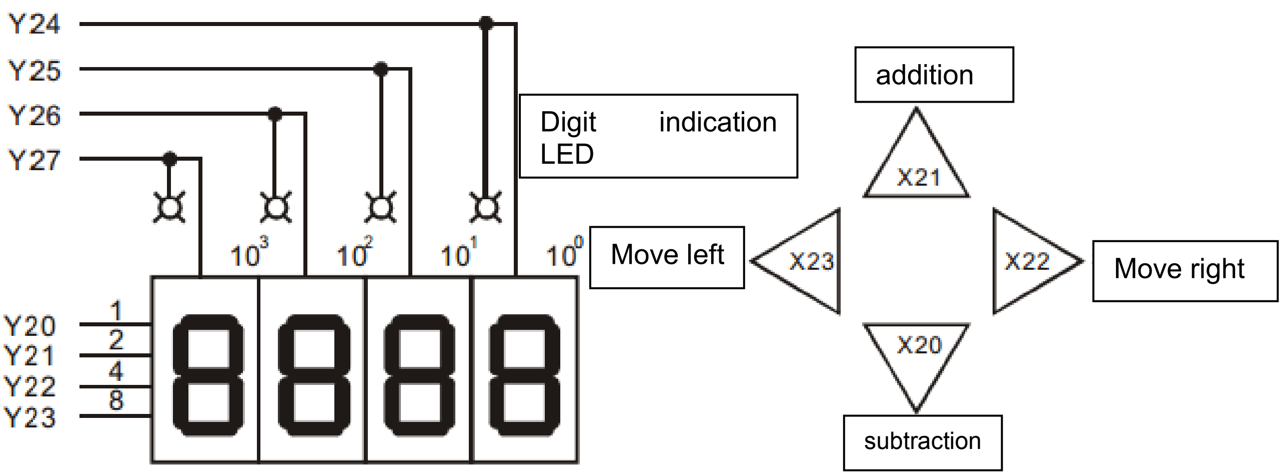
     Every time the higher digit switch is pressed, the digit specification changes in the wa$10^3 \to 10^0 \to 10^1 \to 10^2 \to 10^3$
3) Operation of the LED for displaying a selected digit **(D2.+4 ~ D2.+7)**

   A specified digit can be displayed by the LED offered by the strobe signals **D2.+4 ~ D2.+7**

	4) Operation of the switches for changing data in each digit **(S.、 S.+1)** In a digit specified by a digit selection switch described above, data is changed as follows: - When the increment input turns ON Every time the increment switch is pressed, the contents of **D1.** change in the way 0→1→2→…→8→9→0→1 - When the decrement input turns ON Every time the decrement switch is pressed, the contents of **D1.** change in the way 0→9→8→7…1→0→9 The contents can be displayed in the seven-segment display unit. As described above, a target numeric value can be written to **D1.**using a series of operation while looking at the seven-segment display unit. ● Number of occupied devices Four input devices are occupied starting from **S.** Eight output devices are occupied starting from **D.** ● ARWS instruction can be used only once in a program. When ARWS instruction should be used two or more times, use the indexing (V, Z) function.

Program example





seven-segment showing 4-digit setting values

4 switches are used to move the left and right and the addition and subtraction of the set value

- ◆ This instruction is executed, X20 is defined as the down key, X21 is defined as the up key, X22 is defined as the right key, X23 is defined as the left key, and the operation and display of the external set value are performed by using the up, down, left and right keys. Store the setting value in D20, setting value range: 0~9,999.
- ◆ When X10=ON, the command start digit 10^3 is the effective setting digit. If the left button is pressed, the effective setting digit will show a cycle of 10^3→100^0→10^1→10^2→10^3→100^0.
- ◆ If you press the right shift button, the effective setting bit will show the direction of

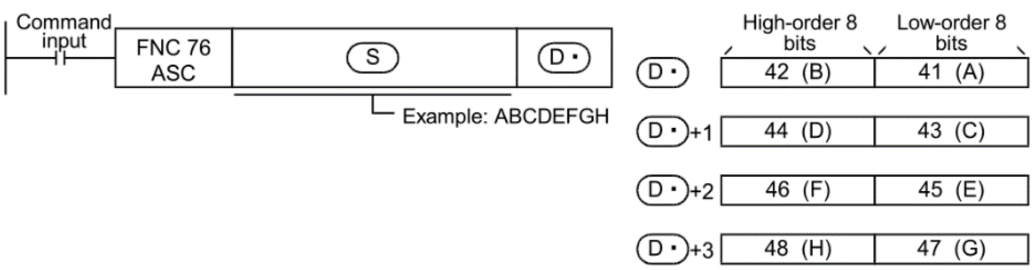
103→102→101→100→103→102 cyclically jumping. Simultaneously with the cycle, the digit indicator connected by Y24~Y27 is also cycled to indicate the effective digit setting.

◆ If you press the up button, the content of the effective setting digits will change from 0→1→2→…8→9→0→1. If the down button is pressed, the content of the effective setting digits changes from 0→9→8→...1→0→9. At the same time, the changed value is also displayed on the seven-segment display.

13.7 ASC/ASCII Code Data Input

This instruction converts a half-width alphanumeric character string into ASCII codes. Use this instruction for selecting one among two or more messages and displaying it on an external display unit.

Instruction	Operand type	Function					
FNC76 ASC	**S. D**	16-bit Instruction 11 steps	Mnemonic ASC	Operation Condition Continuous Operation	32-bit Instruction ——	Mnemonic	Operation Condition
Operand	**S.**	Eight half-width alphanumeric characters input from a personal computer **Target Device**: Character string					Character string(onlyASCII codes)
	D.	Head word device number storing ASCII data **Target Device**: T, C, D, R, retouch					16-bit binary

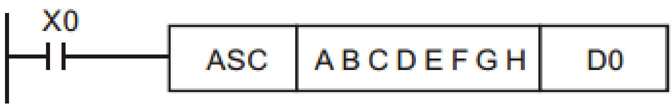
Explanation of function and operation	**16-bit operation(ASC)** The half-width alphanumeric characters specified in **S.** are converted into ASCII codes, and each ASCII code is transferred in turn to **D.** • **S.**can handle half-width characters A to Z, 0 to 9 and symbols (, but cannot handle regular-width characters). A character string is entered when a program is created with a programming tool. • **D.** stores converted ASCII codes in the order of low-order 8 bits and high-order 8 bits by 2 characters/ byte at one time.  ● Related devices

Device	Name	Content
M8161	Extension function flag	OFF: Two characters are stored to low-order 8 bits and high-order 8 bits in this order at one time (2 characters/word). ON : One character is stored to low-order 8 bits at one time (1 character/word).

- Number of occupied devices
  M8161=OFF, **D.**occupies as many devices as the number of characters divided by "2". (The decimal point is rounded up.)
  M8161=ON, **D.**occupies as many devices as the number of characters in the character string.

- ARWS instruction can be used only once in a program. When ARWS instruction should be used two or more times, use the indexing (V, Z) function.

Program example



	b15	b0
D0	42H (B)	41H (A)
D1	44H (D)	43H (C)
D2	46H (F)	45H (E)
D3	48H (H)	47H (G)
	upper 8 bits	lower 8 bits

- When X0=ON, specify A~H to be converted into ASCII code and temporarily stored in D0~D3.
- When M8161=ON, the converted ASCII code of each letter will occupy the lower 8 bits (b7~b0) of a register, and the upper 8 bits are invalidly filled with 0, which means that a letter is used to store a letter.

	b15	b0
D0	00 H	41H (A)
D1	00 H	42H (B)
D2	00 H	43H (C)
D3	00 H	44H (D)
D4	00 H	45H (E)
D5	00 H	46H (F)
D6	00 H	47H (G)
D7	00 H	48H (H)
	upper 8 bits	lower 8 bits

13.8 PR/Print (ASCII Code)

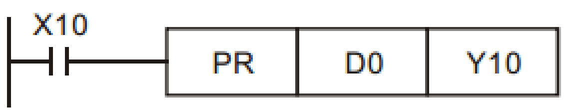
This instruction outputs ASCII code data to outputs (Y) in parallel.

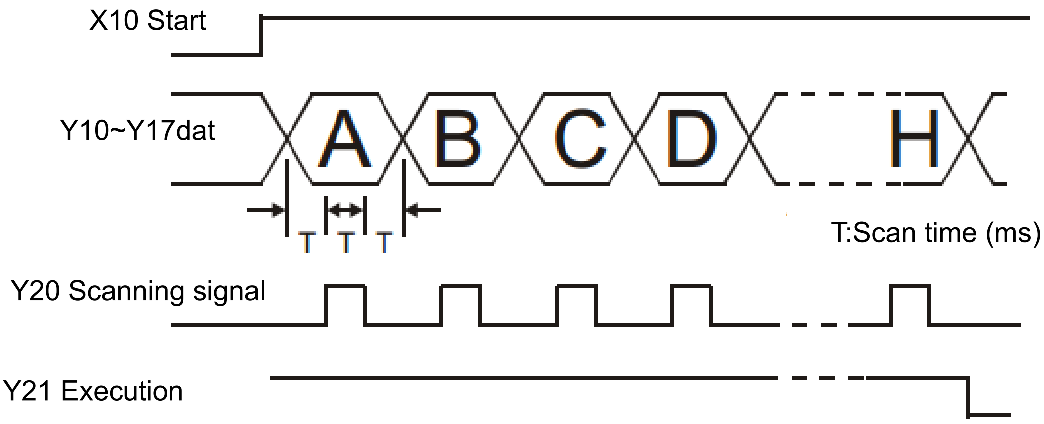
Instruction	Operand Type	Function						
FNC77 PR	**S. D**	16-bit Instruction	Mnemonic	Operation Condition		32-bit Instruction	Mnemonic	Operation Condition
		5 steps	PR	Continuous Operation		——		
Operand	**S.**	Head device number storing ASCII code data **Target Device**: T, C, D, R, retouch						Character string (only ASCII codes)
	D.	Head output (Y) number to which ASCII code data is output **Target Device**: Y, retouch						16-bit binary
Explanation of function and operation	**16-bit operation(PR)** ASCII codes stored in low-order 8 bits (1 byte) of **S. ~ S.+7** are output to **D. ~ D.+7** in turn by one character at a time in the time division method. ● Related devices							

Device	Name	Content
M8027 (from RUN→STOP clear)	PR model	OFF:8-byte serial output (fixed to 8 characters) ON :16-byte serial output (1 to 16 characters)

- While the command input is ON: Even if the command input is continuously ON or if the pulse operation type instruction is used, execution is completed after a series of outputs. M8029 turns ON only while M8027 is ON. While the command input is OFF:The output is all OFFARWS instructions, and only one can be used in the program. To use more than one, please use the index modification (V, Z) function programming.
- This instruction is executed in synchronization with the scan time.
  If the scan time is short, the constant scan mode can be used. If the scan mode is too long, the timer interrupt function can be used.
- When "00H (NUL code)" is contained in the data (while M8027 is ON).The instruction is executed completely, and the data after "00H" is not output. M8029 remains ON during one operation cycle.

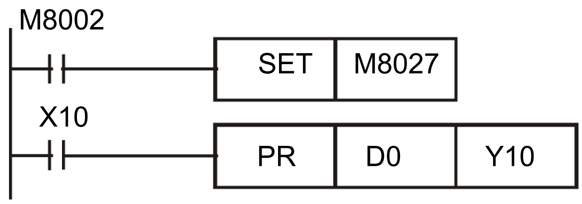
Program example 1

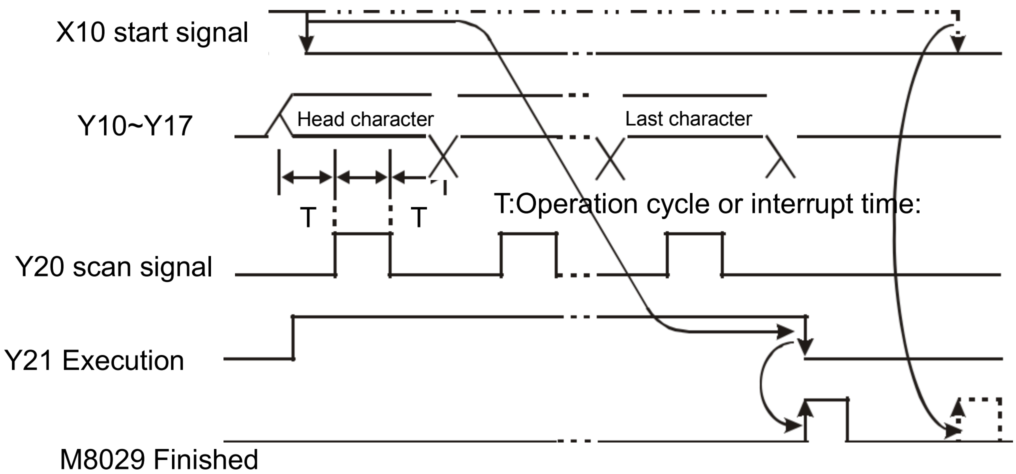




- When M8027=OFF and X10=ON change, the instruction is executed, Y10 (low bit) ~ Y17 (high bit) is designated as the data output point, the scan signal is designated Y20, and the monitoring signal during execution is designated as Y21. This mode can perform sequential output of 8 words. And during the output, if the conditional contact is OFF, the data output will be stopped immediately, and all the outputs will be turned OFF.
- When X10 turns OFF during the execution of the instruction, the data output is interrupted, and when X10 turns ON again, the data is sent again.

Program example 2





- PR instruction is a serial output instruction with 8 bits. When special auxiliary relay M8027=Off, it can execute a serial output of up to 8 words. When M8027=ON, it can execute a serial of 1~16 output.
- When M8027=ON, X10 changes from OFF→ON, the instruction is executed, Y10 (low bit) ~ Y17 (high bit) is designated as the data output point, the scan signal is designated Y20, and the monitoring signal during execution is designated Y21. This mode can perform sequential output of 16 words. And during the output, if the conditional contact is OFF, it will stop after the data output is completed.
- If 00H (NUL) is encountered in the character string, it means the end of the character string, and then the text will not be processed.
- When conditional contact X10 is ON→OFF, the data output automatically stops after one cycle. However, if X10 is always ON, M8029 will not operate.

13.9 FROM/Read From A Special Function Block

This instruction reads the contents of buffer memories (BMF) in a special extension unit/block attached to a PLC.

When a large capacity of buffer memory (BFM) data is read by this instruction, a watchdog timer error may occur. When bad effect is not given to the control even if data to be read is divided, use RBFM (FNC278) instruction.

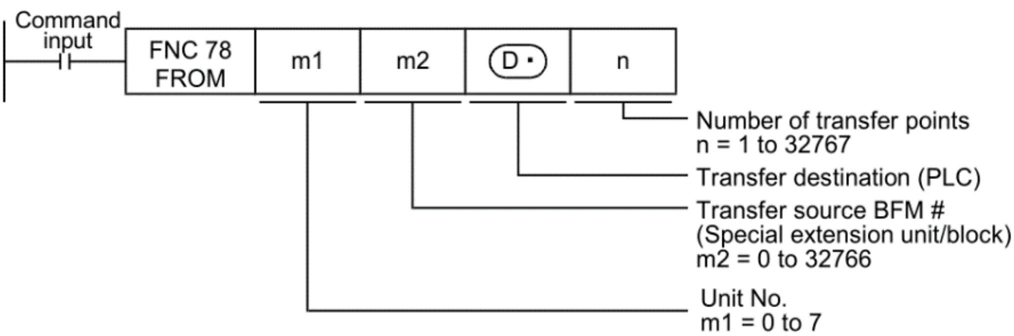
Instruction	Operand Type	Function					
		16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
D **FNC78 FROM** P	**m1 m2 D. n**	9 steps	FROM	Continuous Operation	17 steps	DFROM	Continuous Operation
			FROMP	Pulse (Single) Operation		DFROMP	Pulse (Single) Operation

Operand			
Operand	**m1**	Unit number of a special extension unit/block (K0 to K7 from the right side of the main unit) **Target Device**: D, R, K, H	16/32-bit binary
	m2	Transfer source buffer memory (BFM) number **Target Device**: D, R, K, H	16/32-bit binary
	D.	Transfer destination device number **Target Device**: KnY, KnM, KnS, T, C, D, R, V, Z, retouch	16/32-bit binary
	n	Number of transfer points **Target Device**: D, R, K, H	16/32-bit binary

Explanation of function and operation

1. 16-bit operation (FROM、FROMP)

"n"-point 16-bit data starting from the buffer memory (BFM) # m2 inside a special extension unit/block No. m1 are transferred (read) to "n"-point 16-bit data starting from **D.** inside a PLC.



2. **32-bit operation(DFROM、DFROMP)**

"n" 32-bit data starting from the buffer memory (BFM) # **[m2+1、m2]** inside a special extension unit/block No.m1 are transferred (read) to "n" devices starting from **[D.+1、D.]** inside a PLC.

- Related devices

Device	Name	Content
M8028	Enable interrupt flag	OFF: Disables interrupts.(Interrupts are executed after FROM/TO instruction is executed.) ON : Enables interrupts.

- Digit specification in bit device **D.**
  For the 16-bit operation instruction, specify K1 to K4. For the 32-bit operation instruction, specify K1 to K8.
- Note that when D and R are designated as **m1**, **m2**, and **n** of a 32-bit instruction, the 32-bit values of **[m1+1, m1], [m2+1, m2] [n+1, n]** become effective.
  For example: when DFROM D0 D2 D100 R0, then **m1**=[D1, D0], **m2**=[D3, D2], **n**=[R1, R0].

Program example

◆ Read out the contents of BFM#29 of expansion module number 0 into D0 of PLC, and read out the contents of BFM#30 into D1, and read two strokes at a time (n=2).

◆ When X0=ON, the instruction is executed. When X0 turns OFF, the instruction is not executed, and the content of the previously read data has not changed.

13.10 TO/Write To A Special Function Block

This instruction writes data from a PLC to buffer memories (BFM) in a special extension unit/block.
When a large capacity of data is written to buffer memories (BFM) by this instruction, a watchdog timer error may occur. When splitting the data to be written does not affect the control, use WBFM (FNC279) instruction.

<table>
<tr><th>Instruction</th><th>Operand Type</th><th colspan="7">Function</th></tr>
<tr><td rowspan="3">D FNC79 TO P</td><td rowspan="3">m1
m2
S.
n</td><td>16-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td><td rowspan="3"></td><td>32-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td></tr>
<tr><td rowspan="2">9 steps</td><td>TO</td><td>Continuous Operation</td><td rowspan="2">17 steps</td><td>DTO</td><td>Continuous Operation</td></tr>
<tr><td>TOP</td><td>Pulse (Single) Operation</td><td>DTOP</td><td>Pulse (Single) Operation</td></tr>
<tr><td rowspan="3">Operand</td><td>m1</td><td colspan="6">Unit number of a special extension unit/block
(K0 to K7 from the right side of the main unit)
Target Device: D, R, K, H</td><td>16/32-bit binary</td></tr>
<tr><td>m2</td><td colspan="6">Transfer destination buffer memory (BFM) number
Target Device: D, R, K, H</td><td>16/32-bit binary</td></tr>
<tr><td>S.</td><td colspan="6">Device number storing the transfer source data
Target Device: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, retouch</td><td>16/32-bit binary</td></tr>
</table>

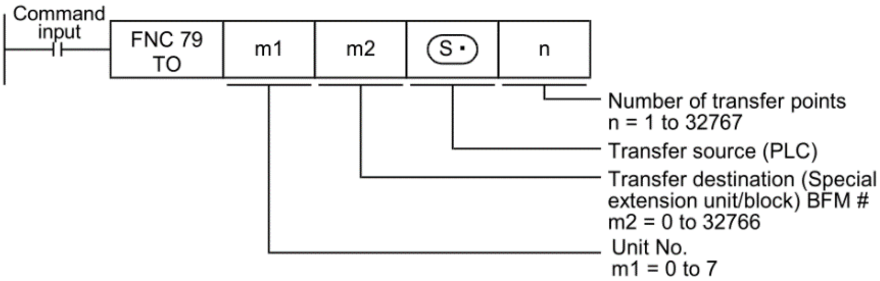
	n	Number of transfer points **Target Device**: D, R, K, H	16/32-bit binary

Explanation of function and operation

1. 16-bit operation(TO、TOP)

"n"-point 16-bit data starting from **S.** inside a PLC are transferred (written) to "n"-point buffer memories

starting from the buffer memory (BFM) # **m2** inside a special extension unit/block No. **m1**.



2. 32-bit operation(DTO、DTOP)

n"-point 32-bit data starting from**[S., S.+1]** inside a PLC are transferred (written) to "n"-point buffer memories starting from the buffer memory (BFM) # [m2+1, m2] inside a special extension unit/block No. m1.

- Related devices

Device	Name	Content
M8028	Enable interrupt flag	OFF: Disables interrupts.(Interrupts are executed after FROM/TO instruction is executed.) ON : Enables interrupts.

- Digit specification in bit device **S.**

  For the 16-bit operation instruction, specify K1 to K4. For the 32-bit operation instruction, specify K1 to K8.
- Note that when D and R are designated as **m1**, **m2**, and **n** of a 32-bit instruction, the 32-bit values of **[m1+1, m1], [m2+1, m2] [n+1, n]** become effective.

  For example: DTO D0 D2 D100 R0, then **m1**=[D1, D0], **m2**=[D3, D2], **n**=[R1, R0].

Program example

- Using the 32-bit instruction DTO, the action of the program is to write the contents of D11 and D10 into BFM#13 and BFM#12 of the expansion module with the number 0, and write only one sum at a time (n=1).
- When X0=ON, the instruction is executed. When X0 turns OFF, the instruction is not executed and the written data remains unchanged.

14 External Device SER (Option equipment)

FNC NO.	instruction	function	Support model		
			3G PLC	2N PLC	MX2N PLC
80	RS	Serial Communication	★	★	★
81	PRUN	Parallel Run	★		
82	ASCI	HEX to ASCII conversion	★	★	★
83	HEX	ASCII to HEX conversion	★	★	★
84	CCD	Check Code	★		★
85	VRRD	Volume Read			
86	VRSC	Volume Scale			
87	RS2	Serial Communication 2	★		
88	PID	PID Control Loop	★	★	★
89	—				

14.1 RS/Serial Communication

This instruction sends and receives data in no-protocol communication by way of a serial port (only the ch1) in accordance with RS-232C or RS-485 provided in the main unit.

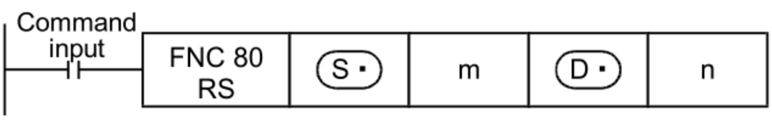
Instruction	Operand type	Function					
FNC 80 RS	**S.** **m** **D.** **n**	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
		9 steps	RS	Continuous Operation		—	

Operand			
Operand	**S.**	Head device of data registers storing data to be sent **Target Device**: D, R, retouch	16-bit binary or character string
	m	Number of bytes of data to be sent [setting range: 0 to 4096]*1 **Target Device**: D, R, K, H	16-bit binary
	D.	Head device of data registers storing received data when receiving is completed **Target Device**: D, R, retouch	16-bit binary or character string
	n	Number of bytes to be received [setting range: 0 to 4096]*1 **Target Device**: D, R, K, H	16-bit binary

Explanation of function and operation

16-bit operation(RS)

This instruction sends and receives data in no-protocol communication by way of serial ports in accordance with RS-232C or RS-485 provided in the main unit.



- Related devices

Device	Name	Device	Name
M8063	Serial communication error 1	**D8120**	Communication format setting
M8121	Sending wait flag	**D8122**	Remaining number of data to be sent
M8122	Sending request	**D8123**	Monitor for number of received data
M8123	Receiving complete flag	**D8124**	Header
M8124	Carrier detection flag	**D8125**	Terminator

		M8129	Time-out check flag	**D8129**	Time-out time setting
		M8161	8-bit processing mode	**D8063**	Error code number of serial communication error 1

❖ **For detailed configuration and case description, please refer to 2N series PLC/MX2N series PLC/3G series PLC programming manual**

14.2 PRUN/Parallel Run (Octal Mode)

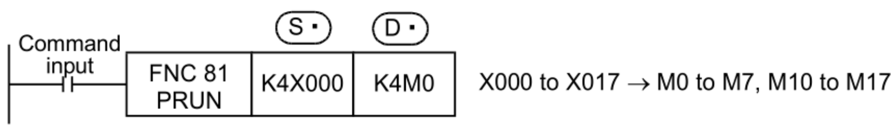
This instruction handles the device number of **S.** with digit specification and the device number of **D.** as octal numbers, and transfers data.

Instruction	Operand type	Function					
D / **FNC 81 PRUN** / P	**S.** **D.**	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
		5 steps	PRUN	Continuous Operation	9 steps	DPRUN	Continuous Operation
			PRUNP	Pulse (Single) Operation		DPRUNP	Pulse (Single) Operation
Operand	**S.**	Digit specification **Target Device**: KnX, KnM, retouch					16/32-bit binary
	D.	Device number of transfer destination **Target Device**: KnY, KnM, retouch					16/32-bit binary

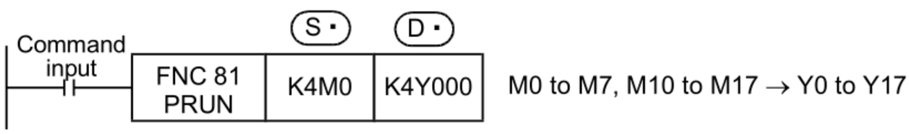
Explanation of function and operation

1. 16-bit operation (PRUN、PRUNP)

Octal bit device → Decimal bit device

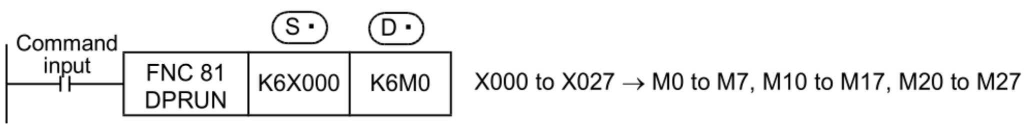


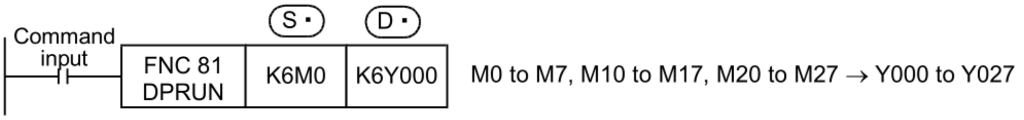
Decimal bit device→Octal bit device



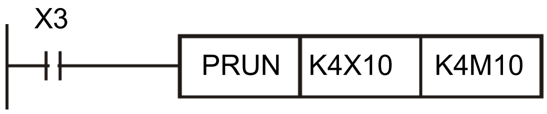
2. 32-bit operation(DPRUN、DPRUNP)

Octal bit device → Decimal bit device

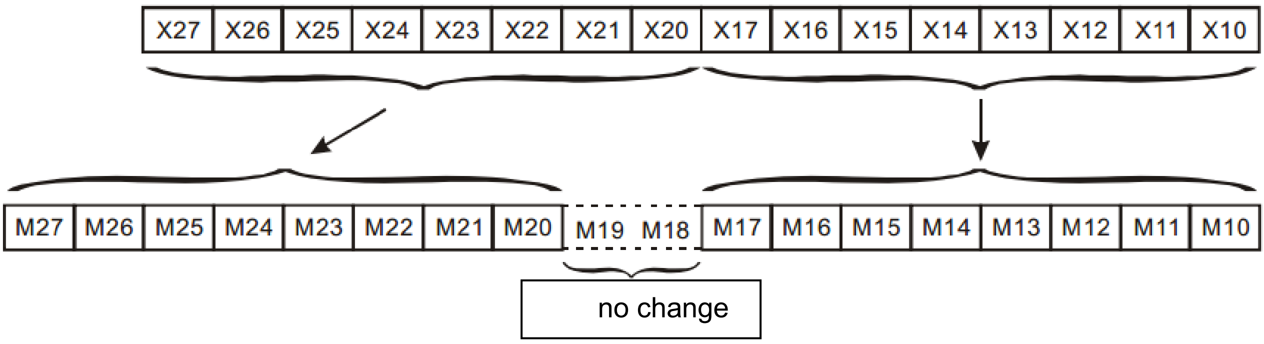


	Decimal bit device → Octal bit device 

Program example



◆ When X3=ON, transfer the contents of K4X10 to K4M10 in octal form.

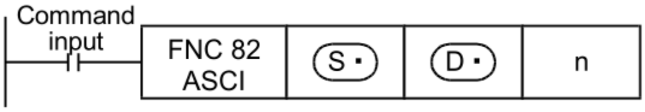


14.3 ASCI/Hexadecimal to ASCII Conversion

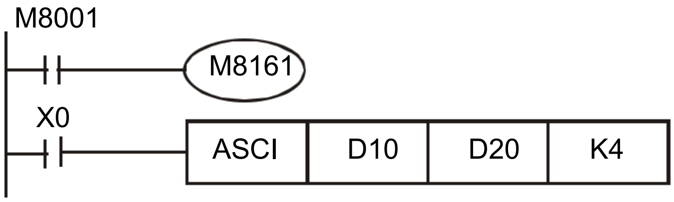
This instruction converts hexadecimal code into ASCII code.

On the other hand, BINDA (FNC261) instruction converts binary data into ASCII code, and ESTR (FNC116) instruction converts binary floating point data into ASCII code.

Instruction	Operand type	Function					
		16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
FNC 82 ASCI **P**	**S.** **D.** **n**	7 steps	ASCI ASCIP	Continuous Operation Pulse (Single) Operation		—	
Operand	**S.**	Head device number storing hexadecimal code to be converted **Target Device**: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, retouch					16-bit binary
	D.	Head device number storing converted ASCII code **Target Device**: KnY, KnM, KnS, T, C, D, R, retouch					Character string (only ASCII code)
	n	Number of characters (digits) of hexadecimal code to be converted [setting range: 1 to 256] **Targer Device**: D, R, K, H					16-bit binary

Explanation of function and operation	
	1. **16-bit operation(ASCI、 ASCINP)** n" hexadecimal code characters (digits) stored in **n** and later are converted into ASCII code, and then stored to the devices **D.** and later. The 16-bit mode and 8-bit mode options are available for this instruction.  **2. 16-bit conversion mode (while M8161 is OFF) (M8161 is also used for the RS, HEX, CCD and CR instructions.)** Each digit of hexadecimal data stored in **S.** and later is converted into ASCII code, and transferred to the high-order 8 bits and low-order 8 bits of each device **D.** and later. The number of digits (characters) to be converted is specified by "n". **3. 8-bit conversion mode (while M8161 is ON) (M8161 is used also for the RS, HEX, CCD and CRC instructions.)** Each digit of hexadecimal data stored in **S.** and later is converted into an ASCII code, and transferred to low-order 8 bits of each device **D.** and later. The number of digits (characters) to be converted is specified by "n". "0" is stored in high-order 8 bits of each device **D.** and later.

Program example 1



- M8161=OFF, which it is 16-bit conversion mode.
- When X0=ON, the four hexadecimal values in D10 are converted into ASCII codes and transferred to the register starting from D20.
- Assumptions:

(D10)=0123 H	'0'=30H	'4'=34H	'8'=38H
(D11)=4567 H	'1'=31H	'5'=35H	'9'=39H
(D12)=89AB H	'2'=32H	'6'=36H	'A'=41H
(D13)=CDEF H	'3'=33H	'7'=37H	'B'=42H

- When n=4, the composition of bits:

D10=0123 H

0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1
0				1				2				3			

D20 High-order Low-order

0	0	1	1	0	0	0	1	0	0	1	1	0	0	0	0
'1'→31H								'0'→30H							

D21 High-order Low-order

0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	0
'3'→33H								'2'→32H							

- When n=6, the composition of bits:

D10=0123 H

0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1
0				1				2				3			

D11=4567 H

0	1	0	0	0	1	0	1	0	1	1	0	0	1	1	1
4				5				6				7			

transfer↓

D20 High-order Low-order

0	0	1	1	0	1	1	1	0	0	1	1	0	1	1	0
‘7’→37H								‘6’→36H							

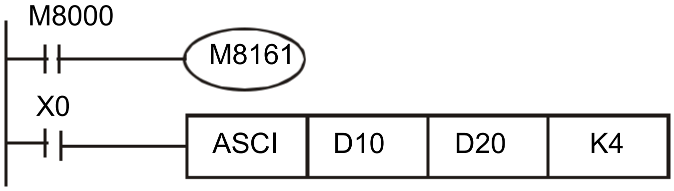
D21 High-order Low-order

0	0	1	1	0	0	0	1	0	0	1	1	0	0	0	0
‘1’→31H								‘0’→30H							

D22 High-order Low-order

0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	0
‘3’→33H								‘2’→32H							

Program example 2



- M8161=ON, specify 8-bit conversion mode.
- When X0=ON, the four hexadecimal values in D10 are converted into ASCII codes and transferred to the register starting from D20.
- Assumptions:

(D10)=0123 H	‘0’=30H	‘4’=34H	‘8’=38H
(D11)=4567 H	‘1’=31H	‘5’=35H	‘9’=39H
(D12)=89AB H	‘2’=32H	‘6’=36H	‘A’=41H
(D13)=CDEF H	‘3’=33H	‘7’=37H	‘B’=42H

- When n=2, the composition of bits:

D10=0123 H

0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1
0				1				2				3			

D20=2 ASCII Code=32H

0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0
								3				2			

D21=3 ASCII code=33H

0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1
								3				3			

- When n=6, the composition of bits:

D10=0123 H

0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1
0				1				2				3			

transfer↓

D20

0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
								‘0’→30H							

D21

0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1
								‘1’→31H							

D22

0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0
								‘2’→32H							

D23

0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1
								‘3’→33H							

14.4 HEX/ASCII to Hexadecimal Conversion

This instruction converts ASCII codes into hexadecimal codes. On the other hand, DABIN (FNC260) instruction converts ASCII codes into binary data, and EVAL (FNC117) instruction converts ASCII codes into binary floating point data.

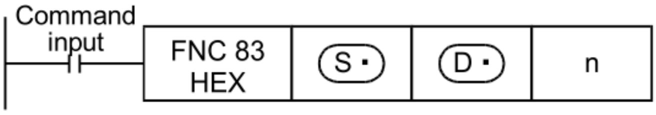
Instruction	Operand type	Function						
FNC 83 HEX **P**	**S.** **D.** **n**	16-bit Instruction	Mnemonic	Operation Condition		32-bit Instruction	Mnemonic	Operation Condition
		7 steps	HEX HEXP	Continuous Operation			—	
Operand	**S.**	Head device number storing ASCII code to be converted **Target Device**: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, retouch						Character string (only ASCII code)
	D.	Head device number storing converted hexadecimal code **Target Device**: KnY, KnM, KnS, T, C, D, R, retouch						16/32-bit binary
	n	Number of ASCII codes (bytes) to be converted [setting range: 1 to 256] **Target Device**: D, R, K, H						16-bit binary

Explanation of function and operation

1. 16-bit operation(HEX、HEXP)

Among the ASCII codes stored in **S.** and later, "n" characters are converted into hexadecimal codes, and

then stored to the devices **D.** and later. The 16-bit mode and 8-bit mode are available for this instruction.



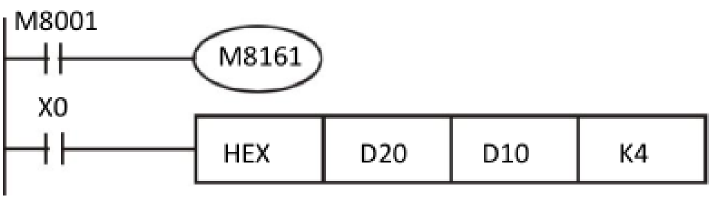
2. 16-bit conversion mode (while M8161 is OFF) (M8161 is used also for the RS, ASCI, CCD, and CRC instructions.)

Each ASCII code stored in high-order 8 bits and low-order 8 bits of devices **S.** and later is converted into a hexadecimal code, and transferred to devices **D.** and later in units of 4 digits. The number of characters to be converted is specified by "n".

3. 8-bit conversion mode (while M8161 is ON) (M8161 is used also for the RS, ASCI, CCD and CRC instructions.)

Each ASCII code stored in the low-order 8 bits of each device **S.** and later is converted into a hexadecimal code, and transferred to device **D.** and later in 4-digits units. The number of characters to be converted is specified by "n".

Program example 1



- M8161=OFF, 16-bit conversion mode
- When X0=ON, the ASCII code in the register starting from D20 is converted into a hexadecimal value, and every 4 digits are transferred to the register starting from D10, the number of converted ASCII codes is n=4.
- Assumptions:

S	ASCII code	HEX code
D20 Low	H43	‘C’
D20 high	H44	‘D’
D21 Low	H45	‘E’
D21 high	H46	‘F’
D22 Low	H38	‘8’
D22 high	H39	‘9’
D23 Low	H41	‘A’
D23 high	H42	‘B’

S	ASCII code	HEX code
D24 Low	H34	‘4’
D24 high	H35	‘5’
D25 Low	H36	‘6’
D25 high	H37	‘7’
D26 Low	H30	‘0’
D26 high	H31	‘1’
D27 Low	H32	‘2’
D27 high	H33	‘3’

- When n=4, the composition of bits:

D20 High-order Low-order

0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1
44H→'D'								43H→'C'							

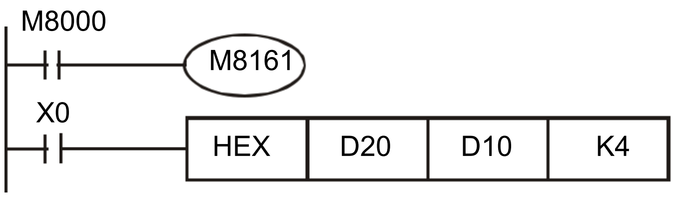
D21 High-order Low-order

0	1	0	0	0	1	1	0	0	1	0	0	0	1	0	1
46H→'F'								45H→'E'							

D10

1	1	0	0	1	1	0	1	1	1	1	0	1	1	1	1
C				D				E				F			

Program example 2



- M8161=ON, 8-bit conversion mode
- Assumptions:

S	ASCII code	HEX code
D20	H43	'C'
D21	H44	'D'
D22	H45	'E'
D23	H46	'F'
D24	H38	'8'
D25	H39	'9'
D26	H41	'A'
D27	H42	'B'

S	ASCII code	HEX code
D28	H34	'4'
D29	H35	'5'
D30	H36	'6'
D31	H37	'7'
D32	H30	'0'
D33	H31	'1'
D34	H32	'2'
D35	H33	'3'

- When n=2, the composition of bits:

D20 High-order Low-order

								0	1	0	0	0	0	1	1
								43H→'C'							

D21 High-order Low-order

								0	1	0	0	0	1	0	0
								44H→'D'							

D10

0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	1
								C				D			

14.5 CCD/Check Code

This instruction calculates the horizontal parity value and sum check value in the error check methods used in communication. There is another check method, CRC (cyclic redundancy check) also. For obtaining CRC value, use CRC instruction.

Instruction		Operand type	Function					
FNC 84 CCD	P	S. D. n	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
			7 steps	CCD CCDP	Continuous Operation		—	

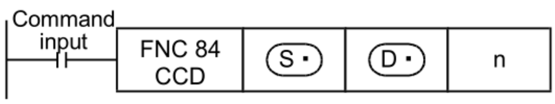
Operand	Operand type	Description	Data type
Operand	**S.**	Head device number of applicable device **Target Device**: KnX, KnY, KnM, KnS, T, C, D, R, retouch	16-bit binary or character string
	D.	Head device number storing the calculated data **Target Device**: KnY, KnM, KnS, T, C, D, R, retouch	16-bit binary or character string
	n	Number of data [setting range: 1 to 256] **Target Device**: D, R, K, H	16-bit binary

Explanation of function and operation

1. 16-bit operation(CCD、CCDP)

The addition data and horizontal parity value of data stored in **S.**～**S.+n-1** are calculated. The addition data is stored to **D.** , and the horizontal parity value is stored to **D. +1**.

The 16-bit mode and 8-bit mode are available in this instruction.



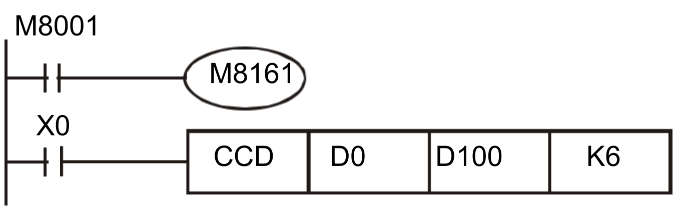
2. 16-bit conversion mode (while M8161 is OFF) (M8161 is also used for the RS, ASCI, HEX and CRC instructions.)

With regard to "n" data starting from **S.** , the addition data and horizontal parity data of high-order 8 bits and low-order 8 bits are stored to **D.** and **D.+1** respectively.

3. 8-bit conversion mode (while M8161 is ON) (M8161 is used also for the RS, ASCI, HEX and CRC instructions.)

With regard to "n" data starting from **S.** , the addition data and horizontal parity data of only low-order 8 bits are stored to **D.** and **D.+1** respectively.

Program example 1



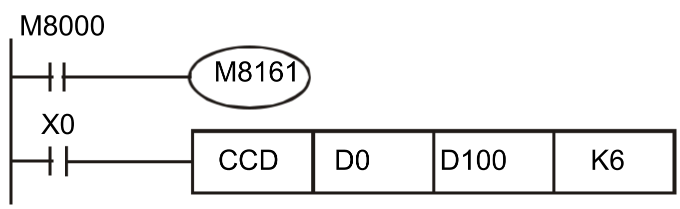
S.	Example of data contents
D0 Low	K100=01100100
D0 high	K111=0110111①
D1 Low	K120=01111000
D1 high	K202=11001010
D2 Low	K123=0111101①
D2 high	K211=1101001①
D100(sum)	K867
D101	0001000①

When the number of "1" is odd, the horizontal parity is "1"
When the number of "1" is even, the horizontal parity is "0".

D100	0	0	0	0	0	0	1	1	0	1	1	0	0	0	1	1	Horizontal parity
D101	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	Horizontal parity

- When M8161=OFF, specify 16-bit conversion mode.
- When X0=ON, the contents of the 6 data (in 8-bit units n=6 represents the specified D0~D2) starting from the starting number of the register specified by D0 are added, and the total result is stored in the specified by D100 In the register, the level check is stored in D101.

Program example 2



S.	Example of data contents
D0 Low	K100=01100100
D0 high	K111=0110111①
D1 Low	K120=01111000
D1 high	K202=11001010
D2 Low	K123=0111101①
D2 high	K211=1101001①
D100(sum)	K867
D101	0001000①

When the number of "1" is odd, the horizontal parity is "1"
When the number of "1" is even, the horizontal parity is "0".

D100	0	0	0	0	0	0	1	1	0	1	1	0	0	0	1	1	"867" in BCD
D101	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	Horizontal parity

- When M8161=ON, specify 8-bit conversion mode.

◆ When X0=ON, the contents of the 6 data starting from the starting number of the register specified by D0 (in 8-bit units n=6 represents the specified D0~D5) are added up, and the totalization result is stored in the specified by D100 In the register, the level check is stored in D101.

14.6 RS2/Serial Communication 2

This instruction sends and receives data in no-protocol communication by way of serial ports in accordance with RS-232C or RS-485 provided in the main unit.

Instruction	Operand type	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
FNC 87 RS2	**S.** **m** **D.** **n** **n1**	11 steps	RS2	Continuous Operation	—		

Operand	Operand type	Function	
Operand	**S.**	Head device of data registers storing data to be sent **Target Device**: D, R, retouch	16-bit binary or character string
	m	Number of bytes of data to be sent [setting range: 0 to 4,096] **Target Device**: D, R, K, H	16-bit binary
	D.	Head device of data registers storing received data when receiving is completed **Target Device**: D, R, retouch	16-bit binary or character string
	n	Number of bytes to be received [setting range: 0 to 4,096] **Target Device**: D, R, K, H	16-bit binary
	n1	Used channel number [contents of setting:K0=ch 2, K1= ch 3, K2: CAN] **Target Device**: K, H	16-bit binary

Explanation of function and operation

16-bit operation(RS2)

This instruction sends and receives data in no-protocol communication by way of serial ports in accordance with RS-232C or RS-485 provided in the main unit.

Command input

FNC 87 RS2	S·	m	D·	n	n1

● Related devices

Device ch2	Device ch3	Device CAN	Name	Device Ch 2	Device Ch 3	Device CAN	Name
M8122	**M8402**	**M8422**	Sending request	**D8120**	**D8400**	**D8420**	Communication format setting
M8123	**M8403**	**M8423**	Receiving complete flag	**D8121**	**D8401**	**D8421**	Communication mode
M8124	**M8404**	**M8424**	Data receiving	**D8122**	**D8402**	**D8422**	Remaining points of sent data

	M8129	**M8409**	**M8429**	Timeout judgment flag	**D8123**	**D8403**	**D8423**	Receive point monitoring
	M8161			8-bit processing mode	**D8124**	**D8410、D8411**	**D8430、D8431**	Header
					D8125	**D8412、D8413**	**D8432、D8433**	Footer
					D8129	**D8409**	**D8429**	Set timeout
					D8063			

❖ **For detailed configuration and case description, please refer to 2N series PLC/MX2N series PLC/3G series PLC programming manual**

14.7 PID/PID Control Loop

This instruction executes PID control which changes the output value according to the input variation. (Coolmay PLC supports step response method)

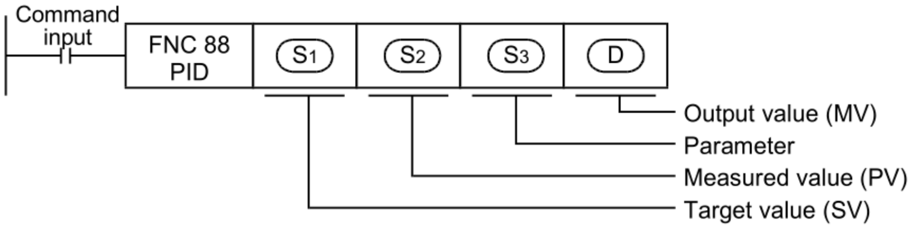
Instruction	Operand type	Function					
FNC 88 PID	**S1. S2. S3. D.**	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
		9 steps	PID	Continuous Operation		—	

Operand	**S1.**	Data register number storing the target value(SV) **Target Device**: D, R	16-bit binary
	S2.	Data register number storing the measured value (PV) **Target Device**: D, R	16-bit binary
	S3.	Data register number storing a parameter **Target Device**: D, R	16-bit binary
	D.	Data register number storing the output value (MV) **Target Device**: D, R	16-bit binary

Explanation of function and operation

1. 16-bit operation(PID)

When the target value **S1.** , measured value **S2.** , and parameters **S3.** ~ **S3.**+6 are set and a program is executed, the operation result (MV) is stored to the output value **D.** at every sampling time **S3.**



2. Set items

Set item		Content	Number of occupied points
S1.	Target value (SV)	Set the target value (SV). PID instruction does not change the contents of setting.	1
S2.	Measured value (PV)	This is the input value in PID control loop	1
S3.	Parameter	Auto tuning: In the case of limit cycle method a)Operation setting (ACT): When bits 1, 2 and 5 are not all "0" Twenty-five devices are occupied from the head device specified in S3. b)Operation setting (ACT): When bits 1, 2 and 5 are all "0" Twenty devices are occupied from the head device specified in S3.	25 20
D.	Output value (MV)	Auto tuning: In the case of step response method Before driving PID instruction, the user should set the initial output value. During auto tuning, PID instruction does not change the MV output.	1

3. List of parameters S3.~S3.+28

Set item			Setting Value	Remarks
S3.	Sampling time(Ts)		1~32767(ms)	It cannot be shorter than the operation cycle.
S3.+1	Operatio-n setting (ACT)	bit0	0: Forward operation 1: Backward operation	Operation direction
		bit1	0: Input variation alarm is invalid. 1: Input variation alarm is valid.	
		bit2	0:Output variation alarm is invalid. 1: Output variation alarm is valid.	Do not set to ON bit 2 and bit 5 at the same time.
		bit3	Not available	
		bit4	0: Auto tuning is not executed.	

				1: Auto tuning is executed.	
			bit5	0: Upper and lower limits of output value are not valid. 1:Upper and lower limits of output value are valid.	Do not set to ON bit 2 and bit 5 at the same time.
			bit6	0: Step response method	Select the auto tuning mode.
			bit7~bit15	Not available	
	S3.+2	Input filter constant(α)		0~99(%)	When "0" is set, the input filter is not provided.
	S3.+3	Proportional gain()		1~32767(%)	
	S3.+4	Integral time()		0~32767(*100ms)	When "0" is set, it is handled as "∞" (no integration).
	S3.+5	Derivative gain ()		0~100(%)	When "0" is set, the derivative gain is not provided.
	S3.+6	Derivative time()		0~32767(*100ms)	When "0" is set, the derivative operation is not executed.
	S3.+7 **...** **S3.+19**	These devices are occupied for internal processing in PID control loop. Do not change the data.			
	S3.+20*1	Input variation (incremental) alarm set value		0~32767	It is valid when bit 1 is set to "1" in S3.+1 for the operation setting (ACT).
	S3.+21*1	Input variation (decremental) alarm set value		0~32767	It is valid when bit 1 is set to "1" in S3.+1 for the operation setting (ACT).
	S3.+22*1	Output variation (incremental) alarm set value		0~32767	It is valid when bit 2 is set to "1" and bit 5 is set to "0" in S3.+1 +1 for the operation setting (ACT).
		Output upper limit set value		-32768~32767	It is valid when bit 2 is set to "0" and bit 5 is set to "1" in S3.+1 +1 for the operation setting (ACT).
	S3.+23*1	Output variation (decremental) alarm set value		0~32767	It is valid when bit 2 is set to "1" and bit 5 is set to "0" in S3.+1 for the operation setting (ACT).
		Output lower limit set value		-32768~32767	It is valid when bit 2 is set to "0" and bit 5 is set to "1" in

				S3.+1 for the operation setting (ACT).
S3.+24*1	Alarm output	bit0	0: Input variation (incremental) is not exceeded. 1: Input variation (incremental) is exceeded.	It is valid when bit 1 is set to "1" or bit 2 is set to "1" in S3.+1 for the operation setting (ACT).
		bit1	0: Input variation (decremental) is not exceeded. 1: Input variation (decremental) is exceeded.	
		bit2	0: Output variation (incremental) is not exceeded. 1: Output variation (incremental) is exceeded.	
		bit3	0:Output variation (decremental) is not exceeded. 1:Output variation (decremental) is exceeded.	

*1: S3.+20~24 are occupied when any bit 1, 2 or 5 is set to "1" in +1 for operation setting (ACT).

- Two or more PID instructions can be executed at the same time. (There is no limitation in the number of loops.) However, make sure that **S3.** , **D.** and other operands specified in each instruction are different to each other.
- Number of devices occupied for parameters starting from **S3.:** In the step response method
  1) Operation setting (ACT): When bits 1, 2 and 5 are not all "0"
     Twenty-five devices are occupied from the head device specified in **S3.**
  2) Operation setting (ACT): When bits 1, 2 and 5 are all "0"
     Twenty devices are occupied from the head device specified in **S3.**

❖ 2N series PLC supports PID, but it does not support auto-tuning, you need to manually adjust the parameters.

❖ More procedures please refer to the official website *Demos of 3G PLC PID Automatic Tuning* and *2N PLC PID output 300℃*

15 Data Transfer 2

FNC NO.	Mnemonic	function	Supported Model		
			3G PLC	2N PLC	MX2N PLC
100	—				
101	—				
102	ZPUSH	Batch Store of Index Registe	★		
103	ZPOP	Batch POP of Index Register	★		
104	—				
105	—				
106	—				
107	—				
108	—				
109	—				

15.1 ZPUSH/Batch Store of Index Register

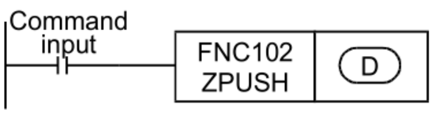
This instruction temporarily batch-stores the present value of the index registers V0 to V7 and Z0 to Z7.

For restoring the present value of temporarily batch-stored index registers, use ZPOP (FNC103) instruction.

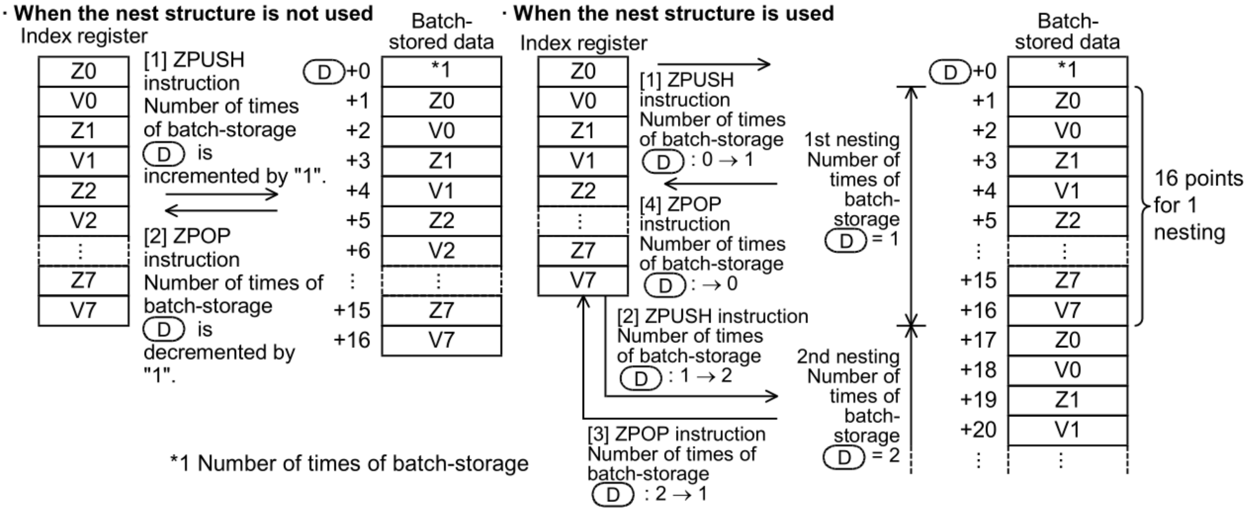
Instruction	Operand Type	Function						
		16-bit Instruction	Mnemonic	Operation Condition		16-bit Instruction	Mnemonic	Operation Condition
FNC102 ZPUSH	**D.**	3 steps	ZPUSH	Continuous Operation			—	
P			ZPUSHP	Pulse (Single) Operation			—	

Operand	**D.**	Head device number batch-storing the present value of the index registers V0 to V7 and Z0 to Z7 **D.** : Number of times of batch-storage **D.+1 ~ D.+16**×Number of times of batch-storage: Batch-stored data storage destination **Target Device**: D, R	BIN16 bit

Explanation of function and operation	**1. 16-bit operation(ZPUSH, ZPUSHP)**

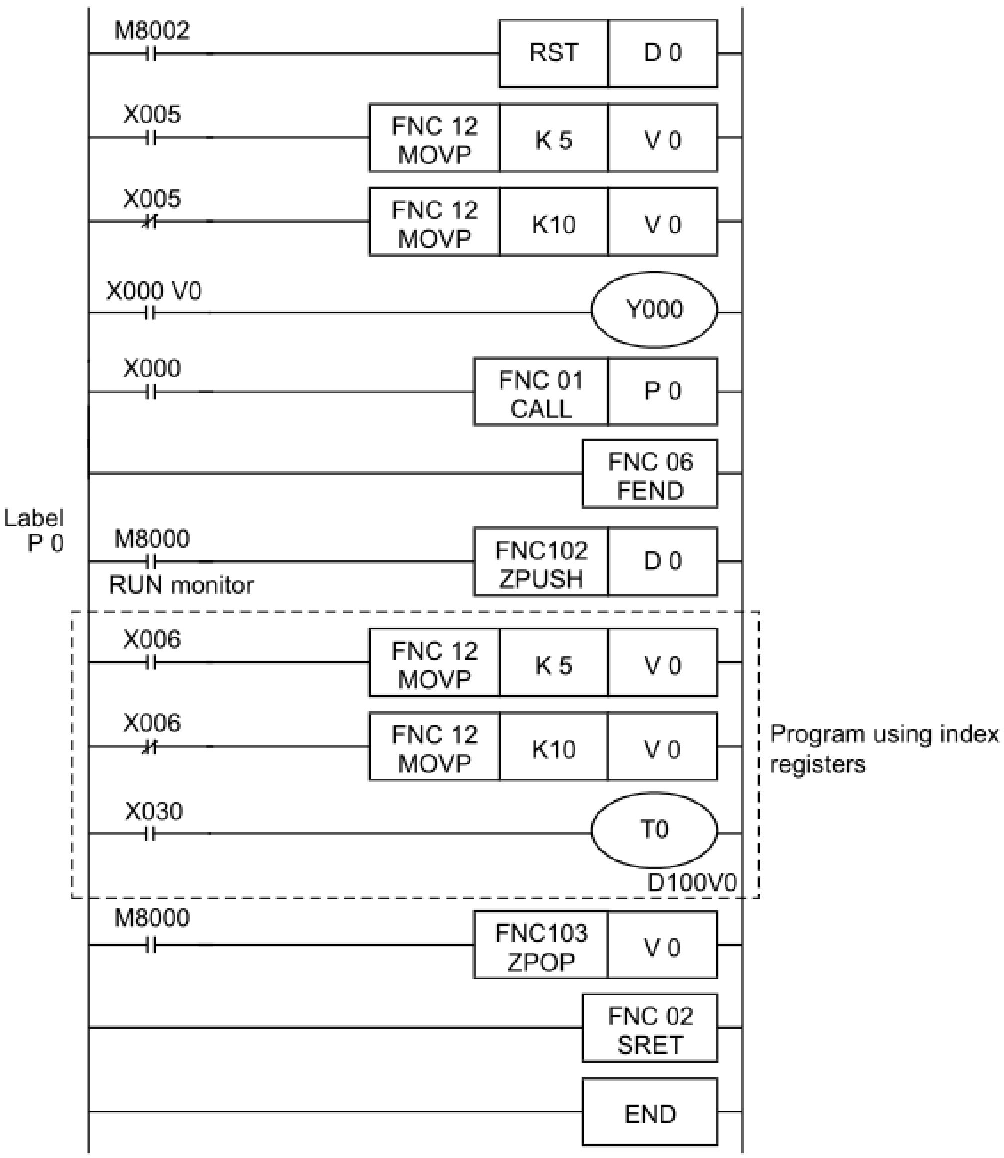


1) The contents of the index registers V0 to V7 and Z0 to Z7 are batch-stored temporarily to **D.** and later.
   When the contents of index registers are batch-stored, the number of times of batch-storage **D.** is incremented by "1".
2) For restoring the batch-stored data, use ZPOP (FNC103) instruction.
   Use ZPUSH (FNC102) and ZPOP (FNC103) instruction as a pair.
3) By specifying a same device to **D.** , ZPUSH (FNC102) and ZPOP (FNC103) instructions can be used in the nest structure.
   In this case, the occupied points are added by "16" after **D.** every time ZPUSH (FNC102) instruction is executed. Secure in advance sufficient area for the number of the next structure.
4) **D.**The figure below shows the data structure batch-stored in **D.**



- When not using the nest structure, clear the number of batch-storage times **D.** before executing ZPUSH (FNC102) instruction.
- When using the nest structure, clear the number of batch-storage times **D.** before executing ZPUSH (FNC102) instruction the first time.

Program example 2

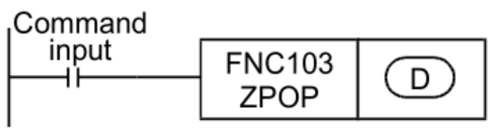


15.2 ZPOP/Batch POP of Index Register

This instruction restores the contents of the index registers V0 to V7 and Z0 to Z7 which were batch-stored temporarily by ZPUSH (FNC102) instruction.

Instruction	Operand Type	Function						
		16-bit Instruction	Mnemonic	Operation Condition		32-bit Instruction	Mnemonic	Operation Condition
FNC103 ZPOP	**D.**	3 steps	ZPOP	Continuous Operation			—	
P			ZPOPP	Pulse (Single) Operation			—	
Operand	**D.**	Head device number temporarily batch-storing the contents of the index registers V0 to V7 and Z0 to Z7 **D.** : Number of times of batch-storage **D.+1 ~ D.+16**×Number of times of batch-storage: Batch-stored data storage destination Target Device: D, R					16-bit binary	

Explanation of function and operation	
	1. 16-bit operation(ZPOP、ZPOPP)



1) The contents of the index registers V0 to V7 and Z0 to Z7 which were batch-stored temporarily to **D.** and later are restored to the original index registers. When the contents of the index registers are restored, the number of times of batch-storage **D.** is decremented by "1".

2) For temporarily batch-storing the data, use ZPUSH (FNC102) instruction.
Use ZPUSH (FNC102) and ZPOP (FNC103) instruction as a pair.

- When there is no nesting action, please clear the batch save times D. before executing the ZPUSH (FNC 102) instruction.
- When there are nested actions, please clear the batch save times D. before the first execution.

❖ **Please refer to the program case 15.1**

16 Floating Point

FNC NO.	Mnemonic	Function	Supported Model		
			3G PLC	2N PLC	MX2N PLC
110	ECMP	Floating Point Compare	★	★	★
111	EZCP	Floating Point Zone Compare	★	★	★
112	EMOV	Floating Point Move	★		
113	—				
114	—				
115	—				
116	ESTR	Floating Point to Character String Conversion	★		
117	EVAL	Character String to Floating Point Conversion	★		
118	EBCD	Binary Floating Point to Decimal Floating Point Conversion	★	★	★
119	EBIN	Decimal Floating Point to Binary Floating Point	★	★	★
120	EADD	Floating Point Addition	★	★	★
121	ESUB	Floating Point Subtraction	★	★	★
122	EMUL	Floating Point Multiplication	★	★	★
123	EDIV	Floating Point Division	★	★	★
124	EXP	Floating Point Exponent	★		
125	LOGE	Floating Point Natural Logarithm	★		
126	LOG10	Floating Point Common Logarithm	★		
127	ESQR	Floating Point Square Root	★	★	★
128	ENEG	Floating Point Negation	★		
129	INT	Floating Point to Integer Conversion	★	★	★
130	SIN	Floating Point Sine	★	★	★
131	COS	Floating Point Cosine	★	★	★
132	TAN	Floating Point Tangent	★	★	★
133	ASIN	Floating Point Arc Sine	★		
134	ACOS	Floating Point Arc Cosine	★		
135	ATAN	Floating Point Arc Tangent	★		
136	RAD	Floating Point Degrees to Radians Conversion	★		
137	DEG	Floating Point Radians to Degrees Conversion	★		

16.1 ECMP/Floating Point Compare

This instruction compares two data (binary floating point), and outputs the result (larger, same or smaller) to three single bit devices.

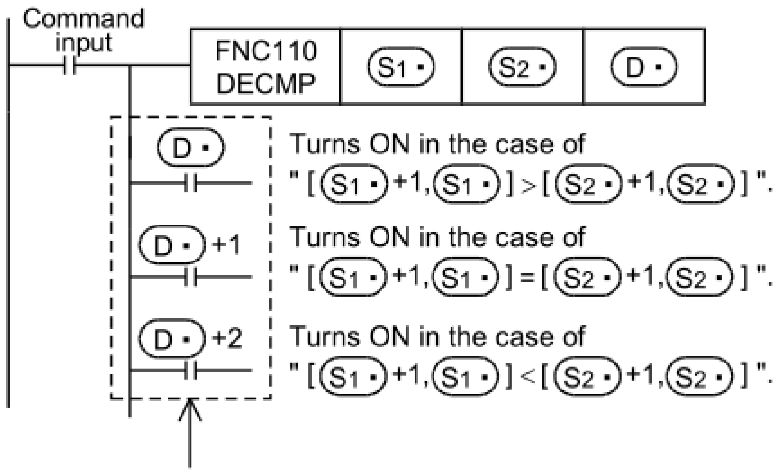
Instruction	Operand Type	Function					
D FNC110 ECMP P	S1. S2. D.	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
			—		13 steps	DECMP	Continuous Operation
						DECMPP	Pulse (Single) Operation

Operand	**S1.**	Device number storing binary floating point data to be compared Target Device: D, R, K, H, E, retouch	Real number (binary)
	S2.	Device number storing binary floating point data to be compared Target Device: D, R, K, H, E, retouch	Real number (binary)
	D.	Head bit device number to which the comparison result is output (Three devices are occupied.) Target Device: Y, M, S, retouch	bit

Explanation of function and operation

32-bit operation (DECMP, DECMPP)

The comparison value**[S1.+1, S1.]** is compared with the comparison source **[S2.+1, S2.]** as floating point data, and either bit among **D.**、**D.+1**、**D.+2** turns ON according to the result (smaller, same or larger).



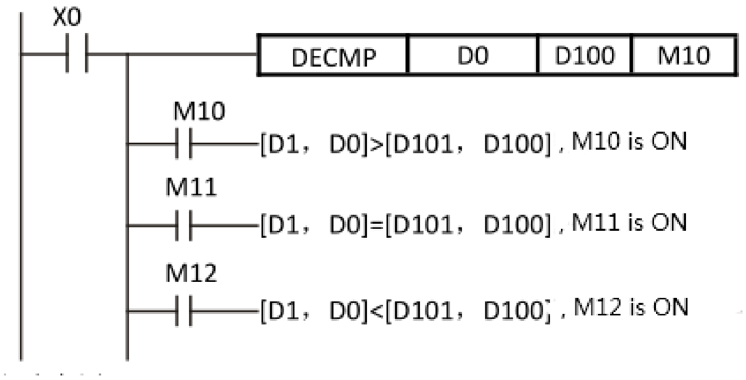
Even if the command input turns OFF and DECMP instruction is not executed, D. ~ D.+2 hold the status

before the please rewrite this command input turned OFF.

- When a constant (K or H) is specified as **[S1.+1, S1.]**、**[S2.+1, S2.]**, it is automatically converted from binary into binary floating point (real number) when the instruction is executed.

	● Three devices are occupied from **[D., D.+1, D.+2]**

Program example



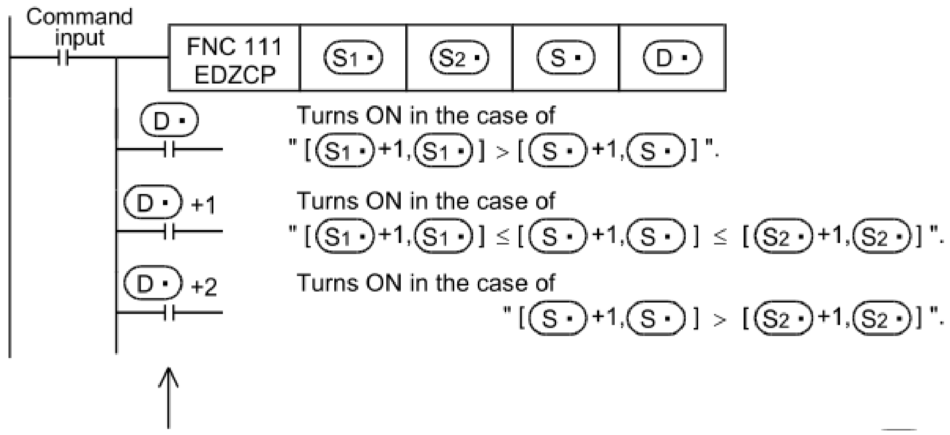
- The designated device is M10, which automatically occupies M10~M12
- When X0=ON, the DECMP instruction is executed, and one of M10~M12 will be ON. When X0=OFF, the DECMP instruction will not be executed, and the state of M10~M12 will remain in the state before X0=OFF....
- If you need to get ≥, ≤, ≠, you can get M0~M2 in series and parallel.
- To clear the comparison result, please use RST or ZRST instruction.

16.2 EZCP/Floating Point Zone Compare

This instruction compares data (binary floating point) with two values (one zone), and outputs the comparison result to three single bit devices.

<table>
<tr><th>Instruction</th><th>Operand Type</th><th colspan="6">Function</th></tr>
<tr><td rowspan="3">D **FNC111 EZCP** P</td><td rowspan="3">**S1.**
S2.
S.
D.</td><td>16-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td><td>32-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td></tr>
<tr><td rowspan="2"></td><td rowspan="2">—</td><td rowspan="2"></td><td rowspan="2">17 steps</td><td>DEZCP</td><td>Continuous Operation</td></tr>
<tr><td>DEZCPP</td><td>Pulse (Single) Operation</td></tr>
<tr><td rowspan="4">Operand</td><td>**S1.**</td><td colspan="5">Data register number storing binary floating point data to be compared
Target Device: D, R, K, H, E, retouch</td><td>Real number (binary)</td></tr>
<tr><td>**S2.**</td><td colspan="5">Data register number storing binary floating point data to be compared
Target Device: D, R, K, H, E, retouch</td><td>Real number (binary)</td></tr>
<tr><td>**S.**</td><td colspan="5">Data register number storing binary floating point data to be compared
Target Device: D, R, K, H, E, retouch</td><td>Real number (binary)</td></tr>
<tr><td>**D.**</td><td colspan="5">Head bit device number to which the comparison result is output (Three devices are occupied.)
Target Device: Y, M, S, retouch</td><td>bit</td></tr>
<tr><td>**Explanation of function and operation**</td><td colspan="7">**32-bit operation(DEZCP、DEZCPP)**
The comparison values **[S1.+1, S1.]**、**[S2.+1, S2.]** are compared with the comparison source</td></tr>
</table>

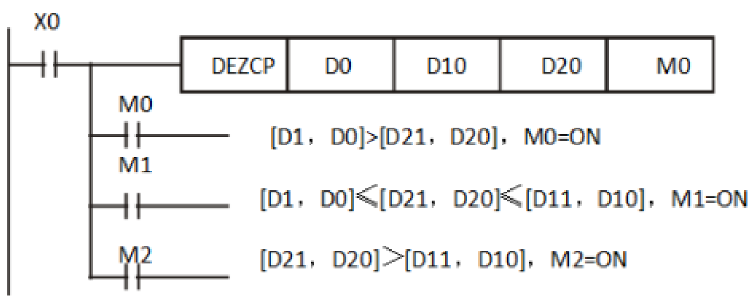
[S.+1, S.] as floating point data, and either bit among **D.**、 **D.+1**、 **D.+2** turns ON according to the result (smaller, same or larger).



Even if the command input turns OFF and DEZCP instruction is not executed, D. ~ D.+2 hold the status before the command input turned OFF.

- When the constants (K, H) are specified in **[S1.+1, S1.], [S2.+1, S2.], [S.+1, S.]**, the value is automatically converted from BIN to 2. After processing the floating point number, process it.
- Three devices are occupied from **[D., D.+1, D.+2]**
- Make sure that two comparison values have the following relationship:**[S1.+1, S1.]≤[S2.+1, S2.]**. In the case of **[S1.+1, S1.] > [S2.+1, S2.]**, the value **[S2.+1, S2.]** is regarded as **[S1.+1, S1.]** value during comparison

Program example



◆ The designated device is M0, which automatically occupies M0~M2. To clear the comparison result, use the RST or ZRST instruction.

◆ When X0=ON, the DEZCP instruction is executed, and one of M0~M2 turns ON; when X0=OFF, the DEZCP instruction is not executed, and the state of M0~M2 remains in the state before X0=OFF.

16.3 EMOV/Floating Point Move

This instruction transfers binary floating point data.

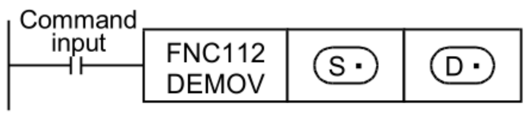
<table>
<tr><th colspan="3">Instruction</th><th>Operand Type</th><th colspan="8">Function</th></tr>
<tr><td rowspan="3">D</td><td rowspan="3">FNC112
EMOV</td><td rowspan="3">P</td><td rowspan="3">S.
D.</td><td>16-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td><td></td><td>32-bit Instruction</td><td>Mnemonic</td><td colspan="2">Operation Condition</td></tr>
<tr><td rowspan="2"></td><td rowspan="2">—</td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2">9 steps</td><td>DEMOV</td><td colspan="2">Continuous Operation</td></tr>
<tr><td>DEMOVP</td><td colspan="2">Pulse (Single) Operation</td></tr>
<tr><td colspan="3" rowspan="2">Operand</td><td>S.</td><td colspan="7">Binary floating point data (transfer source) or device number storing data
Target Device: D, R, E, retouch</td><td>Real number (binary)</td></tr>
<tr><td>D.</td><td colspan="7">Device number receiving floating point data
Target Device: D, R, retouch</td><td>Real number (binary)</td></tr>
</table>

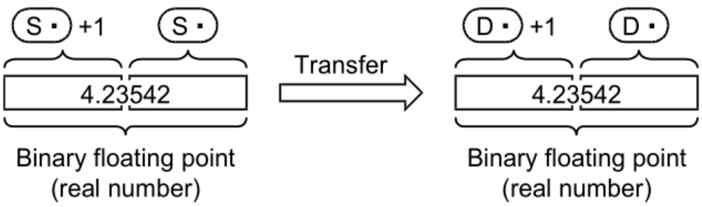
Explanation of function and operation	

Explanation of function and operation

32 bit operation(DEMOV、DEMOVP)

The contents (binary floating point data) of the transfer source**[S.+1, S.]** are transferred to **[D.+1, D.]**. A real number (E) can be directly specified as **S.**





Program example

◆ When X0=OFF, the content of (D11, D10) does not change. If X0=ON, the current value of E3.1415926 floating point number is transferred to the (D11, D10) data register.

16.4 ESTR/Floating Point to Character String Conversion

This instruction converts binary floating point data into a character string (ASCII codes) having a specified number of digits.

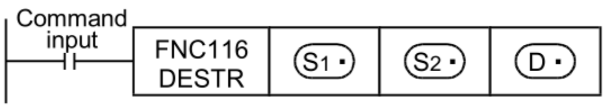
On the other hand, STR (FNC200) instruction converts binary data into a character string (ASCII codes).

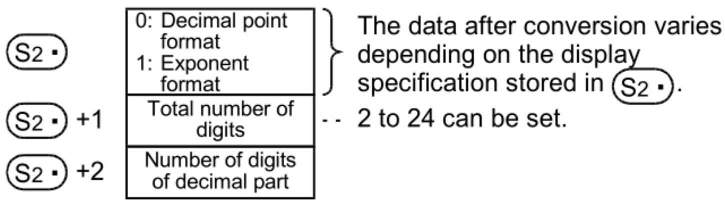
Instruction	Operand type	Function					
		16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
D **FNC116 ESTR** P	**S1.** **S2.** **D.**		—		13 steps	DESTR	Continuous Operation
						DESTRP	Pulse (Single) Operation
Operand	**S1.**	Binary floating point data to be converted or device storing data **Target Device**: D, R, E, retouch					Real number (binary)
	S2.	Head device number storing the display specification of a numeric value to be converted **Target Device**: KnX, KnY, KnM, KnS, T, C, D, R, retouch					16-bit binary
	D.	Head device number storing converted character string **Target Device**: KnY, KnM, KnS, T, C, D, R, retouch					Character string

Explanation of function and operation

1. **32-bit operation(DESTR、DESTRP)**

The contents (binary floating point data) of **[S1.+1, S1.]**are converted into a character string according to the contents specified by **S2., S2.+1, S2.+2**, and then stored to devices **D.** and later. A real number can be directly specified as **S1.**





2. **In the case of decimal point format**

1) The total number of digits which can be specified by **S2.+1** is as follows (24 digits maximum):

When the number of digits of the decimal part is "0", Total number of digits ≥ 2

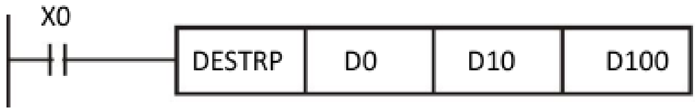
When the number of digits of the decimal part is any value other than "0", Total number of digits ≥ (Number of digits of decimal part + 3)

2) The number of digits of the decimal part which can be specified by **S2.+2** is from 0 to 7.

However, the following must be satisfied, "Number of digits of decimal part ≤ (Total number of

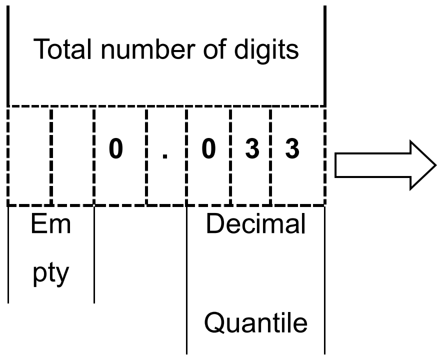
	digits - 3)" **3. In the case of exponent format** 1) The total number of digits which can be specified by**S2.+1** is as follows (24 digits maximum): When the number of digits of the decimal part is "0" Total number of digits ≥ 6 When the number of digits of the decimal part is any value other than "0" Total number of digits ≥ (Number of digits of decimal part + 7) 2) The number of digits of the decimal part which can be specified by **S2.+2** is from 0 to 7. However, the following must be satisfied, "Number of digits of decimal part ≤ (Total number of digits - 7)"

Program example 1



◆ When X0=ON, the contents of D0 and D1 (binary floating point data) are converted according to the contents specified in D10 to D12, and the program stored in the device after D100 is converted.

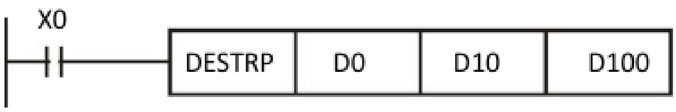
D10	0	Conversion form
D11	3	All digits
D12	7	Decimal places



	b15——b8	b7——b0
D100	20H(space)	20H(space)
D101	2EH(.)	30H(0)
D102	33H(3)	30H(0)
D103	00H	33H(3)

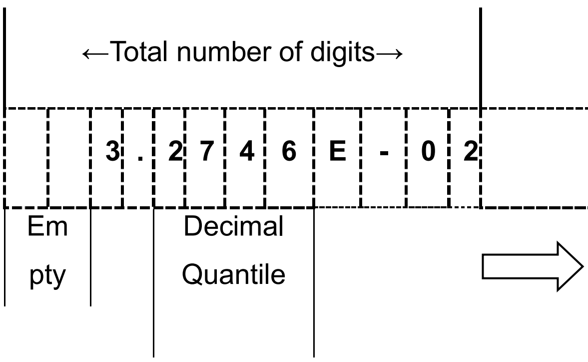
D0=0.0327457

Program example 2



◆ When X0=ON, the contents of D0 and D1 (binary floating point data) are converted according to the contents specified in D10 to D12, and the program stored in the device after D100 is converted.

D10	1	Conversion form
D11	12	All digits
D12	4	Decimal places



	b15——b8	b7——b0
D100	20H(space)	20H(space)
D101	2EH(.)	33H(3)
D102	37H(7)	32H(2)
D103	36H(6)	34H(4)
D104	2DH(-)	45H(E)
D105	32H(2)	30H(0)
D106	0000H	

D0=0.0327457

16.5 EVAL/ Character String to Floating Point Conversion

This instruction converts a character string (ASCII codes) into binary floating point data.

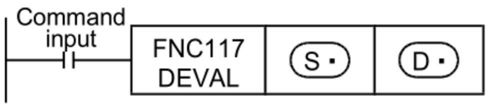
On the other hand, the VAL (FNC201) instruction converts a character string (ASCII codes) into binary data.

Instruction format	Operand Type	Function						
D FNC 117 EVAL P	S. D.	16-bit Instruction	Mnemonic	Operation Condition		32-bit Instruction	Mnemonic	Operation Condition
			—			9 steps	DEVAL	Continuous Operation
							DEVALP	Pulse Operation
Operand	S.	Head device number storing character string data to be converted into binary floating point data **Target Device**: KnX, KnY, KnM, KnS, T, C, D, R, retouch						Character string
	D.	Head device number storing converted binary floating point data **Target Device**: D, R, retouch						Real number (binary)

Explanation of function and operation

1. 32-bit operation(DEVAL, DEVALP)

A character string stored in and later is converted into binary floating point, and stored to **[D.+1, D.]**



- A specified character string may be in the decimal point format or exponent format. A character string in either format can be converted into binary floating point data.

	b15——b8	b7——b0
S.	ASCII code for 1st character	ASCII code for sign
S.+1	ASCII code for 3rd character	ASCII code for 2nd character
S.+2	ASCII code for 5th character	ASCII code for 4th character
S.+3	ASCII code for 7th character	ASCII code for 6th character
S.+4		00H(Indicates the end of the character string.)

→

D.+1	D.

Binary floating point (real number)

a) In the case of decimal point format

	b15 ------ b8	b7 ------ b0
S·	31H(1)	2DH(-)
S· +1	30H(0)	2EH(.)
S· +2	38H(8)	37H(7)
S· +3	32H(2)	31H(1)
S· +4		00H

- 1 . 0 7 8 1 2

D· +1 D·

–1.07812

Binary floating point (real number)

b) In the case of exponent format

	b15 ------ b8	b7 ------ b0
S·	20H(space)	2DH(-)
S· +1	2EH(.)	31H(1)
S· +2	32H(2)	33H(3)
S· +3	31H(1)	30H(0)
S· +4	2BH(+)	45H(E)
S· +5	30H(0)	31H(1)
S· +6		00H

- 1 . 3 2 0 1 E + 1 0

D· +1 D·

–1.3201E+10

Binary floating point (real number)

When a character string to be converted into binary floating point specified by **S.** has 7 digits or more excluding the sign, decimal point and exponent part, the conversion result may contain rounding error.

a) In the case of decimal point format

	b15 ------ b8	b7 ------ b0
S·	20H(space)	2DH(-)
S· +1	31H(1)	20H(space)
S· +2	33H(3)	2EH(.)
S· +3	31H(1)	30H(0)
S· +4	36H(6)	35H(5)
S· +5	31H(1)	38H(8)
S· +6	00H	32H(2)

- 1 . 3 0 1 5 6 8 1 2

Rounded.

D· +1 D·

–1.30156

Up to 6 digits are securely equivalent.

Binary floating point (real number)

b) In the case of exponent format

	b15 — b8	b7 — b0
S·	20H(.)	2DH(-)
S· +1	2EH(.)	31H(1)
S· +2	35H(5)	33H(3)
S· +3	33H(3)	30H(0)
S· +4	31H(1)	34H(4)
S· +5	45H(E)	32H(2)
S· +6	30H(0)	2DH(-)
	00H	32H(2)

- 1 . 3 5 0 3 4 1 2 E - 0 2

Rounded.

⇒ D· +1 D·

–1.35034E–2

Up to 6 digits are securely equivalent.

Binary floating point (real number)

- When "2BH (+)" is specified as the sign in the floating point format or when the sign is omitted, a character string is converted into a positive value.
  When "2DH (-)" is specified as the sign, a character string is converted into a negative value.
- When "2BH (+)" is specified as the sign in the exponent format or when the sign is omitted, a character string is converted into a positive exponent.
  When "2DH (-)" is specified as the sign, a character string is converted into a negative exponent.
- When "20H (space)" or "30H (0)" exists between numbers except the first "0" in a character string specified by **S.** , "20H" or "30H" is ignored during conversion.

	b15 — b8	b7 — b0
S·	20H(space)	2DH(-)
S· +1	31H(1)	30H(0)
S· +2	32H(2)	2EH(.)
S· +3	31H(1)	33H(3)
S· +4		00H

- 0 1 . 2 3 1

Ignored.

⇒ D· +1 D·

–1.231

Binary floating point (real number)

- When "30H (0)" exists between a number and "E" in a character string in the exponent format, "30H" is ignored during conversion.

b15 — b8	b7 — b0
20H(space)	2DH(-)
2EH(.)	31H(1)
34H(4)	30H(0)
33H(3)	35H(5)
2BH(+)	45H(E)
33H(3)	30H(0)
	00H

- 1 . 0 4 5 3 E + 0 3

Ignored.

⇒ D· +1 D·

–1.0453E+3

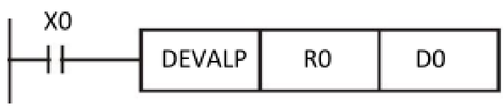
- A character string can consist of up to 24 characters.
  "20H (space)" and "30H (0)" in a character string are counted as one character respectively.

2. Related devices

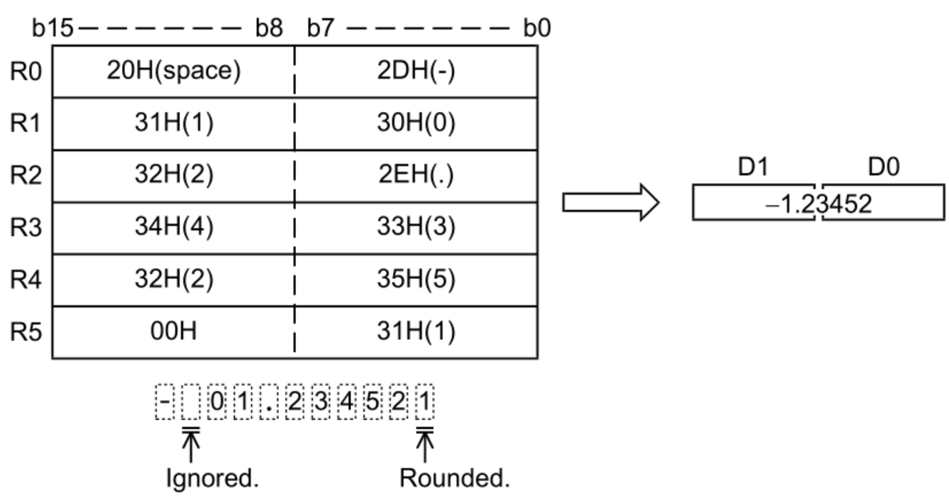
Device	Name	Description	
		Condition	Operation
M8020	Zero flag	The conversion result is true "0". (The mantissa part is "0".)	The zero flag M8020 turns ON.
M8021	Borrow flag	The absolute value of the conversion result is less than "2^{-126}".	The value of **D.** is the minimum value (2^{-126}) of 32-bit real numbers and the borrow flag M8021 turns ON.
M8022	Carry flag	The absolute value of the conversion result is not less than "2^{128}".	The value of **D.** is the maximum value (2^{128}) of 32-bit real numbers and the carry flag M8022 turns ON.

- An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067.
  • When any character other than "30H (0)" to "39H (9)" exists in the integer part or decimal part (error code: K6706)
  • When "2EH (.)" exists in two or more positions in a character string specified by **S.** (error code: K6706)
  • When any character other than "45H (E)", "2BH (+)" or "2DH (-)" exists in the exponent part, or when two or more exponent parts exist (error code: K6706)
  • When "00H" does not exist in the corresponding device range starting from **S.** (error code: K6706)
  • When the number of characters after **S.** is "0" or more than "24" (error code: K6706)

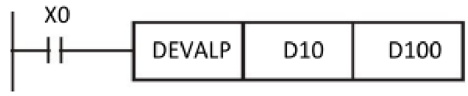
Program example 1



◆ In the program example shown below, a character string stored in R0 and later is converted into binary floating point, and stored to D0 and D1 when X0 turns ON

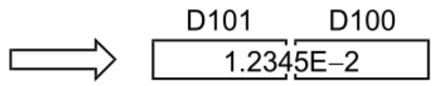


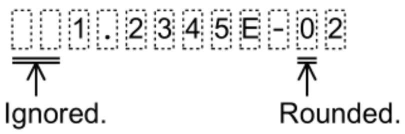
Program example 2



◆ In the program shown below, a character string stored in D10 and later is converted into binary floating point, and stored to D100 and D101 when X0 turns ON

	b15 — — — — — — b8	b7 — — — — — — b0
D10	20H(space)	20H(space)
D11	2EH(.)	31H(1)
D12	33H(3)	32H(2)
D13	35H(5)	34H(4)
D14	2DH(-)	45H(E)
D15	32H(2)	30H(0)
D16		00H





16.6 EBCD/Floating Point to Scientific Notation Conversion

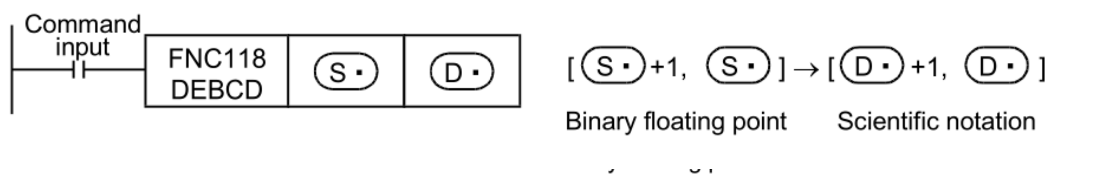
This instruction converts binary floating point into scientific notation.

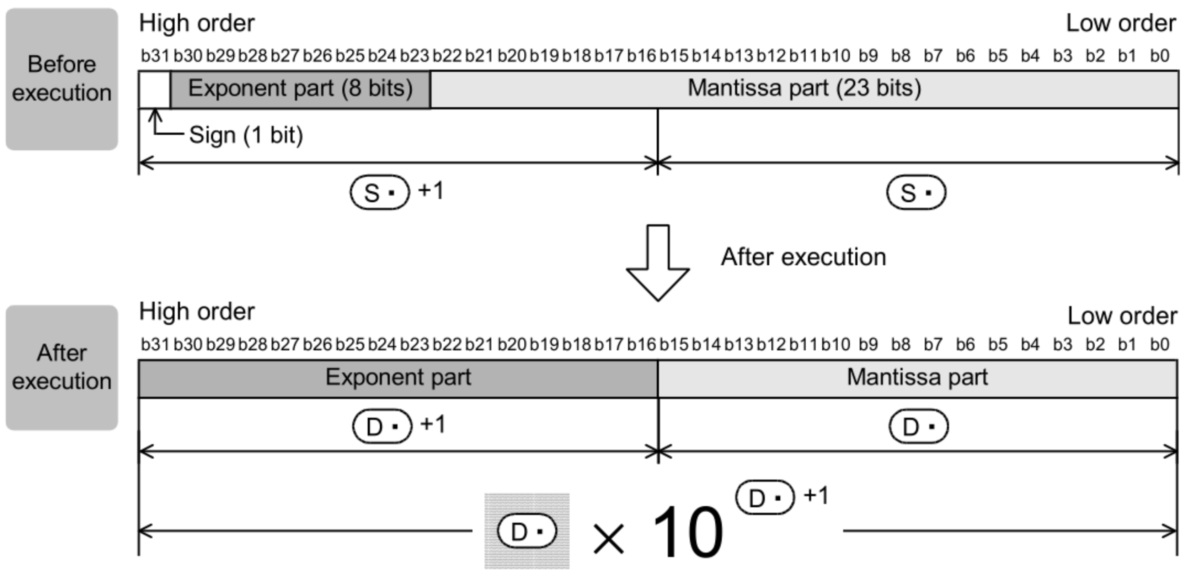
Instruction format	Operand Type	Function						
D / FNC118 EBCD / P	S. D.	16-bit Instruction	Mnemonic	Operation Condition		32-bit Instruction	Mnemonic	Operation Condition
			—			9 steps	DEBCD	Continuous Operation
							DEBCDP	Pulse Operation
Operand	S.	Data register number storing binary floating point **Target Device**: D, R, retouch						Real number (binary)
	D.	Data register number storing converted scientific notation **Target Device**: D, R, retouch						Real number (decimal)

Explanation of function and operation

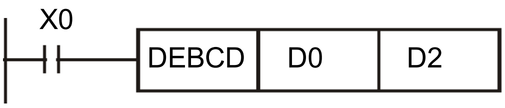
1. 32-bit operation(DEBCD、DEBCDP)

Binary floating point stored in **[S.+1, S.]**is converted into scientific notation, and transferred to **[D.+1, D.]**.





Program example



- When X0=ON, the binary floating-point numbers in D1 and D0 are converted into decimal floating-point numbers and stored in D3 and D2.
- Binary floating point number [D1, D0] The real number is 23 bits, the exponent is 8 bits, and the sign bit is 1 bit After the conversion, the decimal floating point number [D3, D2] is represented by the mathematical formula → [D2] X 10[D3]

16.7 EBIN/ Scientific Notation to Floating Point Conversion

This instruction converts scientific notation stored in devices into binary floating point.

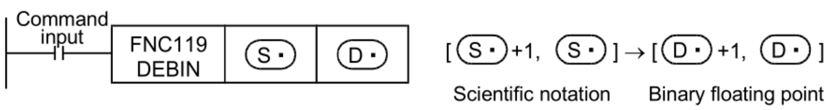
Instruction format	Operand Type	Function: 16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
D FNC119 EBIN P	S. D.		—		9 steps	DEBIN	Continuous Operation
						DEBINP	Pulse (Single) Operation

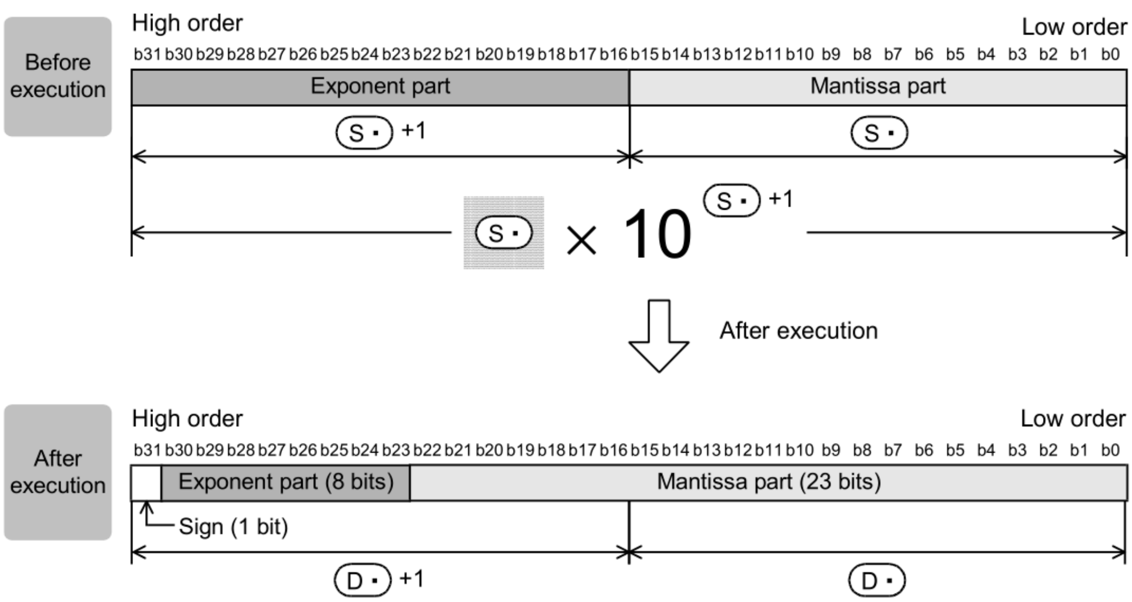
Operand			
Operand	S.	Data register number storing scientific notation data **Target Device**: D, R, retouch	Real number (decimal)
	D.	Data register number storing converted binary floating point. **Target Device**: D, R, retouch	Real number (binary)

Explanation of function and operation

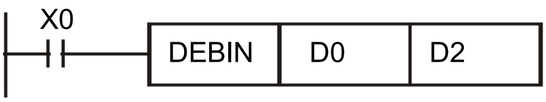
32-bit operation(DEBIN、 DEBINP)

Scientific notation stored in **[S.+1, S.]** is converted into binary floating point, and transferred to **[D.+1, D.]**.





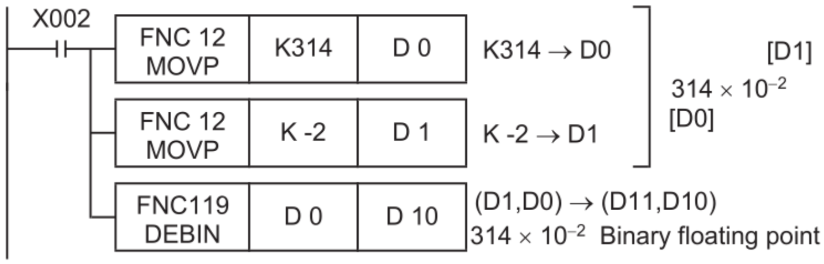
Program example 1



- When X0=ON, the decimal floating-point numbers in D1 and D0 are converted into binary floating-point numbers and stored in D3 and D2.

◆ Decimal floating point number [D1, D0] mathematical expression →[D2] X 10[D3]
After the conversion, the binary floating-point number has 23 real numbers, 8 exponents, and 1 sign

Program example 2



◆ The FLT instruction must be used to convert BIN integers into binary floating-point numbers before performing floating-point operations. The premise of the conversion is that the converted value must be a BIN integer. However, the DEBIN instruction can convert floating-point values into binary floating-point numbers.

◆ By DEBIN instruction, a numeric value containing the decimal point can be directly converted into binary floating point.

◆ When X0=ON, move K3,140 to D0, and move K-3 to D1 to form a decimal floating point number type ($3.14=3{,}140\times10^{-3}$)

16.8 EADD/Floating Point Addition

This instruction executes addition of two binary floating point data.

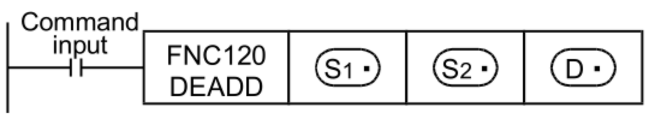
Instruction format	Operand Type	Function						
		16-bit Instruction	Mnemonic	Operation Condition		32-bit Instruction	Mnemonic	Operation Condition
D **FNC120 EADD** P	**S1.** **S2.** **D.**		—			13 steps	DEADD DEADDP	Continuous Operation Pulse (Single) Operation

Operand	**S1.**	Word device number storing binary floating point data used in addition **Target Device**: D, R, K, H, E, retouch	Real number (binary)
	S2.	Word device number storing binary floating point data used in addition **Target Device**: D, R, K, H, E, retouch	
	D.	Data register number storing the addition result **Target Device**: D, R, retouch	

Explanation of function and operation

32-bit operation(DEADD、DEADDP)

Binary floating point data **[S1.+1, S1.]** is added to binary floating point data **[S2.+1, S1.]**, and the addition result in the binary floating point format is transferred to**[D.+1, D.]**.



[S1 • +1, S1 •] + [S2 • +1, S2 •] → [D • +1, D •]

Binary floating point   Binary floating point   Binary floating point

- When a constant (K or H) is specified as**[S1.+1, S1.]** or **[S2.+1, S1.]**, it is automatically converted into binary floating point.
- The same device number can be specified in **[S1.+1, S1.]**,**[S2.+1, S1.]** and **[D.+1, D.]**.

In this case, note that the addition result changes in every operation cycle when the continuous operation
type instruction (DEADD) is used.

16.9 ESUB/ Floating Point Subtraction

This instruction executes subtraction of two binary floating point data.

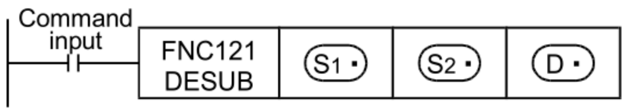
Instruction format	Operand Type	Function						
		16-bit Instruction	Mnemonic	Operation Condition		32-bit Instruction	Mnemonic	Operation Condition
D **FNC121 ESUB** P	**S1.** **S2.** **D.**		—			13 steps	DESUB	Continuous Operation
							DESUBP	Pulse Operation

Operand	**S1.**	Word device number storing binary floating point data used in subtraction **Target Device**: D, R, K, H, E, retouch	Real number (binary)
	S2.	Word device number storing binary floating point data used in subtraction **Target Device**: D, R, K, H, E, retouch	
	D.	Data register number storing the subtraction result **Target Device**: D, R, retouch	

Explanation of function and operation

32-bit operation (DESUB、DESUBP)

Binary floating point data **[S1.+1, S1.]** is subtracted from binary floating point data**[S2.+1, S1.]**, and
the subtraction result in the binary floating point format is transferred to **[D.+1, D.]**.



[S1 •+1, S1 •] − [S2 •+1, S2 •] → [D •+1, D •]

Binary floating point   Binary floating point   Binary floating point

- When a constant (K or H) is specified as **[S1.+1, S1.]** or **[S2.+1, S1.]**, it is automatically converted into binary floating point.
- A same device number can be specified in **[S1.+1, S1.]**,**[S2.+1, S1.]**and **[D.+1, D.]**.
  In this case, note that the subtraction result changes in every operation cycle when the continuous operation type instruction (DESUB) is used.

16.10 EMUL/ Floating Point Multiplication

This instruction executes multiplication of two binary floating point data.

Instruction format	Operand Type	Function					
		16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
D FNC122 EMUL P	S1. S2. D.		—		13 steps	DEMUL	Continuous Operation
						DEMULP	Pulse (Single) Operation

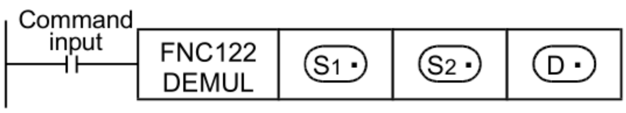
Operand			
Operand	S1.	Word device number storing binary floating point data used in multiplication **Target Device:** D, R, K, H, E, retouch	Real number (binary)
	S2.	Word device number storing binary floating point data used in multiplication **Target Device** : D, R, K, H, E, retouch	
	D.	Data register number storing the multiplication result **Target Device**: D, R, retouch	

Explanation of function and operation	
	32-bit operation(DEMUL、DEMULP)

Explanation of function and operation

32-bit operation(DEMUL、DEMULP)

Binary floating point data **[S1.+1, S1.]** is multiplied by binary floating point data **[S2.+1, S1.]**, and the multiplication result in the binary floating point format is transferred to **[D.+1, D.]**.



[S1· +1, S1·] × [S2· +1, S2·] → [D· +1, D·]

Binary floating point   Binary floating point   Binary floating point

- When a constant (K or H) is specified as **[S1.+1, S1.]** or **[S2.+1, S1.]**, it is automatically converted into binary floating point.
  [S1.+1, S1.] and **[S2.+1, S1.]** and **[D.+1, D.]** can also specify the same device number.
  At this time, if a continuous execution instruction (DEMUL) is used, please note the result of the multiplication operation will change every operation cycle.

16.11 EDIV/ Floating Point Division

This instruction executes division of two binary floating point.

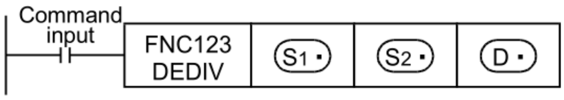
Instruction format	Operand Type	Function						
		16-bit Instruction	Mnemonic	Operation Condition		32-bit Instruction	Mnemonic	Operation Condition
D FNC123 EDIV P	**S1.** **S2.** **D.**		—			13steps	DEDIV	Continuous Operation
							DEDIVP	Pulse (Single) Operation

Operand			
Operand	**S1.**	Word device number storing binary floating point data used in division **Target Device**: D, R, K, H, E, retouch	实数(2 进制)
	S2.	Word device number storing binary floating point data used in division **Target Device**: D, R, K, H, E, retouch	
	D.	Data register number storing binary floating point data obtained by division **Target Device**: D, R, retouch	

Explanation of function and operation

32-bit operation (DEDIV、DEDIVP)

Binary floating point data **[S1.+1, S1.]** is divided by binary floating point data **[S2.+1, S1.]**, and the division result in the binary floating point format is transferred to **[D.+1, D.]**.



Dividend / Divisor

[S1·+1, S1·] ÷ [S2·+1, S2·] → [D·+1, D·]

Binary floating point / Binary floating point / Binary floating point

- When a constant (K or H) is specified as **[S1.+1, S1.]**or **[S2.+1, S1.]**, it is automatically converted into binary floating point.

[S1.+1, S1.] and [S2.+1, S1.] and **[D.+1, D.]** can also specify the same device number.

At this time, if a continuous execution instruction (DEDIV) is used, please note the result of the division operation will change every operation cycle.

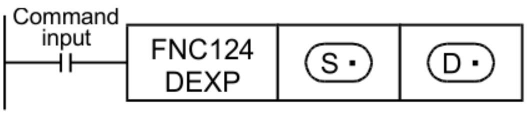
16.12 EXP/ Floating Point Exponent

This instruction executes exponential operation whose base is "e (2.71828)".

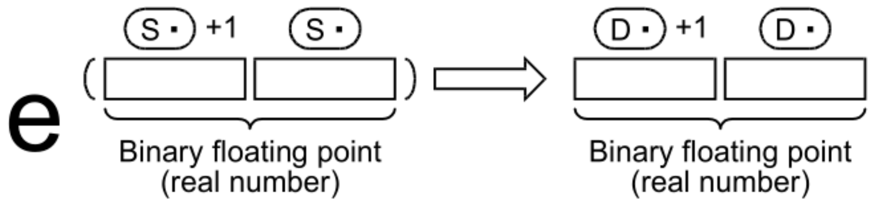
Instruction format	Operand Type	Function					
		16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
D **FNC124 EXP** P	**S.** **D.**		—		9 steps	DEXP	Continuous Operation
						DEXPP	Pulse (Single) Operation

Operand	**S.**	Head device number storing binary floating point data used in exponential operation. **Target Device**: D, R, E, retouch	Real number (binary)
	D.	Head device number storing the operation result. **Targer Device**: D, R, retouch	

Explanation of function and operation	**32-bit operation(DEXP、DEXPP)** The exponent of **[S.+1, S.]** is calculated, and the operation result is stored to **[D.+1, D.]**.



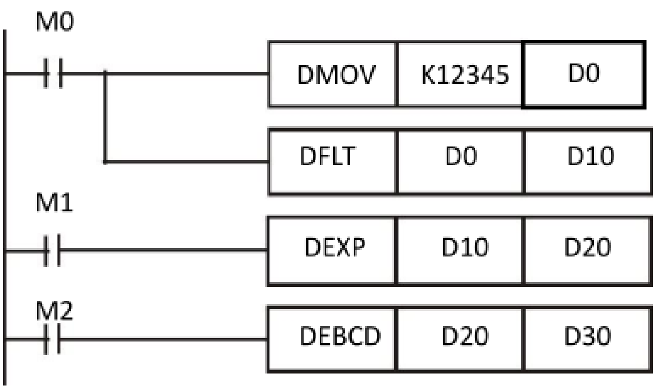
In the exponential operation, the base (e) is set to "2.71828".



- An operation error occurs in the following case; The error flag M8067 turns ON, and the error code is stored in D8067. (error code: K6706)

$2^{-126} \leqq |\text{Operation result}| < 2^{128}$

Program example



- When M0 is ON, convert the value of (D1, D0) into a binary floating-point number and store it in the (D11, D10) register.
- When M1 is ON, (D11, D10) is an exponent for EXP operation, and its value is a binary floating point value and stored in the (D21, D20) register.
- When M2 is ON, convert the (D21, D20) binary floating point value into a decimal floating point value and store it in the (D31, D30) register. (At this time D31 is the 10th power of D30)

16.13 LOGE/ Floating Point Natural Logarithm

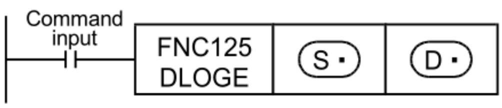
This instruction executes the natural logarithm operation.

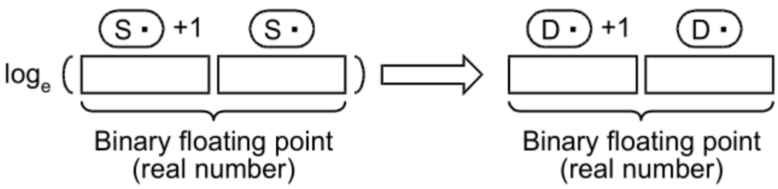
Instruction format	Operand Type	Function						
		16-bit Instruction	Mnemonic	Operation Condition		32-bit Instruction	Mnemonic	Operation Condition
D FNC 125 LOGE P	S. D.		—			9 steps	DLOGE	Continuous Operation
							DLOGEP	Pulse (Single) Operation
Operand	S.	Head device number storing binary floating point data used in the natural logarithm operation **Target Device**: D, R, E, retouch						Real number (binary)
	D.	Head device number storing the operation result **Target Device**: D, R, retouch						

Explanation of function and operation	32-bit operation(DLOGE、DLOGEP)

32-bit operation(DLOGE、DLOGEP)

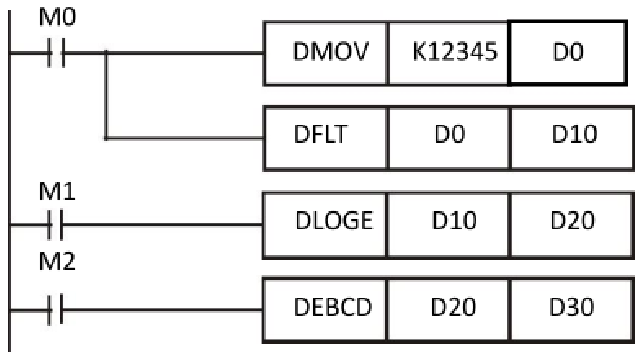
Natural logarithm [logarithm whose base is "e (2.71828)"] of **[S.+1, S.]** is calculated, and the operation result is stored to **[D.+1, D.]**. A real number can be directly specified as **S.**





- Only a positive value can be set in **[S.+1, S.]**. (The natural logarithm operation cannot be executed for a negative value.)
- An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067.
  1) When a negative value is specified in **S.** (error code: K6706)
  2) When "0" is specified in **S.** (error code: K6706)

Program example



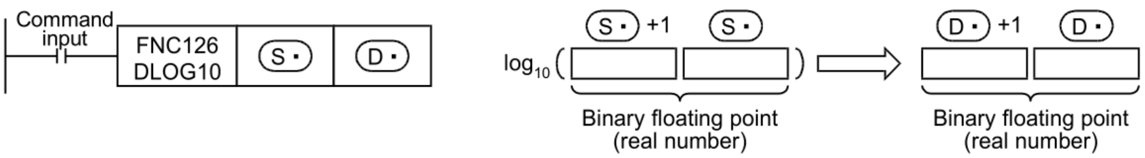
- When M0 is ON, convert the value of (D1, D0) into a binary floating-point number and store it in the (D11, D10) register.
- When M1 is ON, the (D11, D10) register is a true number for LOGE operation, and its value is a binary floating point value and stored in the (D21, D20) register.
- When M2 is ON, the binary floating-point value is converted into a decimal floating-point value and stored in the (D30, D31) register. (At this time D31 is the 10th power of D30)

16.14 LOG10/Floating Point Common Logarithm

This instruction executes the common logarithm operation.

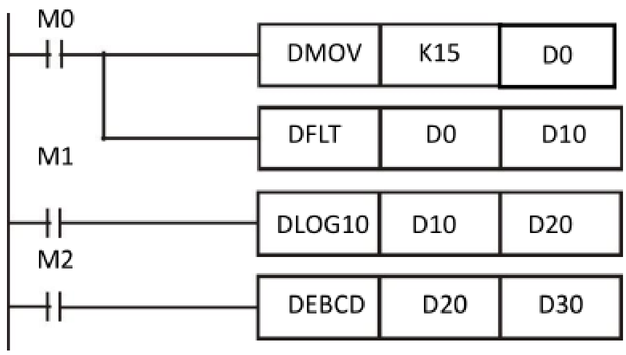
Instruction format	Operand Type	Function					
		16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
D **FNC 126 LOG10** P	**S.** **D.**		—		9 steps	DLOG10	Continuous Operation
						DLOG10P	Pulse Operation
Operand	**S.**	Head device number storing binary floating point data used in the common logarithm operation **Target Device**: D, R, E, retouch					Real number (binary)
	D.	Head device number storing the operation result **Target Device**: D, R, retouch					

Explanation of function and operation	32-bit operation(DLOG10、DLOG10P)
	32-bit operation(DLOG10、DLOG10P) Common logarithm [logarithm whose base is "10"] of **[S.+1, S.]** is calculated, and the operation result is stored to **[D.+1, D.]**. A real number can be directly specified as **S.**



	● Only a positive value can be set in **[S.+1, S.]**. (The common logarithm operation cannot be executed for a negative value.) ● An operation error occurs in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067. 1) When a negative value is specified in **S.** (error code: K6706) 2) When "0" is specified in **S.** (error code: K6706)

Program example



◆ When M0 is ON, convert the value of (D1, D0) into a binary floating-point number and store it in the (D11, D10) register.

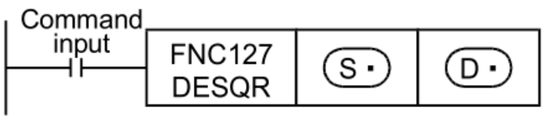
◆ When M1 is ON, find the common logarithm of "15" set in D0, and its value is a binary floating point value and stored in the (D21, D20) register.

◆ When M2 is ON, the binary floating-point value is converted into a decimal floating-point value and stored in the (D30, D31) register. (At this time D31 is the 10th power of D30)

16.15 ESQR/2 Floating Point Square Root

Instruction for computing the square root (open root) of a binary floating point number.

Instruction	Operand type	Functions					
		16-bit instruction	Mnemonic	Operation condition	32-bit instruction	Mnemonic	Operation condition
D FNC127 ESQR P	S. D.	—			9 steps	DESQR	Continuous operation
						DESQRP	Pulse operation
Operand	S.	The starting number of the device holding binary floating-point data that performs square root operation **Devices:** D, R, E, K, H, decoration					Real number (binary)
	D.	Data register number for storing binary floating-point data after square root operation **Device:** D, R, decoration					
Explanation of Instructions	**32-bit operation (DESQR, DESQRP)** **The square root of [S. +1, S.] is calculated (in the binary floating point operation), and the result is transferred to [D. +1, D.].**						



$\sqrt{[S\cdot+1, S\cdot]} \rightarrow [D\cdot+1, D\cdot]$

Binary floating point → Binary floating point

- M8020, zero flag. Turns ON when the operation result is true 0.
- The contents of [S. + 1, S.] are valid only when a positive value is set. When a negative value is set, the operation error flag M8067 turns ON, and the instruction is not executed.

Program Example 1

- When X0 is ON, take the square root of binary floating point numbers (D1, D0) and store the result in the register specified by (D11, D10).
- $\sqrt{(D1, D0)}$ →(D11, D10)

Program Example 2

- When X2 is ON, take the square root of K1,234 (automatically converted to binary floating point), and store the result in (D11, D10).

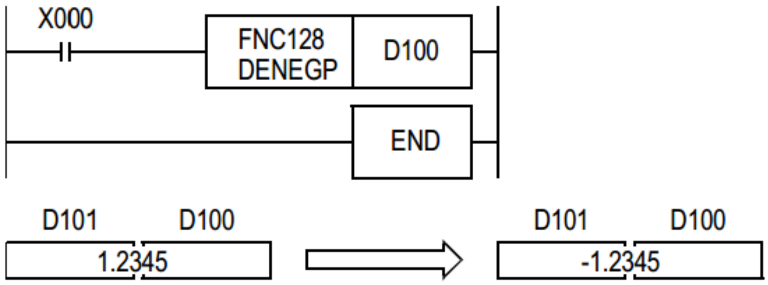
16.16 ENEG/2 Floating Point Negation

This instruction inverts the sign of binary floating point (real number) data.

<table>
<tr><th>Instruction</th><th>Operand type</th><th colspan="7">Functions</th></tr>
<tr><td rowspan="3">D FNC128 ENEG P</td><td rowspan="3">D.</td><td>16-bit instruction</td><td>Mnemonic</td><td>Operation condition</td><td></td><td>32-bit instruction</td><td>Mnemonic</td><td>Operation condition</td></tr>
<tr><td rowspan="2"></td><td rowspan="2">—</td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2">5 steps</td><td>DENEG</td><td>Continuous operation</td></tr>
<tr><td>DENEGP</td><td>Pulse operation</td></tr>
<tr><td>Operand</td><td>D.</td><td colspan="6">Head device number storing binary floating data whose sign is to be inverted.
Device: D, R, decoration</td><td>Real number (binary)</td></tr>
<tr><td>Explanation of Instructions</td><td colspan="8">32-bit operation (DENEG and DENEGP)
The sign of binary floating point stored in [D.+1, D.] is inverted, and the negation result is stored to [D.+1, D.].

</td></tr>
</table>

Program Example



◆ When X0 is ON, the sign of the binary floating-point number data of D100 and D101 is inverted and stored in D100 and D101.

16.17 INT/2 Hexadecimal Floating Point → BIN Integer Conversion

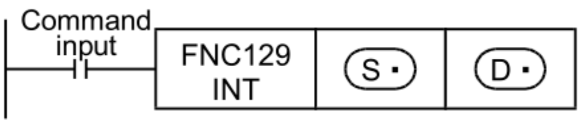
This instruction converts binary floating point data into a binary integer which is a normal data format inside PLCs (binary floating point → binary integer)

Instruction	Operand type	Functions					
D FNC129 INT P	S. D.	16-bit instruction	Mnemonic	Operation condition	32-bit instruction	Mnemonic	Operation condition
		5 steps	INT	Continuous operation	9 steps	DINT	Continuous operation
			INTP	Pulse operation		DINTP	Pulse operation
Operand	S.	Data register number storing binary floating point data to be converted into a binary integer. **Device:** D, R, decoration					Real number (binary)
	D.	Data register number storing a converted binary integer. **Device:** D, R, decoration					BIN 16- or 32-bit binary

Explanation of Instructions

1. 16-bit operation (INT and INTP)

Binary floating point stored in **[S.+1, S.]** is converted into a binary integer, and transferred to **D.**.



(S·)+1, (S·) → (D·)
Binary floating point
16-bit binary integer
The decimal part is cut.

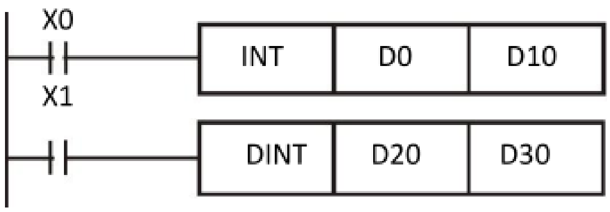
2. 32-bit operation (DINT and DINTP)

Binary floating point stored in **[S.+1, S.]** is converted into a binary integer, and transferred to **[D.+1, D.]**.

- Values after the decimal point are rounded.
- Related devices

Device	Name	Description
M8020	Zero flag	Turns ON when the operation result is 0
M8021	Borrow flag	Turns ON when the conversion result is cut in the decimal part.
M8022	Carry flag	Turns ON when the operation result is more than −32768 (in 16-bit operation) or −2,147,483,648 (in 32-bit operation). Turns OFF when it is out of the range of ON.

Program Example



- When X0 = ON, convert binary floating-point numbers (D1, D0) into BIN integers and store the result in (D10). BIN integer floating-point numbers are cut.
- When X1 = ON, convert binary floating point numbers (D21, D20) into BIN integers and store the result in (D31, D30), BIN integer floating point numbers are cut.

16.18 SIN/2Floating Point Sine

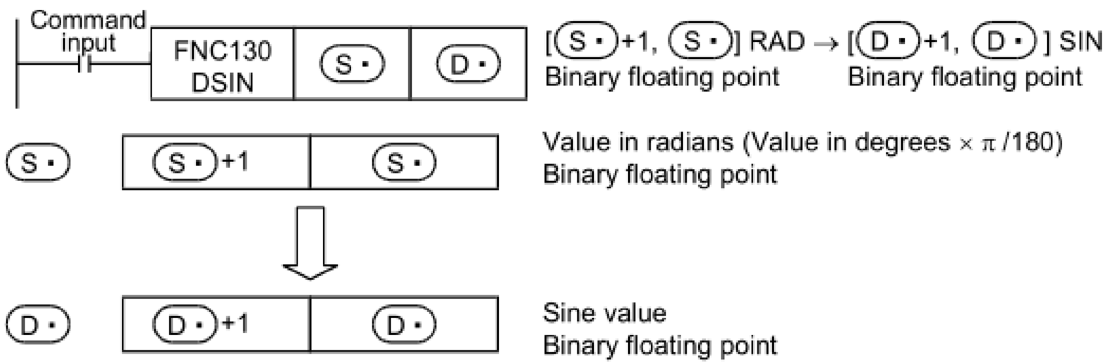
This instruction obtains the sine value of an angle (in radians).

<table>
<tr><th>Instruction</th><th>Operand type</th><th colspan="6">Functions</th></tr>
<tr><td rowspan="3">D
FNC130
SIN
P</td><td rowspan="3">**S.**
D.</td><td>16-bit instruction</td><td>Mnemonic</td><td>Operation condition</td><td>32-bit instruction</td><td>Mnemonic</td><td>Operation condition</td></tr>
<tr><td></td><td>—</td><td></td><td>9 steps</td><td>DSIN</td><td>Continuous operation</td></tr>
<tr><td></td><td>—</td><td></td><td></td><td>DSINP</td><td>Pulse operation</td></tr>
<tr><td rowspan="2">Operand</td><td>**S.**</td><td colspan="5">Device number storing an angle (in radians) in binary floating point
Device: D, R, E, decoration</td><td rowspan="2">Real number (binary)</td></tr>
<tr><td>**D.**</td><td colspan="5">Device number storing the sine value in binary floating point
Device: D, R, decoration</td></tr>
</table>

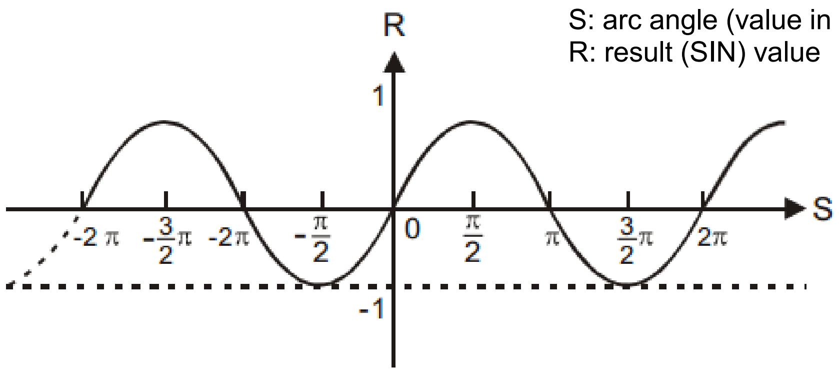
Explanation of Instructions

1. 32-bit operation (DSIN and DSINP)

A value of angle (binary floating point) specified in **[S.+1, S.]** is converted into the sine value, and transferred to **[D.+1, D.]**.



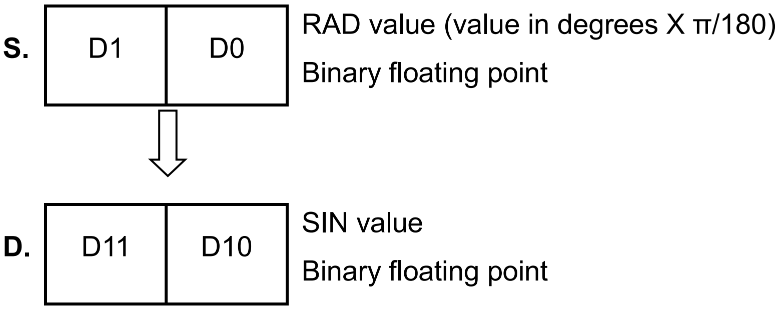
The relationship between arc angle and result:



S: arc angle (value in degrees) data
R: result (SIN) value

Program Example

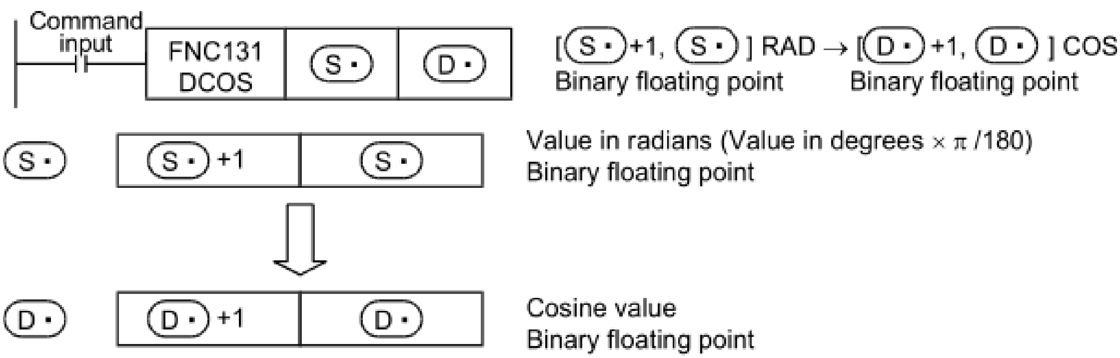
◆ When X0 is ON, the radian (RAD) value of the specified binary floating point number (D1, D0) is calculated and the SIN value is obtained and stored in (D11, D10). The content is binary floating point number.

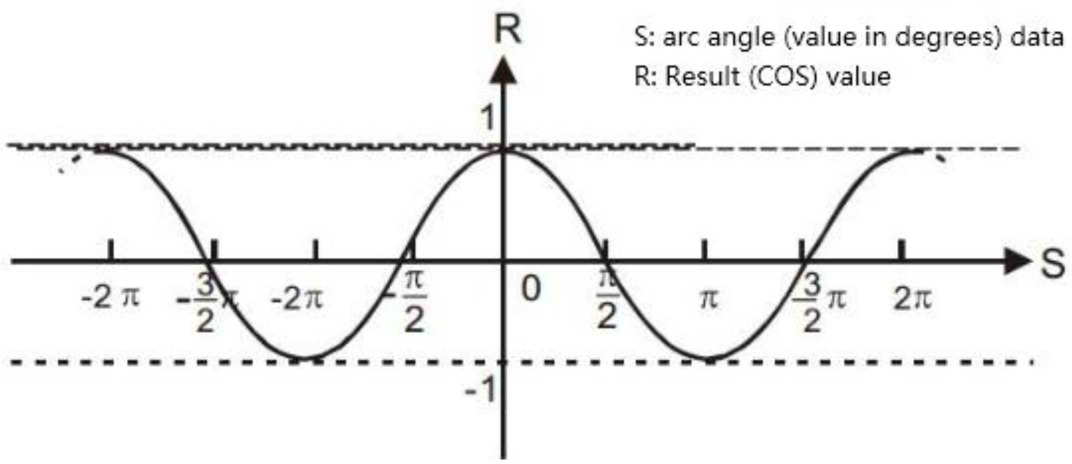


16.19 COS/ Binary Floating Point Cosine

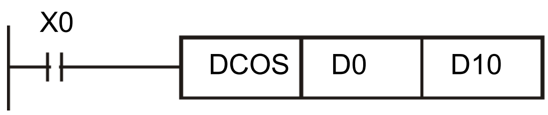
This instruction obtains the cosine value of an angle (in radians).

Instruction	Operand type	Functions					
		16-bit instruction	Mnemonic	Operation condition	32-bit instruction	Mnemonic	Operation condition
D FNC131 COS P	S. D.		—		9 steps	DCOS	Continuous operation
			—			DCOSP	Pulse operation
Operand	S.	Device number storing an angle (in radians) in binary floating point. **Device:** D, R, E, decoration					Real number (binary)
	D.	Device number storing the cosine value in binary floating point **Device:** D, R, decoration					

Explanation of Instructions	**32-bit operation (DCOS and DCOSP)** A value of angle (binary floating point) specified in **[S.+1, S.]** is converted into the cosine value, and transferred to **[D.+1, D.]**.  **The relationship between arc angle and result:**



Program Example



◆ When X0 is ON, the radian (RAD) value of the specified binary floating point number (D1, D0) is calculated and stored in (D11, D10). The content is a binary floating point number.

S.	D1	D0	RAD value (value in degrees X π/180) Binary floating point

⇩

D.	D11	D10	COS value Binary floating point

16.20 TAN/ Binary Floating Point Tangent

This instruction obtains the tangent value of an angle (in radians).

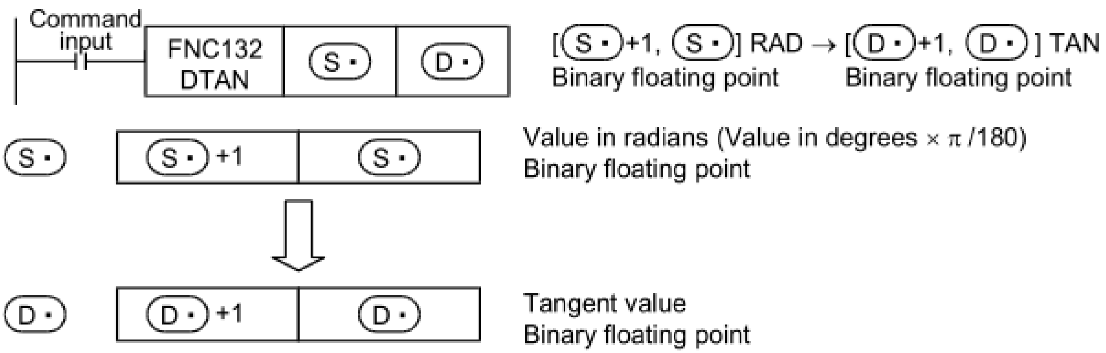
Instruction	Operand type	Functions						
D FNC132 TAN P	S. D.	16-bit instruction	Mnemonic	Operation condition		32-bit instruction	Mnemonic	Operation condition
			—			9 steps	DTAN	Continuous operation
			—				DTANP	Pulse operation

Operand			
Operand	S.	Device number storing an angle (in radians) in binary floating point **Device: D, R, E, decoration**	Real number (Binary)
	D.	Device number storing the tangent value in binary floating point **Device: D, R, decoration**	

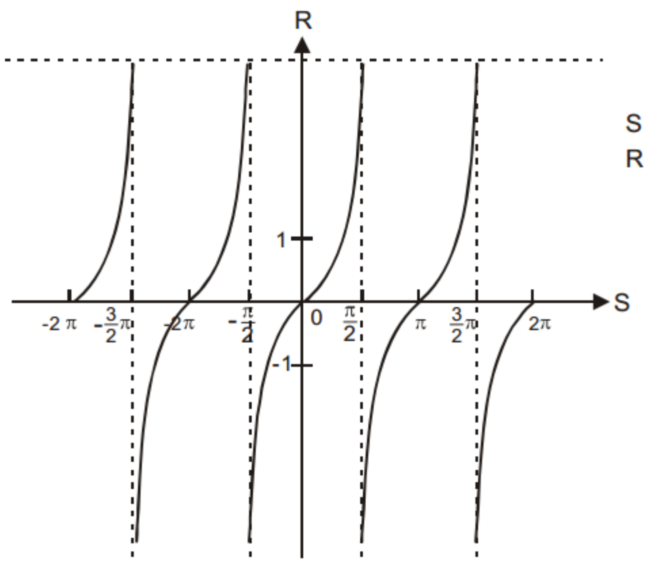
Explanation of Instructions

32-bit operation (DTAN and DTANP)

A value of angle (binary floating point) specified in **[S.+1, S.]** is converted into the tangent value, and transferred to **[D.+1, D.]**.



The relationship between arc angle and result:



S
R

S: arc angle (value in degrees) data
R: Result (TAN) value

16.21 ASIN/Binary Floating Point Arc Sine

This instruction executes the SIN −1 (arc sine) operation.

Instruction	Operand type	Functions					
		16-bit instruction	Mnemonic	Operation condition	32-bit instruction	Mnemonic	Operation condition
D FNC133 ASIN P	S. D.		—		9 steps	DASIN	Continuous operation
			—			DASINP	Pulse operation

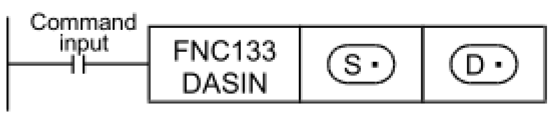
Operand			
Operand	S.	Head device number storing a sine value used in the SIN −1 (arc sine) operation. **Device: D, R, E, decoration**	Real number (Binary)
	D.	Head device number storing the operation result. **D, R, decoration**	

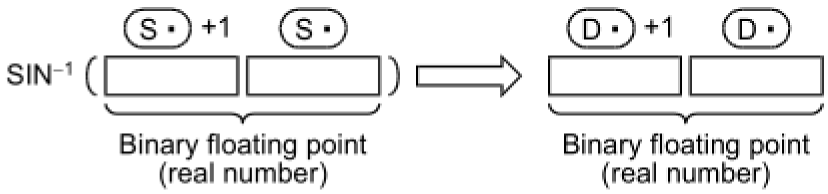
Explanation of Instructions

32-bit operation (DASIN and DASINP)

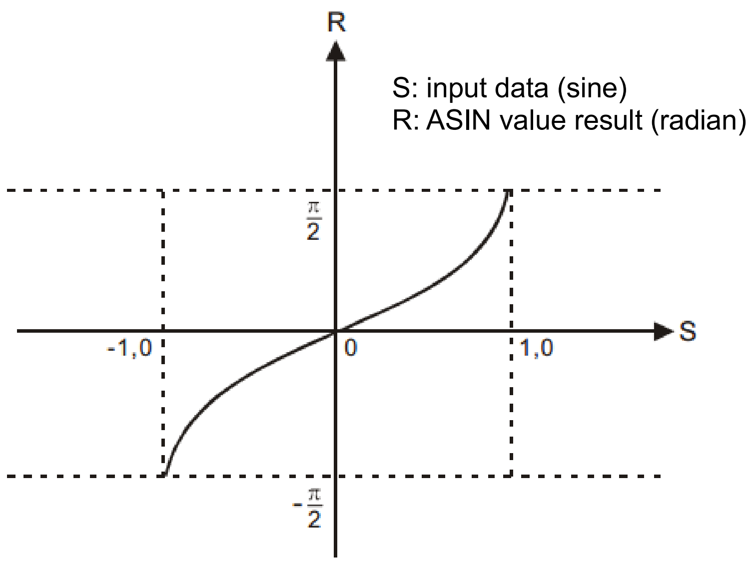
An angle is obtained from the sine value stored in **[S.+1, S.]**, and stored to **[D.+1, D.]**. ASIN value=SIN^{-1}

A real number can be directly specified as S..





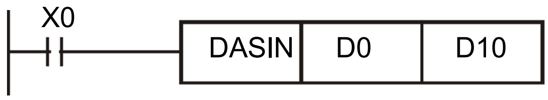
Relationship between input data and results:



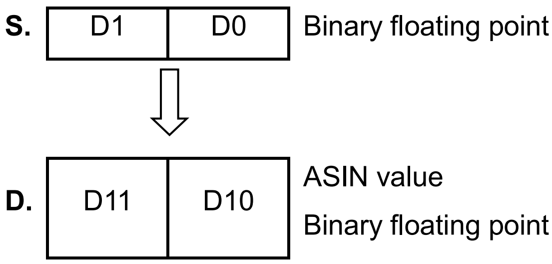
- The sine value stored in **[S.+1, S.]** can be set within the range from −1.0 to +1.0. An operation error

	is caused in the following case; The error flag M8067 turns ON, and the error code is stored in D8067. (error code: K6706) ● The angle (operation result) stored in **[D.+1, D.]** is expressed in radians (from −π/2 to π/2).

Program Example



◆ When X0 is ON, specify the binary floating point number (D1, D0) to find the ASIN value and store it in (D11, D10), the content is binary floating point number.

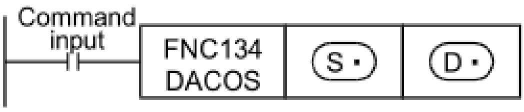


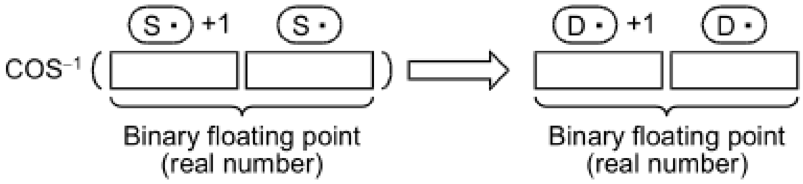
16.22 ACOS/Binary Floating Point Arc Cosine

This instruction executes the COS −1 (arc cosine) operation.

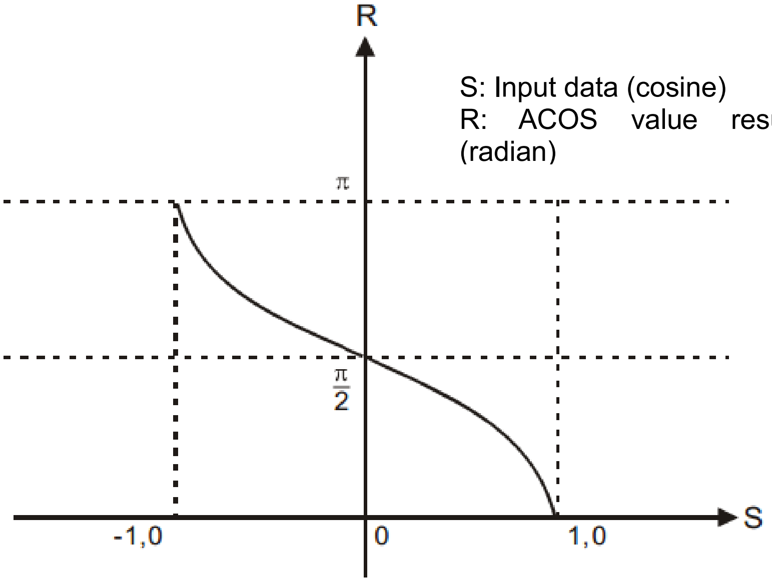
<table>
<tr><th>Instruction</th><th>Operand type</th><th colspan="7">Functions</th></tr>
<tr><td rowspan="3">FNC134
ACOS
D P</td><td rowspan="3">S.
D.</td><td>16-bit instruction</td><td>Mnemonic</td><td>Operation condition</td><td></td><td>32-bit instruction</td><td>Mnemonic</td><td>Operation condition</td></tr>
<tr><td></td><td>—</td><td></td><td></td><td>9 steps</td><td>DACOS</td><td>Continuous operation</td></tr>
<tr><td></td><td>—</td><td></td><td></td><td></td><td>DACOSP</td><td>Pulse operation</td></tr>
<tr><td rowspan="2">Operand</td><td>S.</td><td colspan="6">Head device number storing a cosine value used in the COS −1 (arc cosine)
operation.
Device: D, R, E, decoration</td><td rowspan="2">Real number (binary)</td></tr>
<tr><td>D.</td><td colspan="6">Head device number storing the operation result.
Device: Device: D, R, decoration</td></tr>
</table>

Explanation of Instructions	**32-bit operation (DACOS and DACOSP)** An angle is obtained from the cosine value stored in **[S.+1, S.]**, and stored to **[D.+1, D.]**. A real number can be directly specified as S..



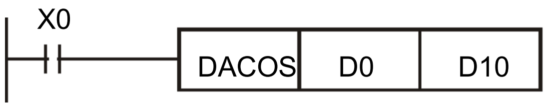


Relationship between input data and results:

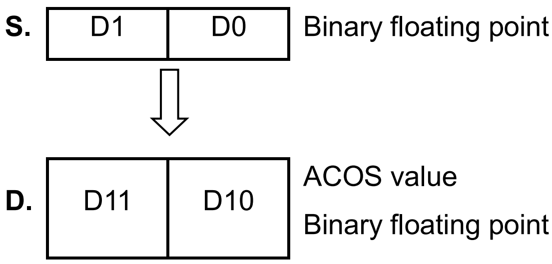


- The cosine value stored in **[S.+1, S.]** can be set within the range from −1.0 to +1.0. An operation error is caused in the following case; The error flag M8067 turns ON, and the error code is stored in D8067. (error code: K6706)
- The angle (operation result) stored in **[D.+1, D.]** is expressed in radians (from 0 to π).

Program Example



◆ When X0 is ON, specify the binary floating point number (D1, D0) to find the ACOS value and store it in (D11, D10), the content is binary floating point number.



16.23 ATAN/Binary Floating Point Arc Tangent

This instruction executes the TAN −1 (arc tangent) operation.

Instruction	Operand type	Functions					
		16-bit instruction	Mnemonic	Operation condition	32-bit instruction	Mnemonic	Operation condition
D **FNC135 ATAN** **P**	**S.** **D.**		—		9步	DATAN	Continuous operation
			—			DATANP	Pulse operation

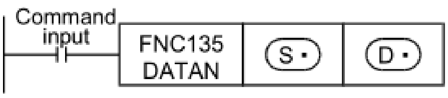
Operand	**S.**	Head device number storing a tangent value used in the TAN −1 (arc tangent) operation **Device: D, R, E, decoration**	Real number (Binary)
	D.	Head device number storing the operation result. **Device: D, R, decoration**	

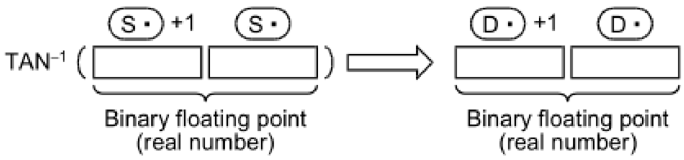
Explanation of Instructions

32-bit operation (DATAN and DATANP)

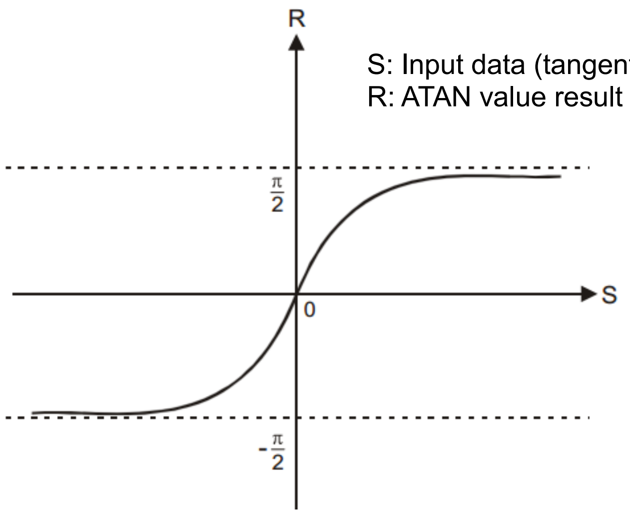
An angle is obtained from the tangent value stored in **[S.+1, S.]**, and stored to **[D.+1,D.]**.

A real number can be directly specified as S..



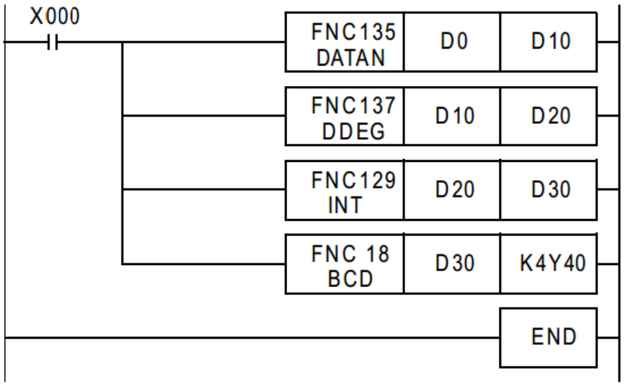


Relationship between input data and results:



- The angle (operation result) stored in [**D.+1**, **D.**] is expressed in radians (from −π/2 to +π/2).

Program example



- In the program example shown above, the TAN −1 value of data (binary floating point) stored in D0 and D1 is calculated, and the angle is output in 4-digit BCD to Y040 to Y057 when X000 turns ON.。

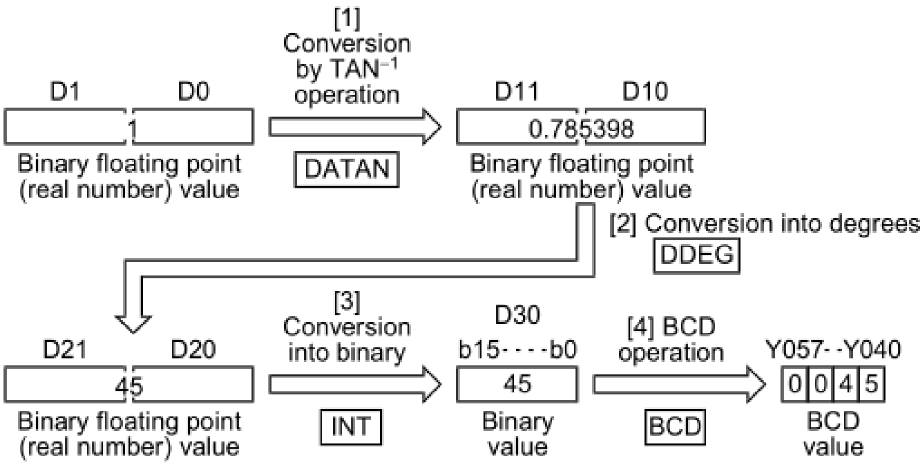
1) The angle (in radians) is calculated by the TAN^{-1} operation (①)

2) The value in radians is converted into the value in degrees (②)

3) The angle expressed in binary floating point (real number) is converted into an integer (binary) (③)

4) The angle expressed in integer (binary) is output to the display unit (④)

Operation when "1" is stored in D0 and D1



16.24 RAD/Binary Floating Point Degrees to Radians Conversion

This instruction converts a value in degrees into a value in radians.

Instruction	Operand type	Functions						
		16-bit instruction	Mnemonic	Operation condition		32-bit instruction	Mnemonic	Operation condition
D **FNC136 RAD** P	**S.** **D.**		—			9 steps	DRAD	Continuous operation
			—				DRADP	Pulse operation

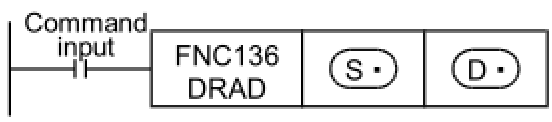
Operand			
Operand	**S.**	Head device number storing a value in degrees to be converted into a value in radians. **Device: D, R, E, decoration**	Real number (Binary)
	D.	Head device number storing a value in radians acquired by conversion. **Device: D, R, decoration**	

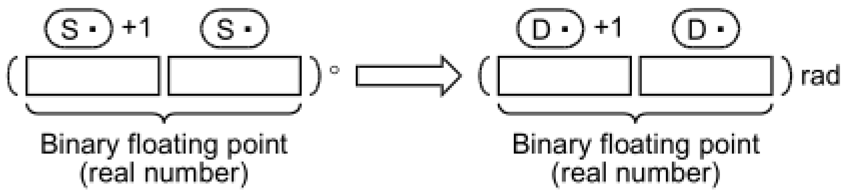
Explanation of Instructions

32-bit operation (DRAD and DRADP)

The unit of **[S.+1, S.]** is converted from degrees into radians, and the operation result is stored to **[D.+1, D.]**.

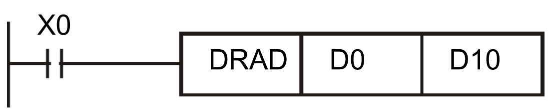
A real number can be directly specified as S..



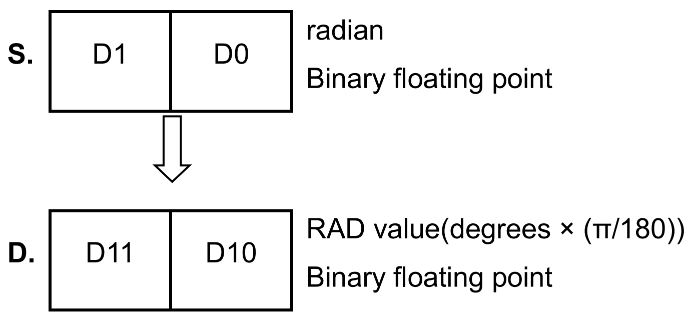


- Radians = Degrees × (π/180)

Program example



- When X0 is ON, specify the angle value of binary floating point number (D1, D0), convert the angle to radian value and store it in (D11, D10), the content is binary floating point number.



16.25 DEG/Binary Floating Point Radians to Degrees Conversion

This instruction converts a value in radians into a value in degrees.

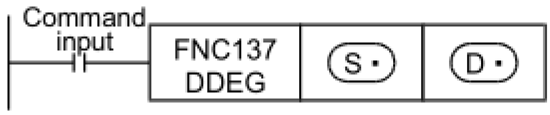
Instruction	Operand type	Functions					
		16-bit instruction	Mnemonic	Operation condition	32-bit instruction	Mnemonic	Operation condition
FNC137 DEG (D, P)	**S.** **D.**		—		9 steps	DDEG	Continuous operation
			—			DDEGP	Pulse operation
Operand	**S.**	Head device number storing a value in radians to be converted into a value in degrees. **Device: D, R, E, decoration**				Real number (Binary)	
	D.	Head device number storing a value in degrees acquired by conversion. **Device: D, R, decoration**					

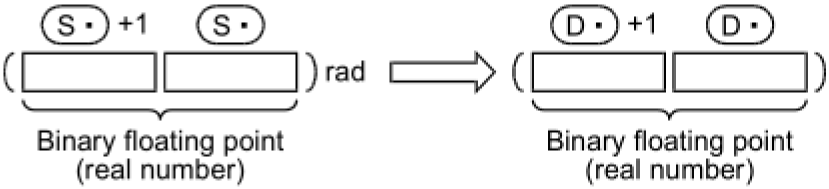
Explanation of Instructions

32-bit operation (DDEG and DDEGP)

The unit of **[S.+1, S.]** is converted from radians into degrees, and the operation result is stored to **[D.+1, D.]**.

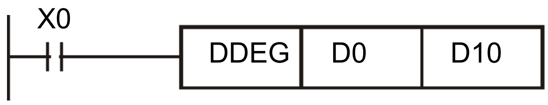
A real number can be directly specified as S..



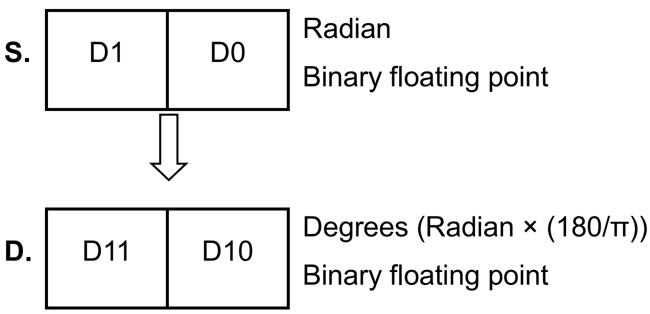


- Degrees = Radians × (180/π)

Program example



◆ When X0 is ON, specify the radian value of the binary floating point number (D1, D0), convert the radian value to the radian value, and store it in (D11, D10). The content is a binary floating point number.



17 Data Processing 2

FNC NO.	Instruction	Functions	Devices		
			3G 系列 PLC	2N 系列 PLC	MX2N 系列 PLC
140	WSUM	Sum of Word Data	★		
141	WTOB	WORD to BYTE	★		
142	BTOW	BYTE to WORD	★		
143	UNI	4-bit Linking of Word Data	★		
144	DIS	4-bit Grouping of Word Data	★		
145	—				
146	—				
147	SWAP	Byte Swap	★	★	★
148	—				
149	SORT2	Sort Tabulated Data 2	★		

17.1 WSUM/ Sum of Word Data

This instruction calculates the sum of consecutive 16-bit or 32-bit data.

When calculating the addition data (sum value) in units of byte (8 bits), use the CCD (FNC 84) instruction.

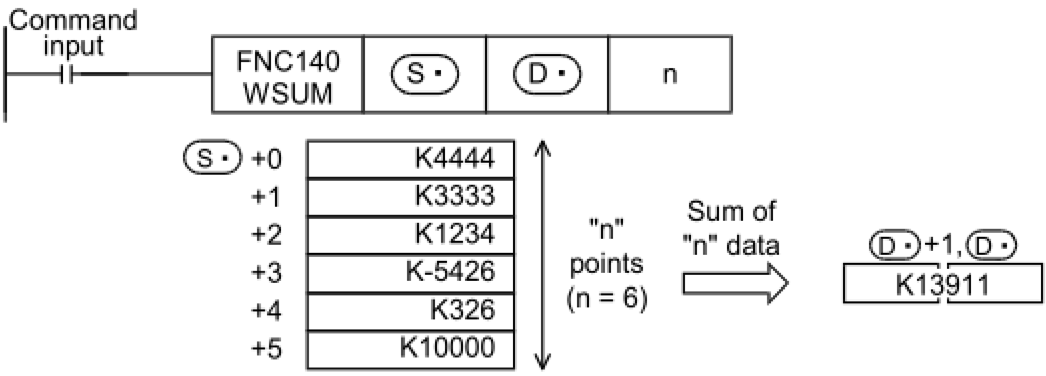
Instruction	Operand type	Functions						
D FNC140 WSUM P	S. D. n	16-bit instruction	Mnemonic	Operation condition		32-bit instruction	Mnemonic	Operation condition
		7 steps	WSUM	Continuous operation		13 steps	DWSUM	Continuous operation
			WSUMP	Pulse operation			DWSUMP	Pulse operation

Operand			
Operand	S.	Head device number storing data whose sum is calculated. **Device: T, C, D, R, decoration**	BIN16/ 32 bit
	D.	Head device number storing sum **Device: T, C, D, R, decoration**	BIN32/ 64 bit
	n	Number of data (0 < n) **Device: D, R, K, H**	BIN16/ 32 bit

Explanation of Instructions

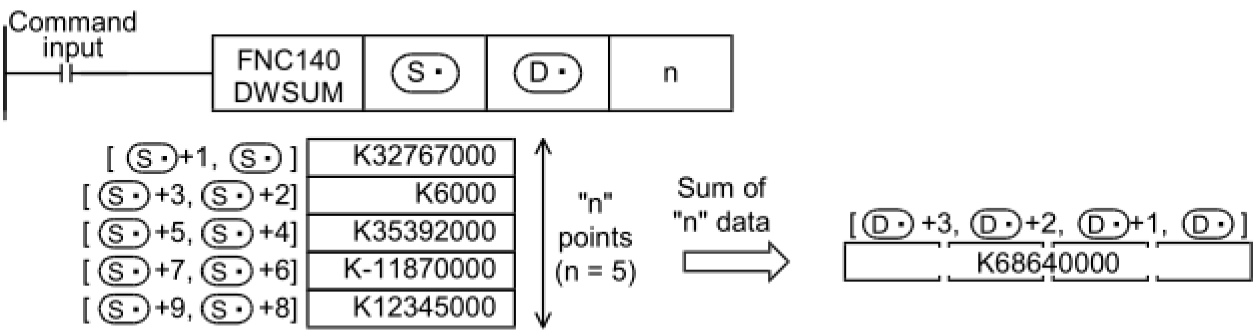
1. 16-bit operation (WSUM and WSUMP)

The sum of "n" 16-bit data starting from **S.** is stored as 32-bit data in **[D.+1, D.]**.



2. 32-bit operation (DWSUM and DWSUMP)

The sum of "n" 32-bit data starting from [**S.+1, S.]** is stored as 64-bit data in **[D.+3, D.+2, D.+1, D.]**.



- In the 32-bit operation, the acquired sum is 64-bit data. FX3U and FX3UC PLCs cannot handle 64-bit data. When the sum is within the numeric range of 32-bit data (K-2,147,483,648 to

	K2,147,483,647), however, FX3U and FX3UC PLCs can handle the low-order 32 bits of 32-bit data as the sum while ignoring the high-order 32 bits. ● When D and R are designated as n of 32-bit instruction, the 32-bit value of [n+1, n] will take effect, so please be careful. For example, when DWSUM D0 D100 R0, then n=[R1, R0] ● An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067. • When "n" points starting from **S.** are outside the specified device range (error code: K6706) • When "n" is smaller than or equivalent to "0" (error code: K6706) • When **D.** are outside the specified device range. (error code: K6706)

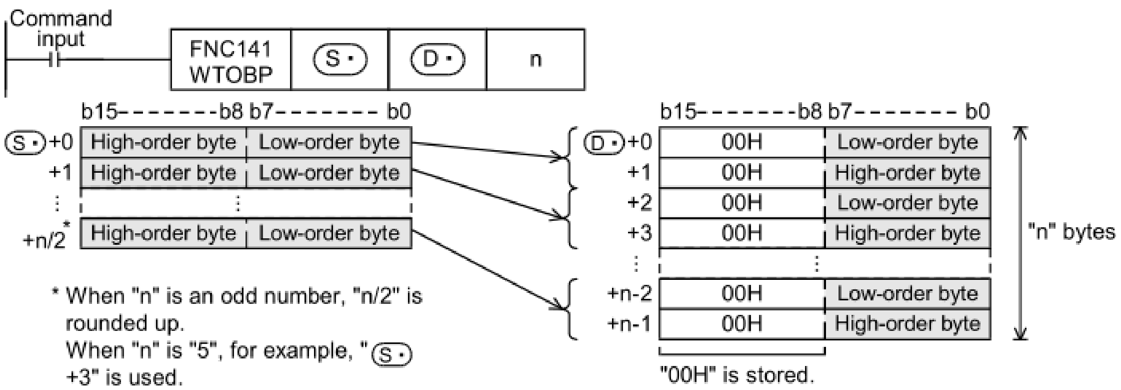
17.2 WTOB/ WORD to BYTE

This instruction separates consecutive 16-bit data in byte units (8 bits).

Instruction	Operand type	Functions					
		16-bit instruction	Mnemonic	Operation condition	32-bit instruction	Mnemonic	Operation condition
FNC141 WTOB	**S.** **D.** **n**	7 steps	WTOB	Continuous operation		—	
P			WTOBP	Pulse operation		—	

Operand	**S.**	Head device number storing data to be separated in byte units **Device: T, C, D, R, decoration**	BIN16/ 32 bit
	D.	Head device number storing result of separation in byte units **Device: T, C, D, R, decoration**	BIN32/ 64 bit
	n	Number of byte data to be separated (0≦n) **Device: D, R, K, H**	BIN16/ 32 bit

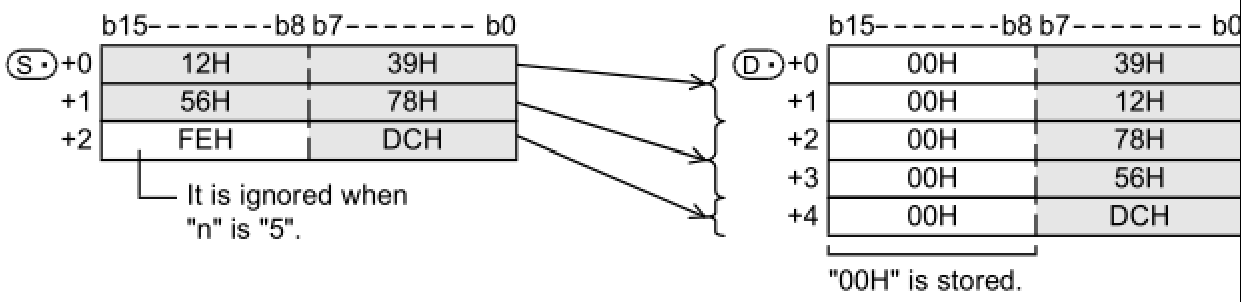
Explanation of Instructions	**1. 16-bit operation (WTOB and WTOBP)** 1) "n/2" 16-bit data stored in **S.** and later is separated into "n" bytes, and stored in "n" devices starting from **D.** as shown below.



2) "00H" is stored in the high-order byte (8 bits) of each device (and later) storing the separated byte data.

3) When "n" is an odd number, only the low-order byte (8 bits) of the final separation source device is regarded as the target data as shown in the figure below.

For example, when "n" is "5", the data from **S.** to the low-order byte (8 bits) of **(S.+2)** is stored in **D.** to **(D.+4).**



4) When "n" is "0", WTOB instruction is not executed.

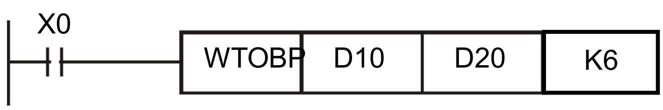
- The device that saves the separated source data and the device that saves the separated data can be reused.
- An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067.

  • When the separation source devices **S.** to **(S. +n/2)** are outside the specified device range (error code: K6706)

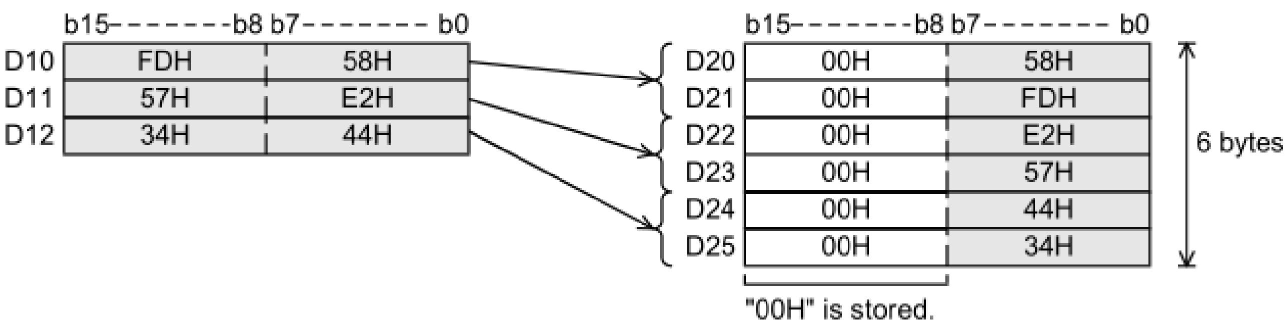
  When "n" is an odd number, the number of a rounded up value decides the number of devices. (error code: K6706)

  • When the separated data destination devices **D.** to **(D.+n-1)** are outside the specified device range (error code: K6706)

Program example



◆ When X0 is ON, the data stored in D10 to D12 is separated in byte units, and stored in D20 to D25.



17.3 BTOW/ BYTE to WORD

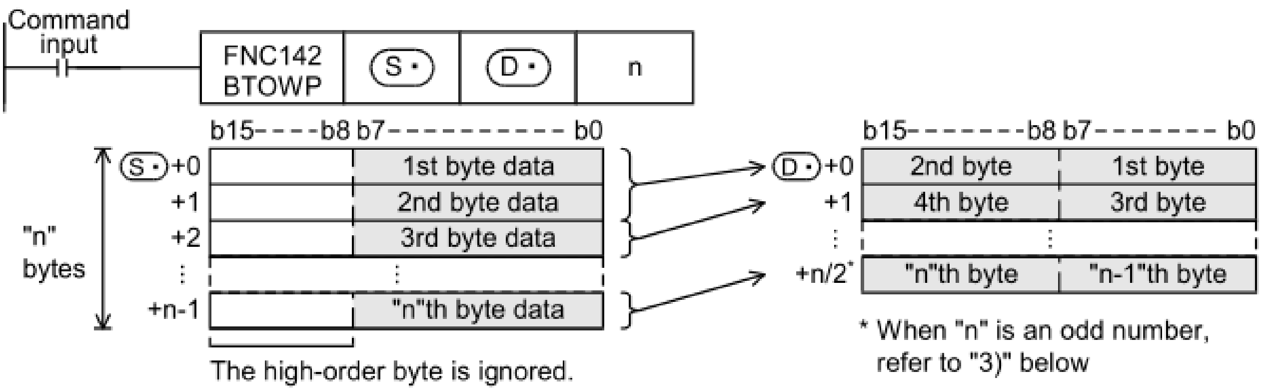
This instruction combines the low-order 8 bits (low-order byte) of consecutive 16-bit data.

Instruction	Operand type	Functions					
		16-bit instruction	Mnemonic	Operation condition	32-bit instruction	Mnemonic	Operation condition
FNC142 BTOW / P	**S.** **D.** **n**	7 steps	BTOW	Continuous operation		—	
			BTOWP	Pulse operation		—	
Operand	**S.**	Head device number storing data to be combined in byte units **Device: T, C, D, R, decoration**				BIN16 bit	
	D.	Head device number storing data acquired by combination in byte units **Device: T, C, D, R, decoration**				BIN16 bit	
	n	Number of byte data to be combined(0≦n) **Device: D, R, K, H**				BIN16 bit	

Explanation of Instructions

16-bit operation (BTOW and BTOWP)

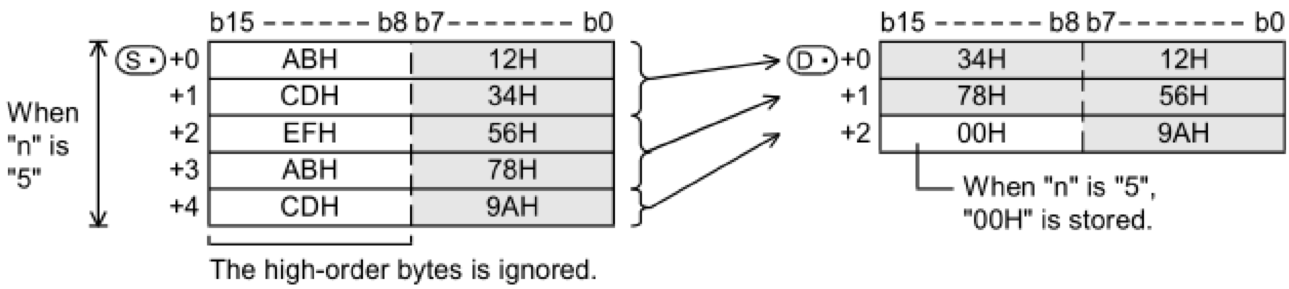
1) The low-order byte (8 bits) of "n" 16-bit data starting from **S.** is combined, and stored in "n/2" devices starting from **D.** as shown below.



2) The high-order byte (8 bits) of each combination source 16-bit data (and later) is ignored.

3) When "n" is an odd number, "00H" is stored in the high-order byte (8 bits) of the final one among the combination result destination devices as shown below.

For example, when "n" is "5", the low-order byte (8 bits) of **S.** to **(S.+4)** is stored in **D.** to **(D.+2)**, and "00H" is stored in the high-order byte (8 bits) of **D.+2**.



4) When "n" is "0", the BTOW instruction is not executed.

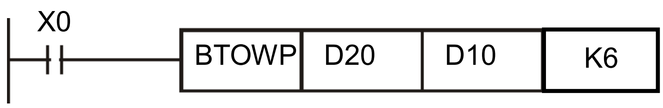
- The device that saves the combined source data and the device that saves the combined data can be reused.
- An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067.

  • When the combination source devices **S.** to **(S. +n-1)** are outside the specified device range (error code: K6706)

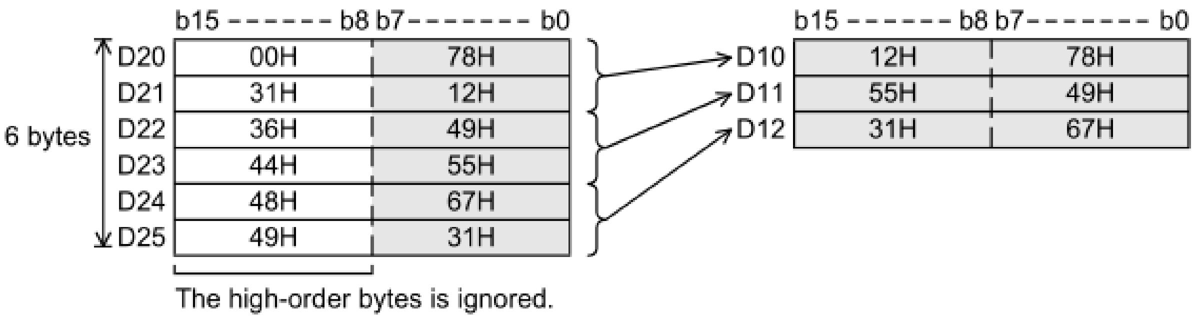
  • When the combined data destination devices **D.** to **(D.+n/2)** are outside the specified device range (error code: K6706)

  When "n" is an odd number, the number of a rounded up value decides the numer of devices. (error code: K6706)

Program example



◆ In the program shown below, the low-order byte (8 bits) data stored in D20 to D25 is combined, and stored in D10 to D12.



17.4 UNI/ 4-bit Linking of Word Data

This instruction combines the low-order 4 bits of consecutive 16-bit data.

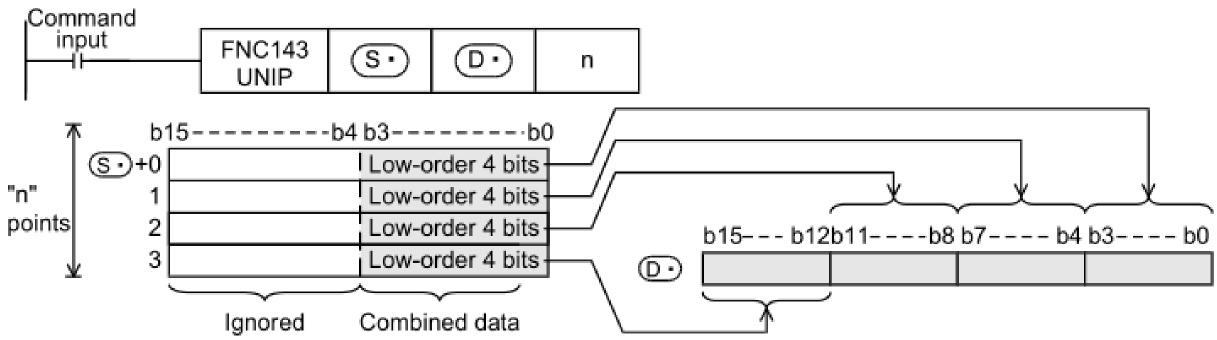
Instruction	Operand type	Functions					
FNC143 UNI	**S.** **D.** **n**	16-bit instruction	Mnemonic	Operation condition	32-bit instruction	Mnemonic	Operation condition
		7 steps	UNI	Continuous operation		—	
P			UNIP	Pulse operation		—	

Operand			
Operand	**S.**	Head device number storing data to be combined **Device: T, C, D, R, decoration**	BIN16 bit
	D.	Device number storing combined data **Device: T, C, D, R, decoration**	BIN16 bit
	n	Number of data to be combined (0 to 4, When "n" is "0", UNI instruction is not executed.) **Device: D, R, K, H**	BIN16 bit

Explanation of Instructions

16-bit operation (UNI/UNIP)

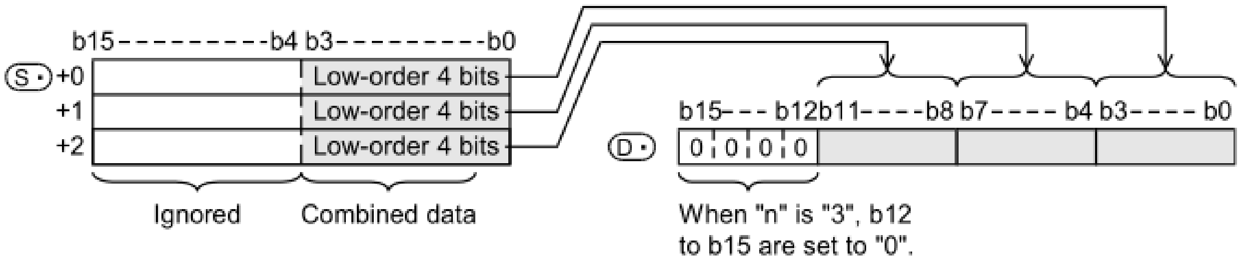
1) The low-order 4 bits of "n" 16-bit data starting from **S.** are combined, and stored in **D.** as shown



2) Specify a number 1 to 4 in "n". In the case of "n = 0", UNI instruction is not executed.

3) In the case of “1≦n≦3”, the high-order {4×(4-n)} bits of **D.** are set to ”0”.

For example, when "n" is "3", the low-order 4 bits of **S.** to **(S.+2)** are stored in b0 to b11 of **D.**, and the high-order 4 bits of **D.** are set to "0".



- An operation error occurs in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067.

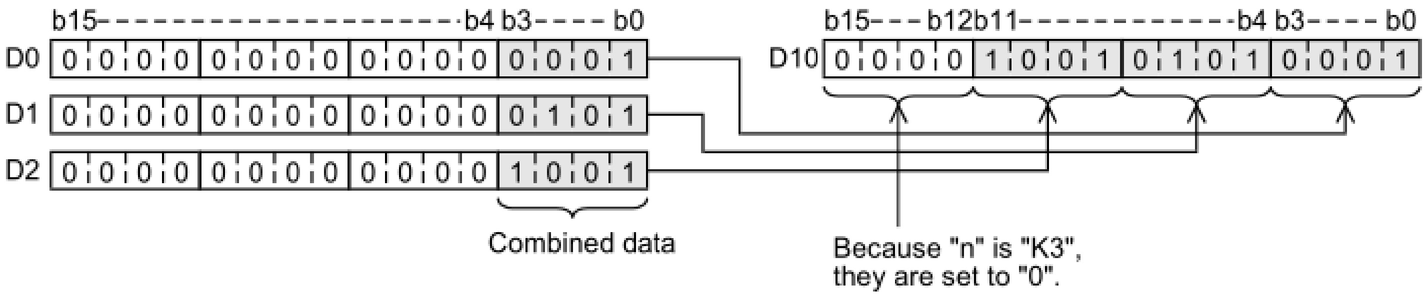
  • When **S.** to **(S. +n)** are outside the specified device range (error code: K6706)

	• When "n" is outside the range from "0 to 4" (error code: K6706)

Program example

X0	UNIP	D0	D10	K3

◆ In the program below, the low-order 4 bits of D0 to D2 are combined and stored in D10 when X0 turns ON.

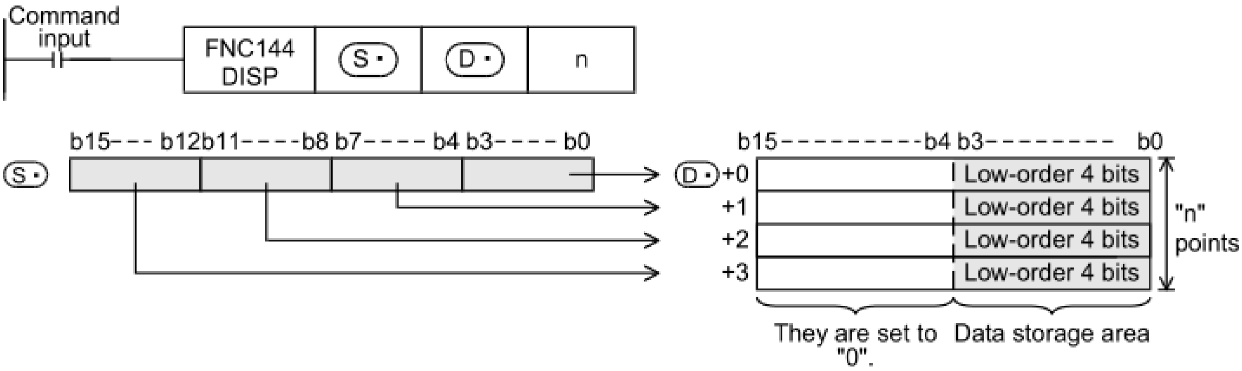


17.5 DIS/ 4-bit Grouping of Word Data

This instruction separates 16-bit data into 4 bit units.

Instruction	Operand type	Functions					
FNC144 DIS P	S. D. n	16-bit instruction	Mnemonic	Operation condition	32-bit instruction	Mnemonic	Operation condition
		7 steps	DIS	Continuous operation		—	
			DISP	Pulse operation		—	
Operand	S.	Device number storing data to be separated **Device: T, C, D, R, decoration**			BIN16 bit		
	D.	Head device number storing separated data **Device: T, C, D, R, decoration**			BIN16 bit		
	n	Number of data to be separated (0 to 4) (When "n" is "0", DIS instruction is not executed.) **Device: D, R, K, H**			BIN16 bit		

Explanation of Instructions	**16-bit operation (DIS and DISP)** 1) 16-bit data stored in **S.** is separated in 4-bit units, and stored in **D.** as shown below.



2) Specify a number 1 to 4 in "n". In the case of "n = 0", DIS instruction is not executed.

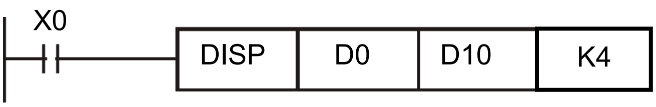
3) High-order 12 bits of "n" devices starting from **D.** are set to "0".

- An operation error occurs in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067.

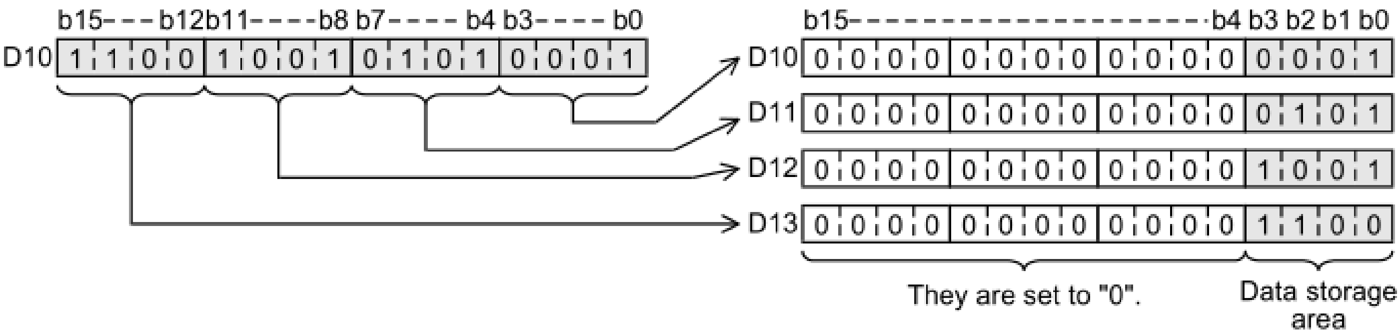
  • When **D.** to (**D.**+n) are outside the specified device range (error code: K6706)

  • When "n" is outside the range from "0 to 4" (error code: K6706)

Program example



◆ In the program below, D0 is separated into 4 bit units and stored in D10 to D13 when X0 turns ON.



17.6 SWAP/ Byte Swap

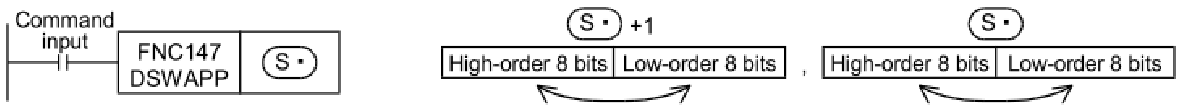
This instruction swaps the high-order 8 bits and low-order 8 bits of a word device.

Instruction	Operand type	Functions					
D FNC147 SWAP P	S.	16-bit instruction	Mnemonic	Operation condition	32-bit instruction	Mnemonic	Operation condition
		3 steps	SWAP	Continuous operation	5 steps	DSWAP	Continuous operation
			SWAPP	Pulse operation		DSWAPP	Pulse operation
Operand	S.	**Word device of high and low byte swap** **Device: KnY, KnM, KnS, T, C, D, R, V, Z, decoration**				BIN16/32 bit	

Explanation of Instructions	
	1. 16-bit operation (SWAP and SWAPP) High-order 8 bits and low-order 8 bits are swapped for each other.

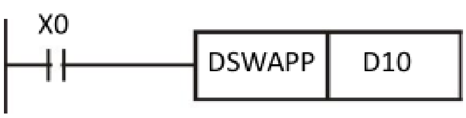
2. 32-bit operation (DSWAP and DSWAPP)

High-order 8 bits and low-order 8 bits are swapped for each other in each word device.

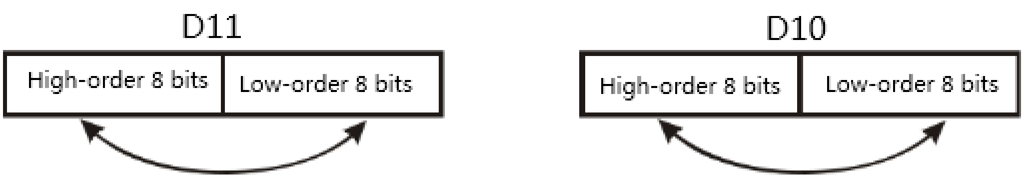


- When the continuous operation type instruction is used, swapping is executed in each operation cycle. This instruction works in the same way as the extension function of the XCH (FNC 17) instruction.

Program example



◆ When X0 is ON, the high-order 8 bits and low-order 8 bits of D11 are exchanged, and the high-order 8 bits and low-order 8 bits of D10 are exchanged.



17.7 SORT2/ Sort Tabulated Data 2

This instruction sorts a data table consisting of data (lines) and group data (columns) based on a specified group data (column) sorted by line in either ascending or descending order. This instruction stores the data (lines) in serial devices facilitating the addition of data (lines).

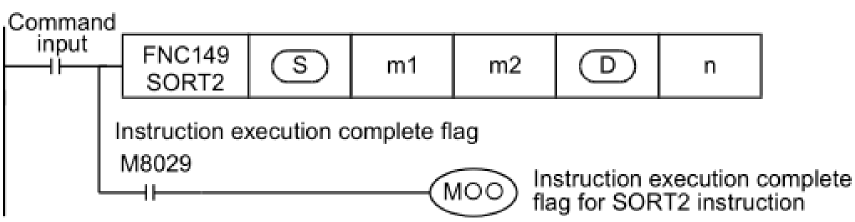
On the other hand, the SORT (FNC 69) instruction stores the group data (columns) in serial devices, and sorts a table in ascending order only.

<table>
<tr><th>Instruction</th><th>Operand type</th><th colspan="7">Functions</th></tr>
<tr><td rowspan="2">D
FNC149 SORT2
P</td><td rowspan="2">**S.
m1
m2
D.
n**</td><td>16-bit instruction</td><td>Mnemonic</td><td>Operation condition</td><td rowspan="2"></td><td>32-bit instruction</td><td>Mnemonic</td><td>Operation condition</td></tr>
<tr><td>11 steps</td><td>SORT2</td><td>Continuous operation</td><td>21 steps</td><td>DSORT2</td><td>Continuous operation</td></tr>
<tr><td rowspan="5">Operand</td><td>**S.**</td><td colspan="6">Head device number storing the data table [which occupies m1 × m2 points]
Device: D, R</td><td rowspan="5">BIN16/32 bit</td></tr>
<tr><td>**m1**</td><td colspan="6">Number of data (lines) [1 to 32]
Device: D, R, K, H</td></tr>
<tr><td>**m2**</td><td colspan="6">Number of group data (columns) [1 to 6]
Device: K, H</td></tr>
<tr><td>**D.**</td><td colspan="6">Head device number storing the operation result [which occupies m1 × m2 points]
Device: D, R</td></tr>
<tr><td>**n**</td><td colspan="6">Column number of group data (column) used as the basis of sorting [1 to m2]
Device: D, R, K, H</td></tr>
</table>

<table>
<tr><td>**Explanation of Instructions**</td><td>

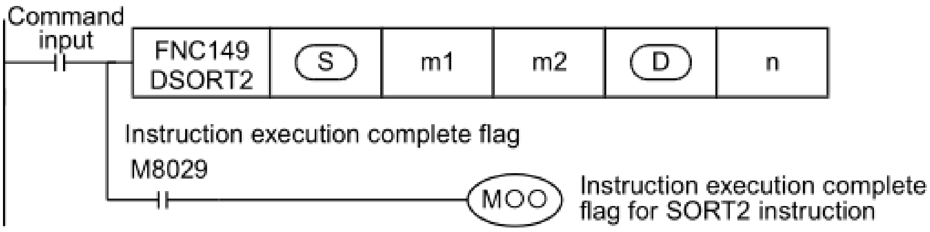
1. 16-bit operation (SORT2)

In the data table (sorting source) having (m1 x m2) points from **S.**, data lines are sorted in the ascending or descending order based on the group data in column No. "n", and the result is stored in the data table (occupying m1 x m2 points) from **D.**.



2. 32-bit operation (DSORT2)

In the data table (sorting source) having (m1 × m2) points from **[S.+1, S.]**, data lines are sorted in the ascending or descending order based on the group data in the column No. "n", and the result is stored in the data table (sorting result) having (m1 × m2) points from **[D.+1, D.].**
</td></tr>
</table>

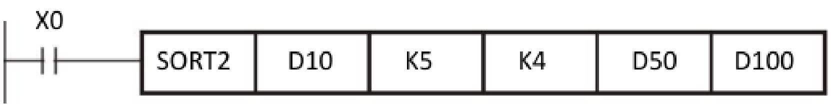


- Set the sorting order by setting M8165 to ON or OFF.
  1) M8165=ON, descending order
  2) M8165=OFF, ascending order
- When the command input turns ON, data sorting is started. Data sorting is completed after "m1" scans, and the instruction execution complete flag M8029 is set to ON.

Cautions:

- **Do not change the contents of operands and data during operation.**
- **To execute SORT2 instruction again, set the command input to OFF once, then ON again.**
- **Limitation in number of SORT2 instructions**
  Up to two SORT2 instructions can be simultaneously driven in a program.
- **Writing during RUN is disabled for a circuit block including SORT2 instruction.**
- **When the same device is specified in S. and D.**
  The source data is overwritten with the data acquired by sorting.
  Pay close attention not to change the contents of until execution of SORT2 instruction is completed.

Program example



◆ When X0=ON, specify to execute the data sorting. When sorting is completed, M8029=ON. Please do not change the contents of the sorted data during the execution of Instruction. If you want to reorder the data, please turn OFF and ON again.

Sorting source data

Line No. \ Column No.		Number of groups (m2 = K4)			
		1	2	3	4
		Control number	Height	Weight	Age
Number of data (m1 = 5)	1	(D10)1	(D11)150	(D12)45	(D13)20
	2	(D14)2	(D15)180	(D16)50	(D17)40
	3	(D18)3	(D19)160	(D20)70	(D21)30
	4	(D22)4	(D23)100	(D24)20	(D25)8
	5	(D26)5	(D27)150	(D28)50	(D29)45

1) Sorting result when the instruction is executed with "n = K2 (column No. 2)" (in the case of ascending order)

Line No. \ Column No.		Number of groups (m2 = K4)			
		1	2	3	4
		Control number	Height	Weight	Age
Number of data (m1 = 5)	1	(D22)4	(D23)100	(D24)20	(D25)8
	2	(D10)1	(D11)150	(D12)45	(D13)20
	3	(D26)5	(D27)150	(D28)50	(D29)45
	4	(D18)3	(D19)160	(D20)70	(D21)30
	5	(D14)2	(D15)180	(D16)50	(D17)40

2) Sorting result when the instruction is executed with "n = K3 (column No. 3)" (in the case of descending order)

Line No. \ Column No.		Number of groups (m2 = K4)			
		1	2	3	4
		Control number	Height	Weight	Age
Number of data (m1 = 5)	1	(D18)3	(D19)160	(D20)70	(D21)30
	2	(D14)2	(D15)180	(D16)50	(D17)40
	3	(D26)5	(D27)150	(D28)50	(D29)45
	4	(D10)1	(D11)150	(D12)45	(D13)20
	5	(D22)4	(D23)100	(D24)20	(D25)8

18 Positioning Control

FNC NO.	Instruction	Function	Device		
			3G PLC	2N PLC	MX2N PLC
150	DSZR	DOG Search Zero Return	★		
151	DVIT	Interrupt Positioning	★		
152	TBL	Batch Data Positioning Mode	★		
153	—				
154	—				
155	ABS	Absolute Current Value Read	★	★	★
156	ZRN	Zero Return	★	★	★
157	PLSV	Variable Speed Pulse Output	★	★	★
158	DRVI	Drive to Increment	★	★	★
159	DRVA	Drive to Absolute	★	★	★

18.1 DSZR/ Dog Search Zero Return

This instruction executes a zero return, and aligns the mechanical position with a present value register inside the PLC.

In addition, this instruction enables the following functions not supported by the ZRN (FNC156) instruction:

• DOG search function

• Zero return by the near-point (dog) signal and zero-phase signal

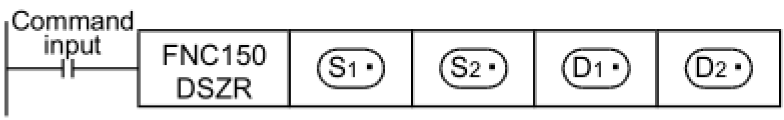
It is not possible, however, to count the zero-phase signal and then determine the zero point.

Instruction	Operand type	Functions						
FNC150 DSZR	**S1. S2. D1. D2.**	16-bit instruction	Mnemonic	Operation condition		32-bit instruction	Mnemonic	Operation condition
		9 steps	DSZR	Continuous operation			— —	

Operand	**S1.**	Device number for near-point signal (dog) **Device:** X, Y, M, S, D □.b, decoration	Bit
	S2.	Input number for zero-phase signal **Device:** X0~X7, decoration	
	D1.	Device number (Y) from which pulses are to be output **Device:** transistor outputs, 3G PLC: Y0~Y3, decoration	
	D2.	Device number to which rotation direction signal is output **Device:** Y(designated pulse direction), decoration	

Explanation of Instructions

16-bit operation (DSZR)



● During RUN, avoid writing while the DSZR (FNC150) instruction is executed (that is, while a pulse is output).

Note that if writing is executed during RUN to a circuit block including the FNC150 instruction while pulses are output, the PLC decelerates and stops pulse output.

18.1.1 Related devices

● 3G PLC special auxiliary relay:

Functions \ Pulse point	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
Send end flag	M8029							
Instruction execution abnormal end flag	M8329							

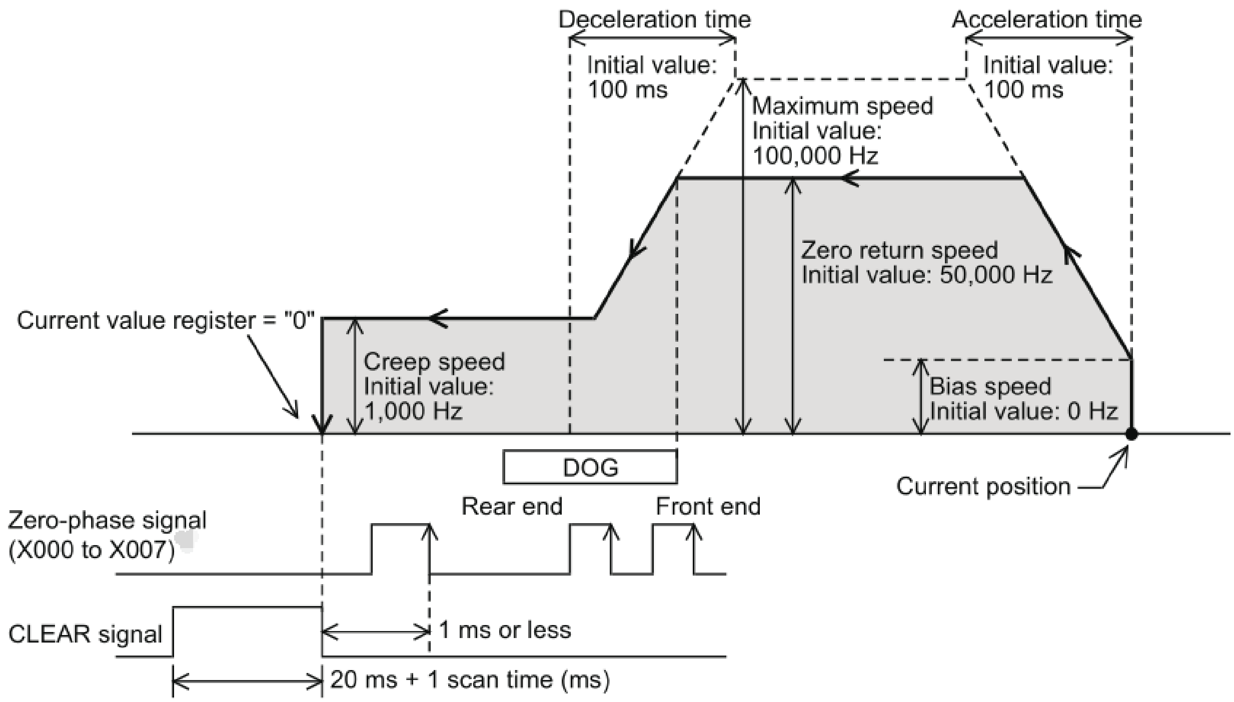
Note: M8029 spans Y0–Y7; M8329 spans Y0–Y3, Y4–Y7 empty.

Pulse operation monitoring	M8340	M8350	M8360	M8370	M8151	M8152	M8153	M8154
Clear signal output function is effective	M8341	M8351	M8361	M8371				
Zero return direction specification	M8342	M8352	M8362	M8372				
Forward limit	M8343	M8353	M8363	M8373				
Reverse limit	M8344	M8354	M8364	M8374				
Near-point signal logic inversion	M8345	M8355	M8365	M8375				
Zero signal logic inversion	M8346	M8356	M8366	M8376				
Positioning Instruction Drive	M8348	M8358	M8368	M8378				
Pulse stop bit	M8349	M8359	M8369	M8379	M8450	M8451	M8452	M8453
Clear signal device specified function is valid	M8464	M8465	M8466	M8467				

3G PLC special data register:

Pulse point / Functions	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
Current value register (32 bits)	D8340 D8341	D8350 D8351	D8360 D8361	D8370 D8371	D8140 D8141	D8142 D8143	D8144 D8145	D8160 D8161
Base speed [Hz]	D8342	D8352	D8362	D8372				
Maximum speed [Hz] (32 bits)	D8343 D8344	D8353 D8354	D8363 D8364	D8373 D8374	D8146 D8147	D8146 D8147	D8146 D8147	D8146 D8147
Creep speed [Hz]	D8345	D8355	D8365	D8375				
Return to origin speed [Hz] (32 bit)	D8346 D8347	D8356 D8357	D8366 D8367	D8376 D8377				
Acceleration time [ms]	D8348	D8358	D8368	D8378	D8148	D8148	D8148	D8148
Deceleration time [ms]	D8349	D8359	D8369	D8379				
Clear signal device designation	D8464	D8465	D8466	D8467				

18.1.2 Function and operation



1. For **S1.**, specify the near-point signal (DOG) input device number.
   To specify the logic of this near-point signal (DOG), turn the "DOG signal logic reverse" relay on or off as shown in the following table.

Pulse output destination device D1.	“DOG signal logic reverse” relay	Description
Y0	M8345	OFF: Positive logic (Turning on the input will turn on the near-point signal.) ON: Negative logic (Turning off the input will turn on the near-point signal.)
Y1	M8355	
Y2	M8365	
Y3	M8375	

2. For **S2.**, specify the zero-phase signal input number in the range of X000 to X007.
   To specify the logic of this zero-phase signal, turn the "Zero-point signal logic reverse" relay on or off as shown in the following table.
   If the same input is specified for both the near-point signal and the zero-phase signal, the logic of the zero-phase signal will be specified by the device of the near-point signal (DOG), and not by one of the following devices. In this case, the operation will be performed at the front and rear ends of the near-point signal (DOG) without using the zero-phase signal. This is similar to the operation of the ZRN instruction.

Pulse output destination device D1.	"Zero-point signal logic reverse" flag	Description
Y0	M8346	OFF: Positive logic (Turning on the input will turn on the near-point signal.) ON: Negative logic (Turning off the input will turn on the near-point signal.)
Y1	M8356	
Y2	M8366	
Y3	M8376	

3. Zero return direction

To specify the zero return direction, turn "zero return direction specification" relay on or off as shown in the following table.

Pulse output destination device D1.	"Zero return direction specification" relay	Description
Y0	M8342	To perform zero return in the forward rotation direction: Turn on the relay. To perform zero return in the reverse rotation direction: Turn off the relay.
Y1	M8352	
Y2	M8362	
Y3	M8372	

4. Zero return speed

Use the devices shown in the following table to set the zero return speed. Be sure to set the zero return speed so that the relation with the other speeds is "bias speed≦zero return speed≦maximum speed".

- If "zero return speed > maximum speed", the operation will be performed at the maximum speed.

Pulse output destination device D1.	Bias speed	Zero return speed	Maximum speed	Initial value
Y0	D8342	D8347, D8346	D8344, D8343	5000Hz
Y1	D8352	D8357, D8356	D8354, D8353	
Y2	D8362	D8367, D8366	D8364, D8363	
Y3	D8372	D8377, D8376	D8374, D8373	

5. Creep speed

"Bias speed ≦creep speed≦maximum speed".

Pulse output destination device D1.	Bias speed	Creep speed	Maximum speed	Initial value
Y0	D8342	D8345	D8344,D8343	1000Hz
Y1	D8352	D8355	D8354,D8353	
Y2	D8362	D8365	D8364,D8363	
Y3	D8372	D8375	D8374,D8373	

18.1.3 Zero return operation

1. Specify the zero return direction.

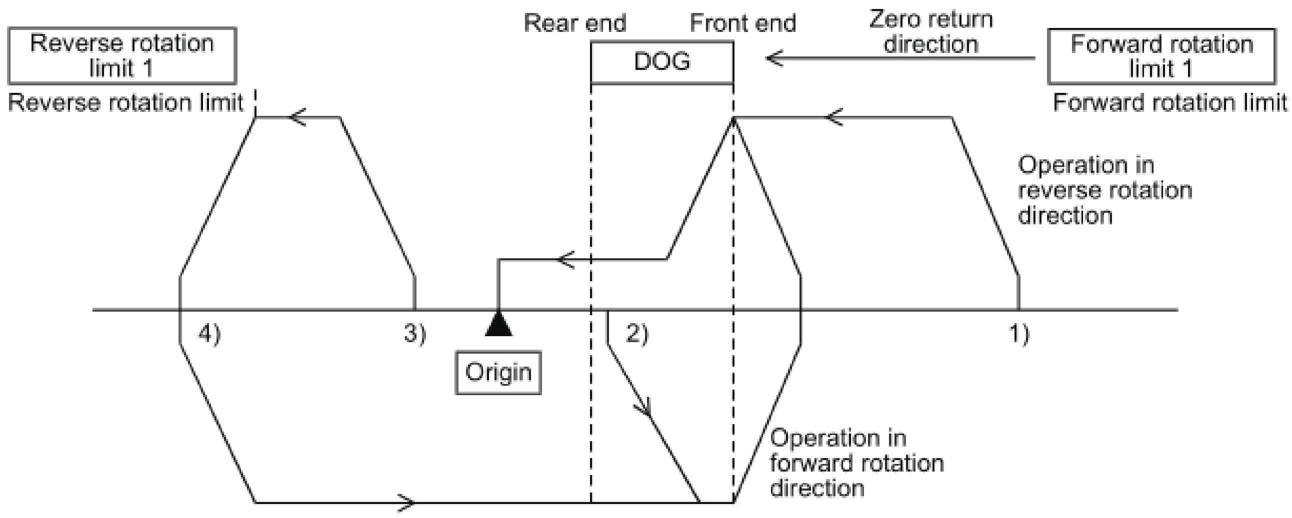
Turn the "zero return direction specification" relay (M8342) on or off to specify the zero return direction.

2. Execute the DSZR instruction to perform zero return.
3. Transfer operation will be performed in the direction specified by the "zero return direction designation" flag (M8342) at the speed specified by the "zero return speed designation" device (D8347, D8346).
4. If the near-point signal (DOG) specified by is turned on , the speed will be reduced to the creep speed (D8345).
5. After turning the near-point signal (DOG) OFF *1 , if the zero-phase signal specified by is turned on, the pulse outputting operation will immediately stop.

   If the same input is specified for both the near-point signal and the zero-phase signal, turning the near-point signal (DOG) OFF *1 will immediately stop the pulse outputting operation (just like the ZRN instruction where the zero-phase signal is not used).
6. If the CLEAR signal output function (M8341) is enabled (set to ON), the CLEAR signal (Y004) will be turned on within 1ms after the zero-phase signal is turned ON, and will be kept ON for "20 ms + 1 scan time (ms)".
7. The current value register (D8341, D8340) will be reset to "0" (will be cleared).
8. The "Instruction execution complete" flag (M8029) will turn on, and the zero return operation will be completed.

18.1.4 DOG search function

If the forward rotation limit and the reverse rotation limit are set, the DOG search function can be used for zero return. The zero return operation depends on the zero return start position.



1. If the start position is before the DOG:
   a) When the zero return instruction is executed, zero return will be started.
   b) Transfer operation will be started in the zero return direction at the zero return speed.
   c) If the front end of the DOG is detected, the speed will be reduced to the creep speed.
   d) After detecting the rear end of the DOG, if the first zero-phase signal is detected, the operation will be stopped.
2. If the start position is in the DOG area:
   a) When the zero return instruction is executed, zero return will be started.
   b) Transfer operation will be started in the opposite direction of the zero return direction at the zero return speed.
   c) If the front end of the DOG is detected, the speed will decelerate and the operation will stop. (The workpiece will come out of the DOG area.)
   d) Transfer operation will be restarted in the zero return direction at the zero return speed (and the workpiece will enter the DOG area again).
   e) If the front end of the DOG is detected, the speed will be reduced to the creep speed.

f) After detecting the rear end of the DOG, if the first zero-phase signal is detected, the operation will be stopped.

3. If the start position is in the near-point signal OFF area (after the DOG):
   a) When the zero return instruction is executed, zero return will be started.
   b) Transfer operation will be started in the zero return direction at the zero return speed.
   c) If the reverse rotation limit 1 (reverse rotation limit) is detected, the speed will decelerate, and the operation will stop.
   d) Transfer operation will be started in the opposite direction of the zero return direction at the zero return speed.
   e) If the front end of the DOG is detected, the speed will be reduced and the operation will be stopped. (The workpiece will detect the DOG and then come out of the DOG area.)
   f) Transfer operation will be restarted in the zero return direction at the zero return speed. (The workpiece will enter the DOG area again.)
   g) If the front end of the DOG is detected, the speed will be reduced to the creep speed.
   h) After detecting the rear end of the DOG, if the first zero-phase signal is detected, the operation will be stopped.
4. If the limit switch in the zero return direction turns ON (if the start position is at forward rotation limit 1 or reverse rotation limit 1):
   a) When the zero return instruction is executed, zero return will be started.
   b) Transfer operation will be started in the opposite direction of the zero return direction at the zero return speed.
   c) If the front end of the DOG is detected, the speed will decelerate and the operation will stop. (The workpiece will detect the DOG and then come out of the DOG area.)
   d) Transfer operation will be restarted in the zero return direction at the zero return speed (and the workpiece will enter the DOG area again).
   e) If the front end of the DOG is detected, the speed will be reduced to the creep speed.
   f) After detecting the rear end of the DOG, if the first zero-phase signal is detected, the operation will be stopped.

18.2 DVIT/ Interrupt Positioning

This instruction executes one-speed interrupt constant quantity feed.

Instruction	Operand type	Functions					
D FNC151 DVIT	S1. S2. D1. D2.	16-bit instruction	Mnemonic	Operation condition	32-bit instruction	Mnemonic	Operation condition
		9 steps	DVIT	Continuous operation	17 steps	DDVIT	Continuous operation

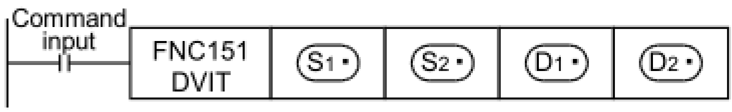
Operand			
Operand	S1.	Number of output pulses (incremental address) after interrupt *1 **Device:** KnX, KnY, KnM, KnS, T, C, D, R, K, H, decoration	BIN16/32-bit
	S2.	Output pulse frequency*2 **Device:** KnX, KnY, KnM, KnS, T, C, D, R, K, H, decoration	BIN16/32-bit
	D1.	Device number (Y) from which pulses are to be output **Device:** transistor outputs, 3G PLC: Y0~Y3, decoration	Bit
	D2.	Device number to which rotation direction signal is output **Device:** Y, M, S, decoration	Bit

*1. Setting range: −32768 to +32767 (except 0) in 16-bit operation
−999,999 to +999,999 (except 0) in 32-bit operation

*2. Setting range: 10 to 32767 Hz in 16-bit operation
10 to 100,000 Hz in 16-bit operation

Explanation of Instructions

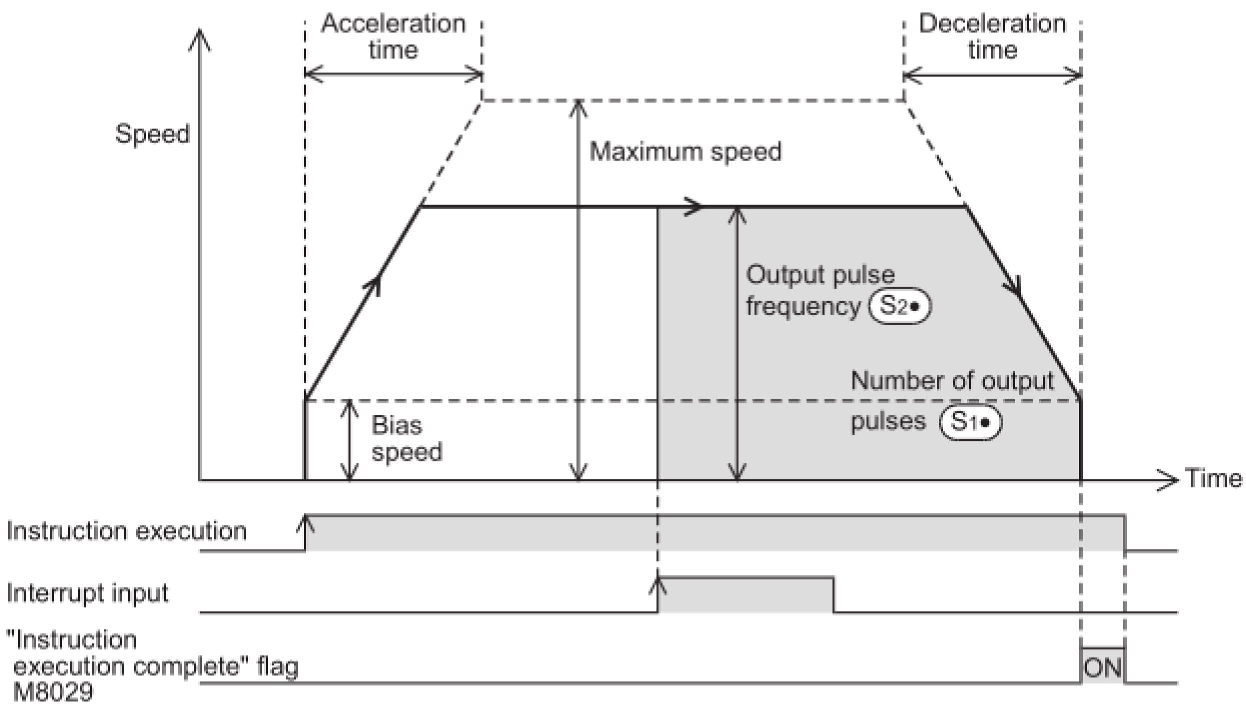
16-bit operation (DVIT)



- During RUN, avoid writing while the DVIT (FNC151) instruction is executed (that is, while a pulse is output).
  Note that if writing is executed during RUN to a circuit block including the FNC151 instruction while pulses are output, the PLC decelerates and stops pulse output.

❖ **Related devices refer to “18.1 DSZR/ Dog Search Zero Return”**

18.2.1 Function and operation



1) The rotation direction ON/OFF status of the specified device is shown in the following table.

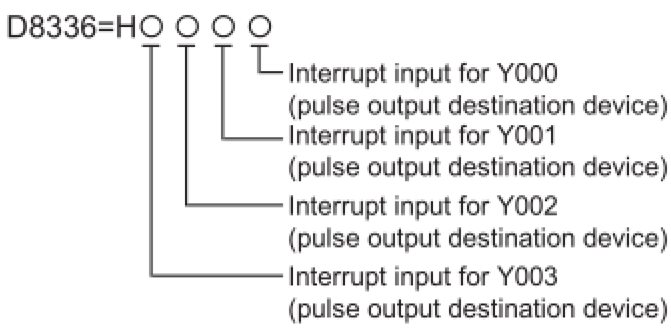
During instruction execution, however, do not use the output **D2.** for other purposes.

ON/OFF status of device specified by D2.	Rotation direction (increase/decrease current value)
ON	If the number of pulses to be output after interruption (specified by **S1.**) is set to a positive number, the operation will be performed in the forward rotation direction. Forward rotation (Outputting pulses from **D1.** will increase the current value.)
OFF	If the number of pulses to be output after interruption (specified by **S1.**) is set to a negative number, the operation will be performed in the reverse rotation direction. Reverse rotation (Outputting pulses from **D1.** will decrease the current value.)

2) Designation of interrupt input using M8336 interrupt input specification function:

a) Turn on the M8336.

b) Set the interrupt input number (X000 to X005) in D8336, or specify the user interrupt input command.



Setting value	Description of setting
0	Specifies X000 for the interrupt input signal.
1	Specifies X001 for the interrupt input signal.

<table>
<tr><td>...</td><td>...</td></tr>
<tr><td>5</td><td>Specifies X005 for the interrupt input signal.</td></tr>
<tr><td>8</td><td>Specifies a user interrupt input command for the interrupt input signal.
<table>
<tr><th>Pulse output destination device</th><th>User interrupt input command</th></tr>
<tr><td>Y0</td><td>M8460</td></tr>
<tr><td>Y1</td><td>M8461</td></tr>
<tr><td>Y2</td><td>M8462</td></tr>
<tr><td>Y3</td><td>M8463</td></tr>
</table></td></tr>
<tr><td>9~E</td><td>Do not specify these values.</td></tr>
<tr><td>F</td><td>Set "F" for a pulse output destination device if the device is not used for the Interrupt Positioning (DVIT) instruction.</td></tr>
</table>

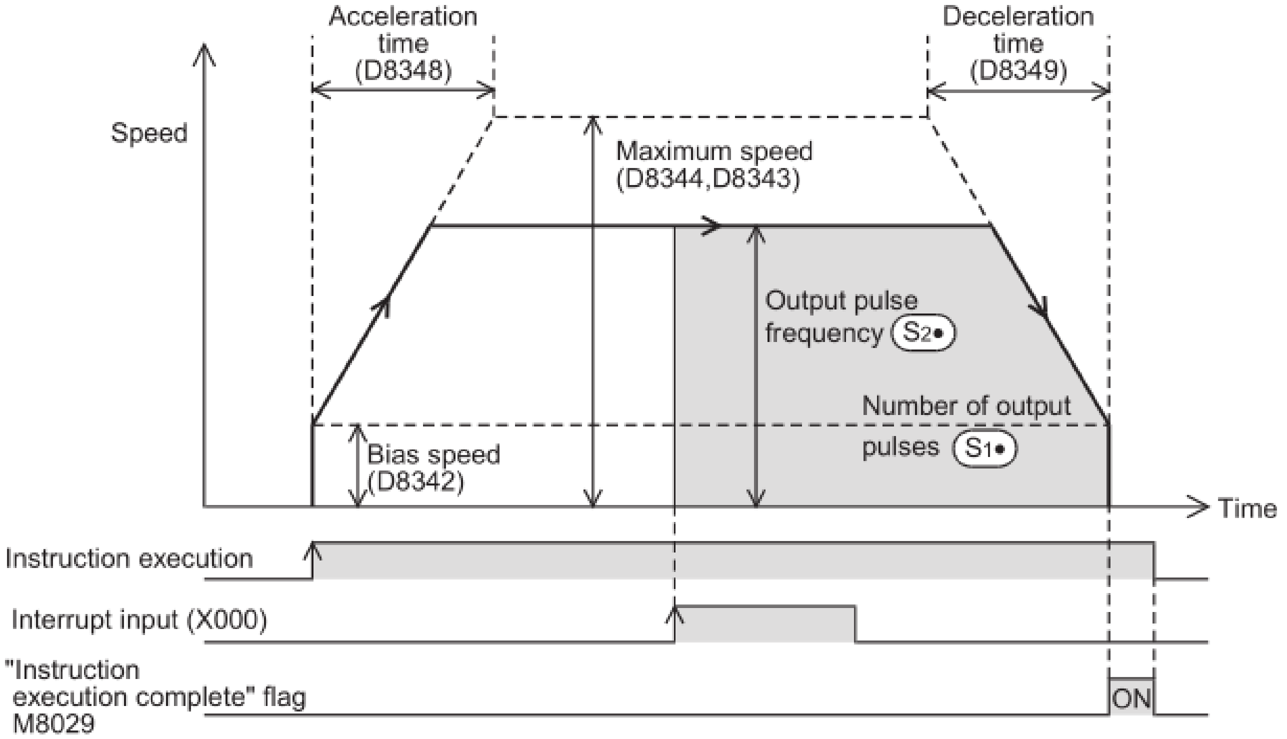
3) Interrupt input signal logical NOT

Turn the "Interrupt signal logic reverse" relay ON or OFF (see the following table) to specify the logic of the interrupt input signal. However, if the user interrupt input command is set for the pulse output destination device, the interrupt input signal logical NOT function cannot be used.

<table>
<tr><th>Pulse output destination device D1.</th><th>"Interrupt signal logic reverse" relay</th><th>Description</th></tr>
<tr><td>Y0</td><td>M8347</td><td rowspan="4">OFF: Positive logic (Turning the input ON will turn on the interrupt input signal.)
ON: Negative logic (Turning the input OFF will turn on the interrupt input signal.)</td></tr>
<tr><td>Y1</td><td>M8357</td></tr>
<tr><td>Y2</td><td>M8367</td></tr>
<tr><td>Y3</td><td>M8377</td></tr>
</table>

18.2.2 Interruption positioning operation

The interruption positioning operation is described below assuming that Y000 is specified as the pulse output destination device by **D1.**.



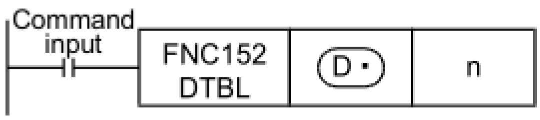
1) Execute the Interrupt Positioning (DVIT) instruction.
2) Transfer operation will be performed in the direction specified by the sign attached to the number of output pulses (specified by **S1**.) at the speed specified by the output pulse frequency (specified by **S2.**).
3) If interrupt input X000 is turned on, pulses will be output until the number of output pulses reaches the number specified by **S1.**, and then the operation will stop.
4) The "instruction execution complete" flag (M8029) will turn on, and the interruption positioning operation will be completed.
5)

❖ **Program example please refer to “DVIT instruction test” on our website:** https://en.coolmay.com/Download-177.html

18.3 TBL/ Batch Data Positioning Mode

This instruction executes one specified table operation from the data table set in GX Developer (Ver.8.23Z or later).

Instruction	Operand type	Functions						
D **FNC152** **TBL**	**D.** **n**	16-bit instruction	Mnemonic	Operation condition		32-bit instruction	Mnemonic	Operation condition
			— —			17 steps	DTBL	Continuous operation
Operand	**D.**	Device number (Y) from which pulses are to be output. **Device:** transistor output Y						Bit
	n	Table entry number [1 to 100] to be executed **Device:** K, H						32-bit binary

Explanation of Instructions	**16-bit operation (DTBL)**  ● Caution on writing during RUN Writing is disabled to a circuit block including the TBL (FNC152) instruction during RUN.

❖ **Related devices please refer to “18.1 DSZR/ Dog Search Zero Return”.**

Function and operation

Use the pulse output D of DTBL instruction and the positioning table number (n) to designate the operation D. pre-set in the "positioning setting" parameter of GX Developer, and operate according to the settings in the specified table. Set the positioning setting parameters in GX Developer (Ver. 8.23Z or later). The "Pulse number" and "Frequency" in the positioning table set in the positioning setting parameters can be changed by the program, display module, HMI, etc.

18.4 ABS/ Absolute Current Value Read

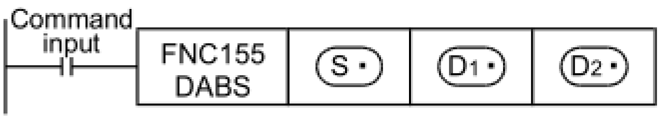
This instruction reads the absolute position (ABS) data. The data is converted into a pulse when being read.

Instruction	Operand type	Functions					
		16-bit instruction	Mnemonic	Operation condition	32-bit instruction	Mnemonic	Operation condition
FNC155 ABS D	**S. D1. D2.**	—	—		13 steps	DABS	Continuous operation

Operand	**S.**	Head device number inputting absolute (ABS) data output signal sent from servo amplifier. Three points are occupied from **S.**. **Device:** X, Y, M, S, D □.b, decoration	BIN 16 bit
	D1.	Head device number outputting absolute (ABS) data control signal to servo amplifier. Three points are occupied from **D1.**. **Device:** Y, M, S, D □.b, decoration	BIN 16 bit
	D2.	Device number storing absolute (ABS) data (32-bit value). **Device:** KnY, KnM, KnS, T, C, D, R, V, Z, decoration	BIN 32 bit

Explanation of Instructions

32-bit operation (DABS)



- Caution on writing during RUN

  Writing is disabled to a circuit block including the ABS(FNC 152) instruction during RUN.

18.4.1 Related device

2N series PLC special auxiliary relay:

	Y0	Y1	Y6	Y7	Y10
Send end flag	M8029	M8029	M8029	M8029	M8029
Pulse output stop bit	M8145	M8146	M8155	M8156	M8159
Pulse output busy flag	M8147	M8148	M8157	M8158	M8161

2N series PLC special register:

	Y0	Y1	Y6	Y7	Y10
Position pulse (32 bit)	D8140 D8141	D8142 D8143	D8150 D8151	D8152 D8153	D8154 D8155
Base speed [Hz]	D8145	D8145	D8145	D8145	D8145

Maximum speed [Hz] (32 bits)	D8146 D8147	D8146 D8147	D8146 D8147	D8146 D8147	D8146 D8147
Acceleration and deceleration time during execution	D8148	D8148	D8148	D8148	D8148

MX2N series PLC special device:

	Y0	Y1	Y2	Y3
Minimum output frequency (Default: 0)	D8145	D8145	D8159	D8159
Maximum output frequency	D8146 D8147	D8146 D8147	D8160	D8160
Acceleration and deceleration time (default: 100ms)	D8148	D8148	D8162	D8162
The output pulse stops immediately	M8145	M8146	M8155	M8156
Output pulse	M8147	M8148	M8157	M8158
Output pulse accumulation	D8140、D8141	D8142、D8143	D8154	D8156
Output pulse accumulation (Y0 and Y1)	D8136、D8137		D8166、D8167	

3G PLC special auxiliary relay

Pulse point / Function	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
Send end flag	M8029							
Instruction execution abnormal end flag	M8329							
Pulse operation monitoring	M8340	M8350	M8360	M8370	M8151	M8152	M8153	M8154
Clear signal output function is effective	M8341	M8351	M8361	M8371				
Forward limit	M8343	M8353	M8363	M8373				
Reverse limit	M8344	M8354	M8364	M8374				
Positioning Instruction driver	M8348	M8358	M8368	M8378				
Pulse stop bit	M8349	M8359	M8369	M8379	M8450	M8451	M8452	M8453
Clear signal device specified function is valid	M8464	M8465	M8466	M8467				

3G PLC special register

Pulse point / Function	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
Current value register (32 bits)	D8340 D8341	D8350 D8351	D8360 D8361	D8370 D8371	D8140 D8141	D8142 D8143	D8144 D8145	D8160 D8161
Base speed [Hz]	D8342	D8352	D8362	D8372				
Maximum speed [Hz] (32 bits)	D8343 D8344	D8353 D8354	D8363 D8364	D8373 D8374	D8146 D8147	D8146 D8147	D8146 D8147	D8146 D8147

Acceleration time [ms]	D8348	D8358	D8368	D8378	D8148	D8148	D8148	D8148
Deceleration time [ms]	D8349	D8359	D8369	D8379				
Clear signal device designation	D8464	D8465	D8466	D8467				

1. Function and Operation

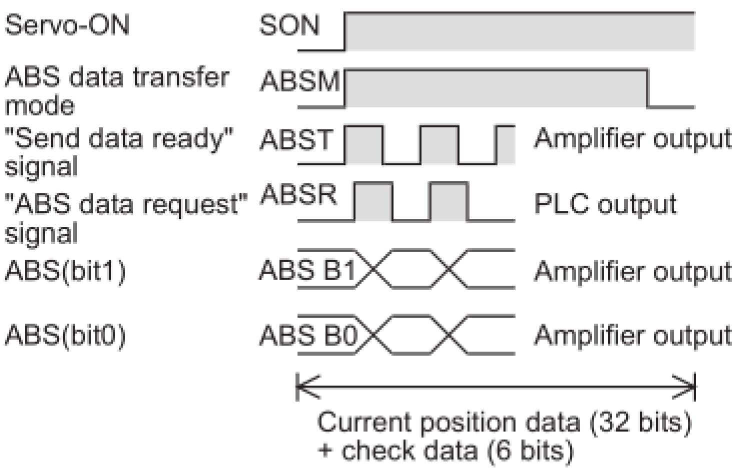
1) For **S.**, specify the first number of the device that inputs the absolute position (ABS) data from the servo amplifier. Number of occupied points: 3 (**S.** is ABS (bit 0), **S.+1** is ABS (bit 1), and **S.+2** is the "send data ready" signal.)

2) For **D1.**, specify the first number of the device that outputs the absolute position (ABS) data control signal to the servo amplifier. Be sure to use transistor outputs for the PLC outputs. Number of occupied points: 3 (**D1.** is the "servo-ON" signal, **D1.+1** is the ABS data transfer mode, and **D1.+2** is the "ABS data request" signal.)

3) For **D2.**, specify the absolute position (ABS) data (32-bit value) storage device number to store the data read out from the servo amplifier. Handle the absolute position (ABS) data as the table above.

2. Detection of absolute position

1) If the DABS (FNC155) instruction turns ON, the PLC will activate the servo-ON output and the ABS transfer mode output.
2) 32+6-bit data communication will be performed while mutually checking the data sending/receiving condition using the "send data ready" signal and the "ABS data request" signal.
3) The 2-bit line (line for ABS bit 0 and bit 1) will be used for data transmission.
4) At the completion of ABS data reading, the "Instruction execution complete" flag (M8029) will turn on.

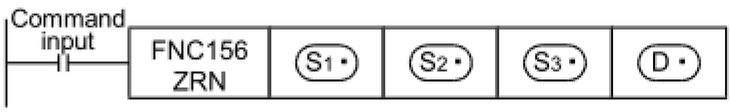


18.5 ZRN/ Zero Return

Instruction for reading absolute position (ZRN) data. The data is read out as pulse converted values.

<table>
<tr><th>Instruction</th><th>Operand type</th><th colspan="7">Functions</th></tr>
<tr><td rowspan="2">D
FNC156
ZRN</td><td rowspan="2">S1.
S2.
S3.
D1.</td><td>16-bit instruction</td><td>Mnemonic</td><td>Operation condition</td><td rowspan="2"></td><td>32-bit instruction</td><td>Mnemonic</td><td>Operation condition</td></tr>
<tr><td>9 steps</td><td>ZRN
—</td><td>Continuous operation</td><td>17 steps</td><td>DZRN</td><td>Continuous operation</td></tr>
<tr><td rowspan="4">Operand</td><td>S1.</td><td colspan="6">Initial zero return speed [*1]
Device: KnX, KnY, KnM, KnS, T, C, D, R, K, H, decoration</td><td rowspan="2">16- or 32-bit binary</td></tr>
<tr><td>S2.</td><td colspan="6">Creep speed [10 to 32767 Hz], decoration
Device: KnX, KnY, KnM, KnS, T, C, D, R, K, H</td></tr>
<tr><td>S3.</td><td colspan="6">Device number for near-point signal (dog)
Device: X, Y, M, S, D □.b, decoration</td><td rowspan="2">Bit</td></tr>
<tr><td>D1.</td><td colspan="6">Device number (Y) from which pulses are to be output
Device: Y, M, S, decoration</td></tr>
</table>

Note:*1. Setting range: 10 to 32767 Hz for 16-bit operation, 10 to 100,000 for 32-bit operation

<table>
<tr><td>Explanation of Instructions</td><td>16-bit operation (ZRN)

● During RUN, avoid writing while the ZRN (FNC156) instruction is executed (that is, while pulses are output).
Note that if writing is executed during RUN to a circuit block including the FNC156 instruction while pulses are output, the PLC decelerates and stops pulse output.</td></tr>
</table>

18.5.1 Related devices

2N PLC special auxiliary relay:

	Y0	Y1	Y6	Y7	Y10
Send end flag	M8029	M8029	M8029	M8029	M8029
Pulse output stop bit	M8145	M8146	M8155	M8156	M8159
Pulse output busy flag	M8147	M8148	M8157	M8158	M8161

2N PLC special register

	Y0	Y1	Y6	Y7	Y10
Position pulse	D8140	D8142	D8150	D8152	D8154

(32 bit)	D8141	D8143	D8151	D8153	D8155
Base speed [Hz]	D8145	D8145	D8145	D8145	D8145
Maximum speed [Hz] (32 bits)	D8146 D8147	D8146 D8147	D8146 D8147	D8146 D8147	D8146 D8147
Acceleration and deceleration time during execution	D8148	D8148	D8148	D8148	D8148

MX2N PLC special devices

	Y0	Y1	Y2	Y3
Minimum output frequency (Default: 0)	D8145	D8145	D8159	D8159
Maximum output frequency	D8146 D8147	D8146 D8147	D8160	D8160
Acceleration and deceleration time (Default: 100ms)	D8148	D8148	D8162	D8162
The output pulse stops immediately	M8145	M8146	M8155	M8156
Output pulse	M8147	M8148	M8157	M8158
Output pulse accumulation	D8140、D8141	D8142、D8143	D8154	D8156
Output pulse accumulation (Y0 and Y1)	D8136、D8137		D8166、D8167	

3G PLC special auxiliary relay

Pulse point / Function	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
Send end flag	M8029							
Instruction execution abnormal end flag	M8329							
Pulse operation monitoring	M8340	M8350	M8360	M8370	M8151	M8152	M8153	M8154
Clear signal output function is effective	M8341	M8351	M8361	M8371				
Forward limit	M8343	M8353	M8363	M8373				
Reverse limit	M8344	M8354	M8364	M8374				
Positioning Instruction driver	M8348	M8358	M8368	M8378				
Pulse stop bit	M8349	M8359	M8369	M8379	M8450	M8451	M8452	M8453

Clear signal device specified function is valid	M8464	M8465	M8466	M8467				

3G PLC special register

Pulse point / Function	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
Current value register (32 bits)	D8340 D8341	D8350 D8351	D8360 D8361	D8370 D8371	D8140 D8141	D8142 D8143	D8144 D8145	D8160 D8161
Base speed [Hz]	D8342	D8352	D8362	D8372				
Maximum speed [Hz] (32 bits)	D8343 D8344	D8353 D8354	D8363 D8364	D8373 D8374	D8146 D8147	D8146 D8147	D8146 D8147	D8146 D8147
Acceleration time [ms]	D8348	D8358	D8368	D8378	D8148	D8148	D8148	D8148
Deceleration time [ms]	D8349	D8359	D8369	D8379				
Clear signal device designation	D8464	D8465	D8466	D8467				

18.5.2 Function and Operation

1) For **S3.**, specify the near-point signal (DOG) input device number (NO contact).
   Turning on the near-point signal will reduce the speed to the creep speed. Turning off the near-point signal will complete the zero return operation.
2) Zero return direction
   For this instruction, the zero return direction is set to the reverse rotation direction.
   (During zero return operation, the value indicated in the current value register will be decreased.)
   To perform zero return in the forward rotation direction, follow the example program below to control the direction output.
   a) Turn on Y□□□ (rotational direction signal).
   b) Refresh Y□□□ output using the REF (FNC 50) instruction.
   c) Execute the ZRN instruction (zero return instruction).
   d) With the execution completion flag (M8029) of the ZRN instruction (zero return instruction), reset Y□□□ (rotational direction signal).

18.5.3 Zero return operation

Zero return operation is described below assuming that Y000 is specified as the pulse output destination device **D.**.

1) Execute the ZRN instruction to carry out zero return.
2) Transfer operation will be performed at the zero return speed specified by **S1.**.

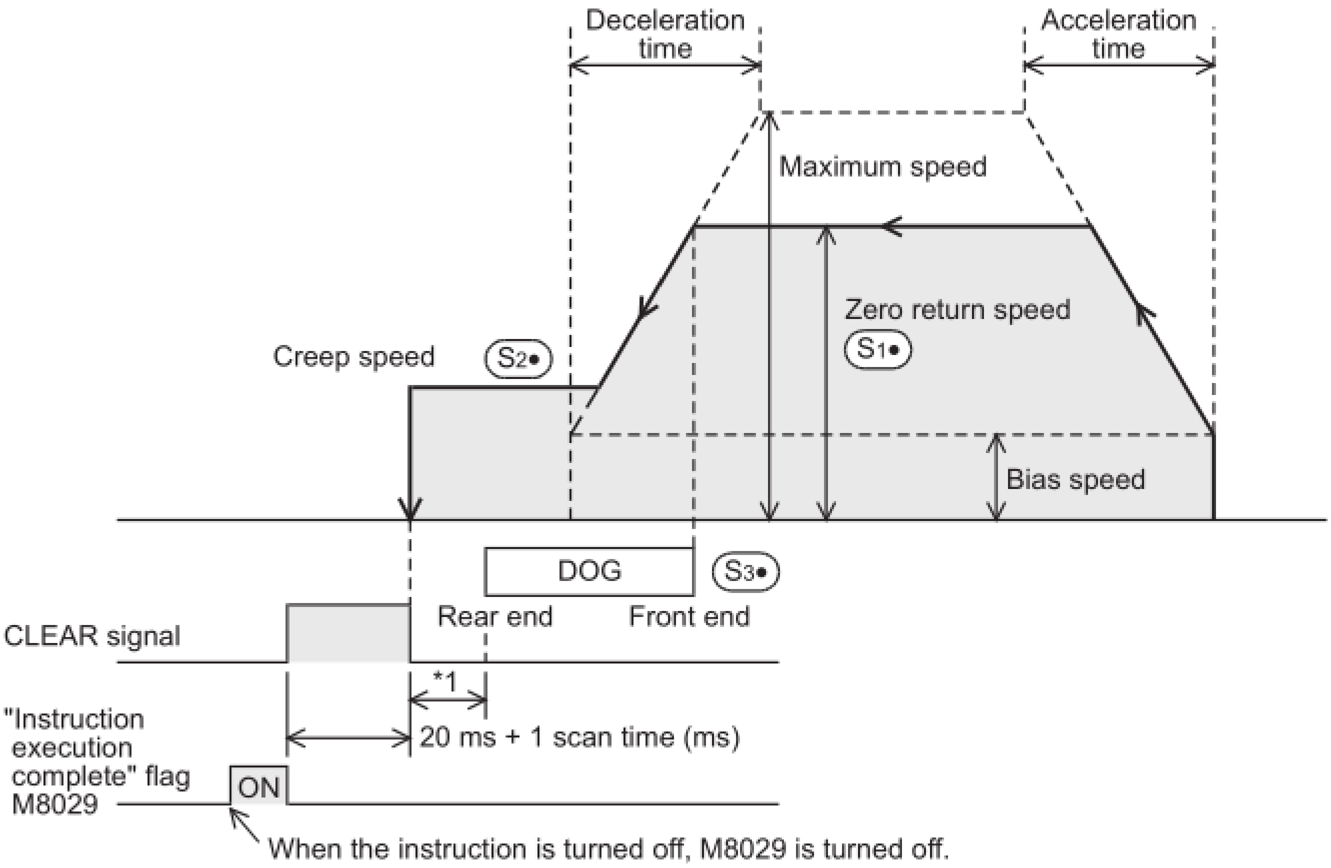
3) If the near-point signal (DOG) specified by **S3.** is turned on, the speed will be reduced to the creep speed specified by **S2.**.

4) If the near-point signal (DOG) specified by **S3.** is turned off, the pulse outputting operation will be immediately stopped.

5) If the CLEAR signal output function (M8341) is enabled (set to ON), the CLEAR signal (Y004) will be turned on within 1 ms *1 after the near-point signal (DOG) is turns from ON to OFF, and will be kept ON for "20 ms + 1 scan time (ms)". *2

6) The current value register (D8341, D8340) will be reset to "0" (will be cleared).

7) "Instruction execution complete" flag will be turned on, and the zero return operation will be completed.



18.5.4 Cautions

- If the near-point input signal in **S3.** is specified as X000 to X007, the PLC interruption function will be used to stop the operation.
  Under the following condition, however, operation may be affected by the input filter or the scan time of the sequence program.

  - An input number of X010 or higher (or other device (auxiliary relay, etc.)) is specified.

  If input relay X010 or higher is specified for the near-point signal (DOG), the effects of the input filter will be applied.

- If an input X000 to X007 (X000 to X005 for FX 3S PLC) of the main unit is specified for the near-point signal (DOG), the input cannot be used for the following items:

  - High-speed counter
  - Input interruption
  - Pulse catch

- SPD instruction
- DSZR instruction
- DVIT instruction *1

- Properly set the DOG so that the near-point signal (DOG) can be kept ON until the speed is reduced to the creep speed.
  This instruction will start speed reduction at the front end of the DOG, and will stop the operation at the rear end of the DOG. The current value register will then be cleared (reset to "0").
  If the speed is not reduced to the creep speed before detecting the rear end of the DOG, the operation may not be stopped at the specified position.
- The creep speed should be sufficiently slow.
  The zero return instruction will not decelerate at the stop point. Therefore, if the creep speed is not slow enough, the operation may not stop at the specified position due to inertia.
- The zero-phase signal of the servo motor cannot be used. For this reason, if fine adjustment of the origin position is needed, adjust the position of the near-point signal (DOG).
- If the instruction activation contact is turned off during zero return operation, the speed will decelerate and the operation will stop. In this case, the "Instruction execution complete" flag (M8029) will not turn on.
- While the "pulse output monitor" (BUSY/READY) flag is on, a positioning instruction (including PLSR and PLSY) that uses the same output cannot be executed.
  If the "pulse output monitor" (BUSY/READY) flag is still on after the instruction activation contact is turned off, do not execute a positioning instruction (including PLSR and PLSY instructions) that uses the same output number.
- In the following case, the "Instruction execution abnormal end" flag (M8329) will be turned on, and execution of the instruction will be completed.

- If the forward limit relay or the reverse limit relay is turned on, the speed will decelerate and the operation will stop. In this case, the "Instruction execution abnormal end" flag (M8329) will be turned on when execution of the instruction is complete.
- If the limit relay (forward or reverse) on the opposite side of the operation direction is turned on, the speed will decelerate and the operation will stop.
  In this case, the "Instruction execution abnormal end" flag (M8329) will be turned on when execution of the instruction is complete.。

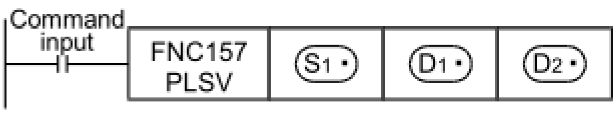
18.6 PLSV/ Variable Speed Pulse Output

This instruction outputs variable speed pulses with an assigned rotation direction.

Instruction	Operand type	Functions						
		16-bit instruction	Mnemonic	Operation condition		32-bit instruction	Mnemonic	Operation condition
FNC157 PLSV **D**	**S1. D1. D2.**	9 steps	PLSV —	Continuous operation		17 steps	DPLSV	Continuous operation

Operand	**S1.**	Device number for output pulse frequency[*1] **Device:** KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, decoration	16- or 32-bit binary
	D1.	Device number (Y) from which pulses are to be output **Device:** transistor output Y, decoration	Bit
	D2.	Device number to which rotation direction signal is output **Device:** Y, M, S, D □.b, decoration	

Note:*1. Setting range: −32768 to +32767 Hz for 16-bit operation, -100,000 to 000 Hz for 32-bit operation.

Explanation of Instructions	**16-bit operation (PLSV)**  ● During RUN, avoid writing while PLSV (FNC157) instruction is executed (that is, while pulses are output). Note that if writing is executed during RUN to a circuit block including FNC157 instruction while pulses are output, the PLC executes the operation shown below.

❖ **Related devices refer to 18.5 ZRN/ Zero Return**

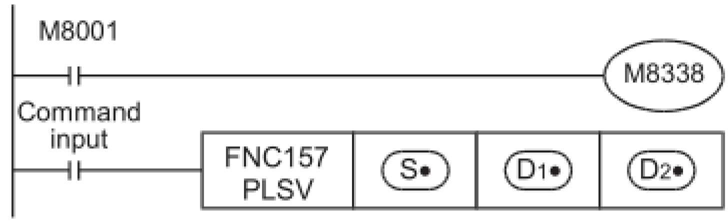
18.6.1 Function and operation

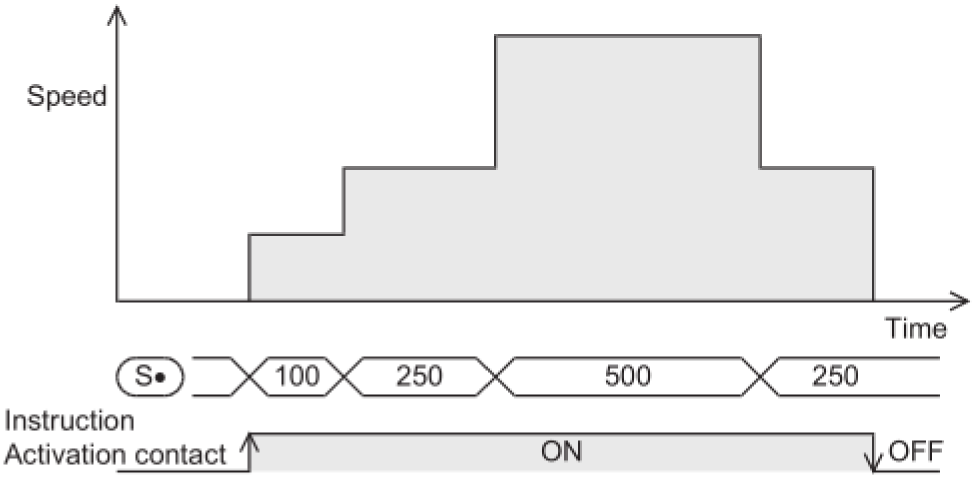
The variable speed pulse output instruction changes the speed while using the rotation direction output.

The acceleration/deceleration function applies for the variable speed pulse output (PLSV) instruction, which makes it possible to specify whether acceleration/deceleration will be used or not.

18.6.1.1 Operation without Acceleration/Deceleration (M8338 = OFF)

If the output pulse frequency **S.** value is changed after turning the acceleration/deceleration function (M8338) OFF, the variable speed pulse output (PLSV) instruction will change the output frequency without using acceleration/deceleration.





1) For **S.**, specify the output pulse frequency.

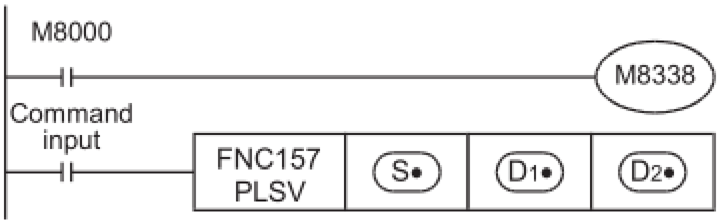
Even if pulses are being output, the output pulse frequency **S.** can be changed freely. Acceleration/deceleration, however, will not be performed.

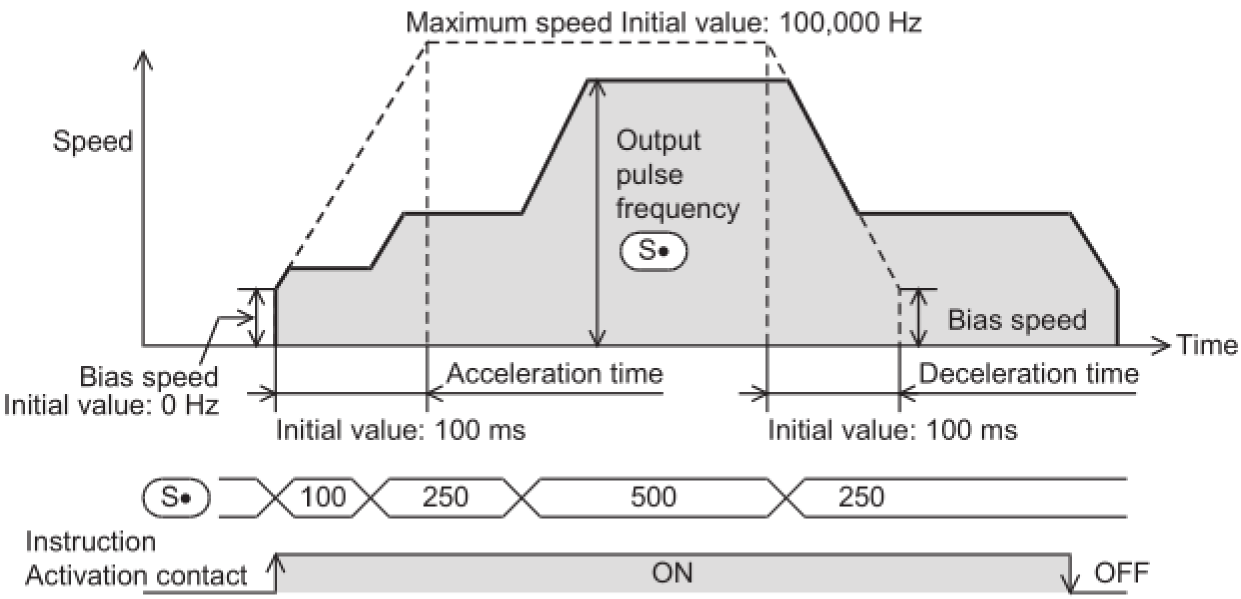
2) The rotation direction ON/OFF status of the specified device is shown in the following table.

ON/OFF status of device specified by D2.	Rotation direction (increase/decrease current value)
ON	If the number of output pulses specified by **S1.** is a positive number, the operation will be performed in the forward rotation direction. Forward rotation (Outputting pulses from **D1.** will increase the current value.)
OFF	If the number of output pulses specified by **S1.** is a negative number, the operation will be performed in the reverse rotation direction. Reverse rotation (Outputting pulses from **D1.** will decrease the current value.)

18.6.1.2 Operation with Acceleration/Deceleration (M8338 = ON)

If the output pulse frequency **S.** value is changed after turning the acceleration/deceleration (M8338) ON, the variable speed pulse output (PLSV) instruction will accelerate or decelerate to the changed output.





1) For **S.**, specify the output pulse frequency.

Even if pulses are being output, the output pulse frequency **S.** can be changed freely. Acceleration/deceleration will be performed.

2) The rotation direction ON/OFF status of the specified device is shown in the following table.

ON/OFF status of device specified by S.	Rotation direction (increase/decrease current value)
ON	If the number of output pulses specified by **S.** is a positive number, the operation will be performed in the forward rotation direction. Forward rotation (Outputting pulses from **D1.** will increase the current value.)
OFF	If the number of output pulses specified by **S.** is a negative number, the operation will be performed in the reverse rotation direction. Reverse rotation (Outputting pulses from **D1.** will decrease the current value.)

18.6.2 Important points

• During pulse output operation, if the output pulse frequency is changed to "K0", the PLC will reduce the speed and then stop the pulse outputting operation if the acceleration/deceleration function is ON.

However, if the acceleration/deceleration function is not activated, the PLC will immediately stop the pulse outputting operation.

Before outputting pulses again, check that the "pulse output monitor" (BUSY/READY) flag is off, and then wait until 1 or more cycles of operation have been completed. After that, set (change) the output pulse frequency to a value other than "K0".

• During pulse outputting operation, do not change the sign attached to the output pulse frequency value **S.**.

If it is necessary to change the sign, stop the servo motor first by setting the output pulse frequency value **S.** to "K0", and wait for the motor to stop completely after decelerating to stop. And then, change the sign attached to the output pulse frequency value **S.**.

If the sign attached to the output pulse frequency value **S.** is changed during pulse outputting operation, the operation may be changed as follows, and the machine, therefore, may be damaged:

1) The pulse outputting operation may be stopped.

2) "Pulse output monitor" (BUSY/READY) flag may be turned off.

(The pulse outputting operation may be stopped, but the motor may not be stopped immediately.)

3) Operation may be performed in the specified direction at the frequency specified by the output pulse frequency value **S.**.

• If the instruction activation contact is turned off during pulse outputting operation while the acceleration/deceleration function is ON, the speed will decelerate and the operation will stop.

If the instruction activation contact is turned off during pulse outputting operation while the acceleration/deceleration function is OFF, the operation will stop immediately.

The "Instruction execution complete" flag (M8029) will not turn on.

• If a limit flag (forward rotation or reverse rotation) in the operation direction is turned ON, the speed will decelerate and the operation will stop in the case that the acceleration/deceleration function is ON. In this case, the "Instruction execution abnormal end" flag (M8329) will turn on when execution of the instruction is complete.

• If the "pulse output monitor" (BUSY/READY) flag is on, a positioning instruction (including PLSR and PLSY) that uses the same output cannot be executed.

If the "pulse output monitor" (BUSY/READY) flag is still on after the instruction activation contact is turned off, do not execute a positioning instruction (including PLSR and PLSY instructions) that uses the same output number.

• After executing the instruction, the rotation direction signal output will turn off.

18.7 DRVI/ Drive to Increment

This instruction executes one-speed positioning by incremental drive. The movement distance from the present position can be specified with a positive or negative sign.

Instruction	Operand type	Functions						
D **FNC158 DRVI**	**S1. S2. D1. D2.**	16-bit instruction	Mnemonic	Operation condition		32-bit instruction	Mnemonic	Operation condition
		9 steps	DRVI	Continuous operation		17 steps	DDRVI	Continuous operation
			—					

Operand	Operand type	Description	Type
Operand	**S1.**	Number of output pulses (relative address) *1 **Device:** KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, decoration	BIN16/32 bit
	S2.	Output pulse frequency*2 **Device:** KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, decoration	
	D1.	Device number (Y) from which pulses are to be output **Device:** transistor output Y, decoration	Bit
	D2.	Device number to which rotation direction signal is output **Device:** Y, M, S, D □.b, decoration	

Note:*1. Setting range: −32768 to +32767 (except 0) for 16-bit operation, and −999,999 to +999,999 (except 0) for 32-bit operation

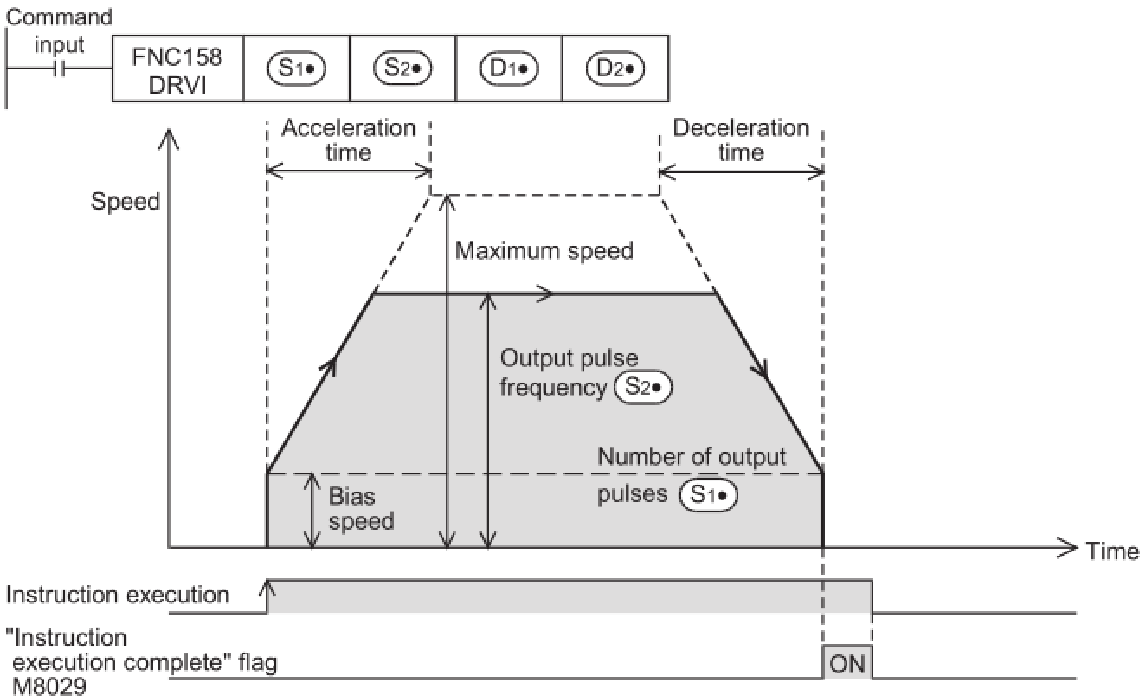
*2. Setting range: 10 to 32767 Hz for 16-bit operation, and 10 to 100,000 Hz for 32-bit operation

Explanation of Instructions	● During RUN, avoid writing while DRVI (FNC158) instruction is executed (that is, while pulses are output). Note that if writing is executed during RUN to a circuit block including FNC158 instruction while pulses are output, the PLC decelerates and stops pulse output.

❖ **For related devices, refer to "18.1 DSZR/ Dog Search Zero Return".**

18.7.1 Function and Operation

This instruction uses a relative drive method to perform a 1-speed positioning instruction. For this instruction, the transfer distance from the current position to the target position should be specified together with a plus or minus sign. This method is also referred to as the incremental (relative) drive method.



The rotation direction ON/OFF status of the specified device is shown in the following table.

ON/OFF status of device specified by D2.	Rotation direction (increase/decrease current value)
ON	If the number of output pulses specified by **S1.** is a positive number, the operation will be performed in the forward rotation direction. Forward rotation (Outputting pulses from **D1.** will increase the current value.)
OFF	If the number of output pulses specified by **S1.** is a negative number, the operation will be performed in the reverse rotation direction. Reverse rotation (Outputting pulses from **D1.** will decrease the current value.)

18.7.2 Important Points

- If the instruction activation contact is turned off during execution of the instruction, the speed will decelerate and the operation will stop.
  In this case, the "Instruction execution complete" flag (M8029) will not be turned on.
- If the limit flag (forward or reverse) in the operation direction is turned on, the speed will decelerate and the operation will stop.
  In this case, the "Instruction execution abnormal end" flag (M8329) will be turned on when execution of the instruction is complete.
- The direction of rotation is specified by the sign of **S1.**.

18.8 DRVA/ Drive to Absolute

This instruction uses an absolute drive method to perform a 1-speed positioning instruction.

For this instruction, the distance from the origin (zero-point) to the target position should be specified.

Instruction	Operand type	Functions						
D FNC159 DRVA	S1. S2. D1. D2.	16-bit instruction	Mnemonic	Operation condition		32-bit instruction	Mnemonic	Operation condition
		9 steps	DRVA —	Continuous operation		17 steps	DDRVA	Continuous operation

Operand			
Operand	S1.	Number of output pulses (absolute address)*1 **Device:** KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, decoration	BIN16/32 bits
	S2.	Output pulse frequency*2 **Device:** KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, decoration	
	D1.	Device number (Y) from which pulses are to be output **Device:** transistor output Y, decoration	Bit
	D2.	Device number to which rotation direction signal is output **Device:** Y, M, S, D □.b, decoration	

Note:*1. Setting range: −32768 to +32767 (except 0) for 16-bit operation, and −999,999 to +999,999 (except 0) for 32-bit operation

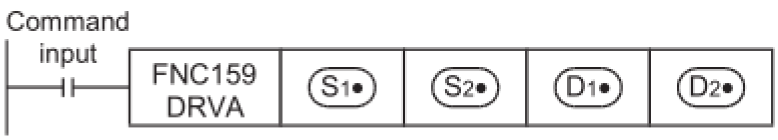
*2. Setting range: 10 to 32767 Hz for 16-bit operation, and 10 to 100,000 Hz for 32-bit ope

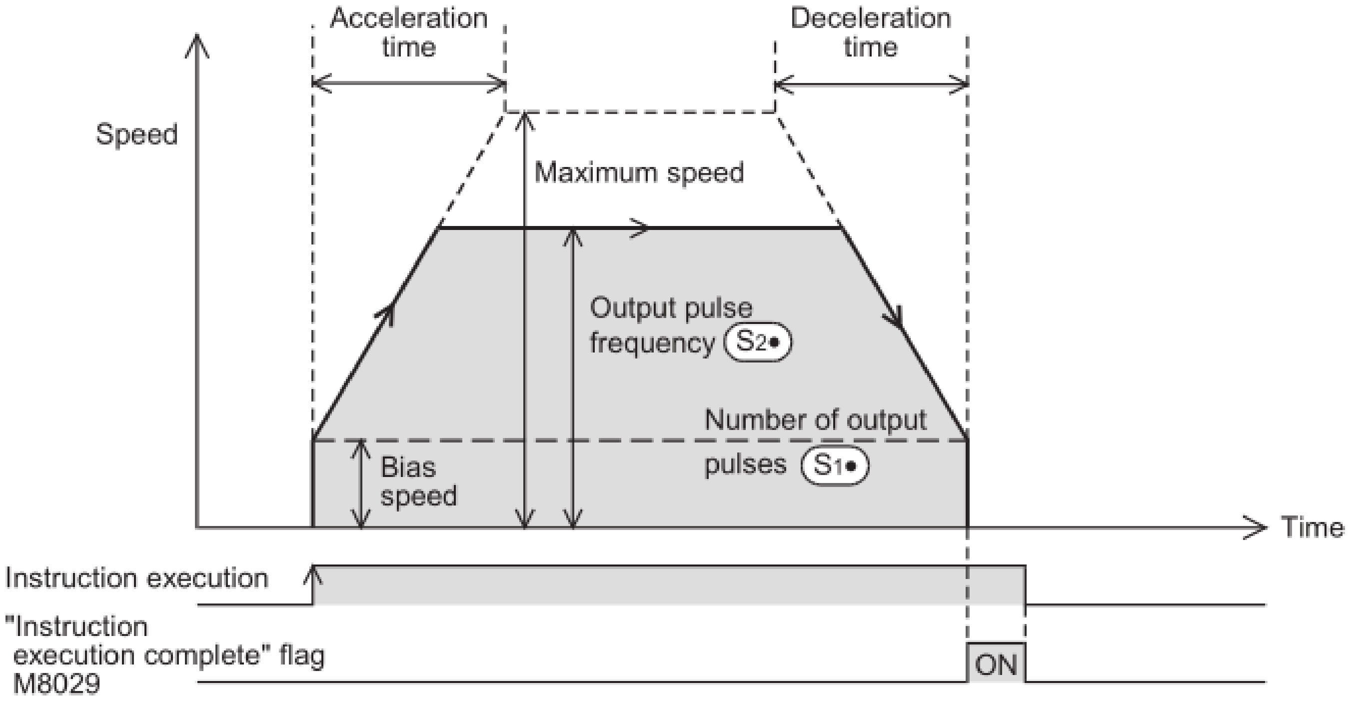
Explanation of Instructions	● During RUN, avoid writing while DRVA (FNC159) instruction is executed (that is, while pulses are output). Note that if writing is executed during RUN to a circuit block including FNC159 instruction while pulses are output, the PLC decelerates and stops pulse output.

❖ **For related devices, refer to "18.1 DSZR/ Dog Search Zero Return".**

18.8.1 Function and Operation

This instruction executes one-speed positioning by absolute drive. The movement distance from the zero point can be specified.





The rotation direction ON/OFF status of the specified device is shown in the following table. During instruction execution, however, do not use the output for other purposes.

ON/OFF status of device specified by D2.	Rotation direction (increase/decrease current value)	
ON	Forward rotation (Outputting pulses from **D1.** will increase the current value.)	The rotation direction (normal or reverse rotation) depends on which value is larger; the number of output pulses specified by **S.** (absolute address) or the value indicated in the current value register.
OFF	Reverse rotation (Outputting pulses from **D1.** will reduce the current value.)	

18.8.2 Important Points

- If the instruction activation contact is turned off during execution of the instruction, the speed will decelerate and the operation will stop.
  In this case, the "Instruction execution complete" flag (M8029) will not be turned on.
- If the limit flag (forward or reverse) in the operation direction is turned on, the speed will decelerate and the operation will stop. In this case, the "Instruction execution abnormal end" flag (M8329) will be turned on when execution of the instruction is complete.

19 Real Time Clock Operation

FNC NO.	Instruction	Function	Device		
			3G PLC	2N PLC	MX2N PLC
160	TCMP	Clock data comparison	★	★	★
161	TZCP	Clock data interval comparison	★	★	★
162	TADD	Clock data addition	★	★	★
163	TSUB	Clock data subtraction	★	★	★
164	HTOS	Second conversion of hour, minute and second data	★		
165	STOH	[Hour, Minute, Second] conversion of second data	★		
166	TRD	Read clock data	★	★	★
167	TWR	Write clock data	★	★	★
168	—				
169	HOUR	Chronograph	★	★	★

19.1 TCMP/ RTC Data Compare

This instruction compares the comparison time with the time data, and turns ON or OFF bit devices according to the comparison result.

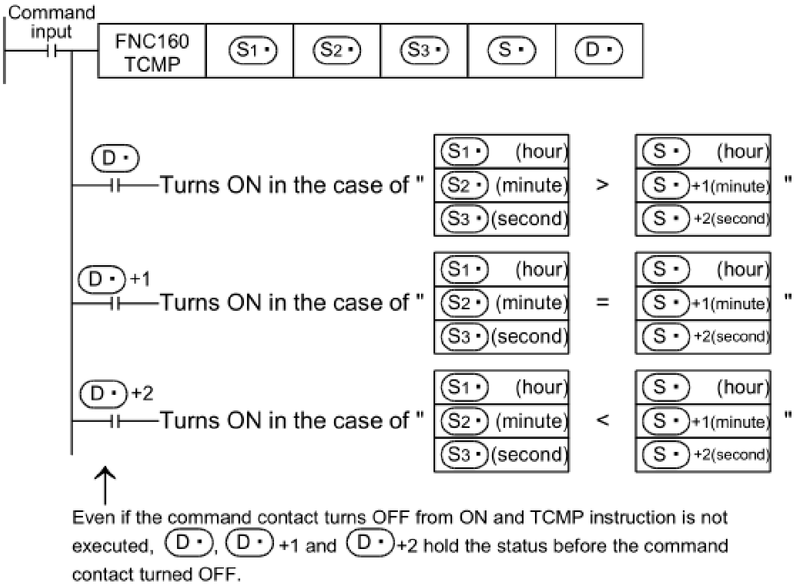
Instruction	Operand type	Functions					
		16-bit instruction	Mnemonic	Operation condition	32-bit instruction	Mnemonic	Operation condition
FNC160 TCMP	S1. S2. S3. S. D.	11 steps	TCMP	Continuous operation		—	
P			TCMPP	Pulse operation			

Operand	Operand type	Description	Data type
Operand	S1.	Specifies "hour" of the comparison time [setting range: 0 to 23]. **Device:** KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, decoration	16-bit binary
	S2.	Specifies "minute" of the comparison time [setting range: 0 to 59]. **Device:** KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, decoration	
	S3.	Specifies "second" of the comparison time [setting range: 0 to 59]. **Device:** KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, decoration	
	S.	Specifies "hour" of the time data (hour, minute, and second). (Three devices are occupied.) **Device: T, C, D, R, decoration**	
	D.	Turns ON or OFF according to the comparison result. **Device:** Y, M, S, D □.b, decoration	Bit

Explanation of Instructions

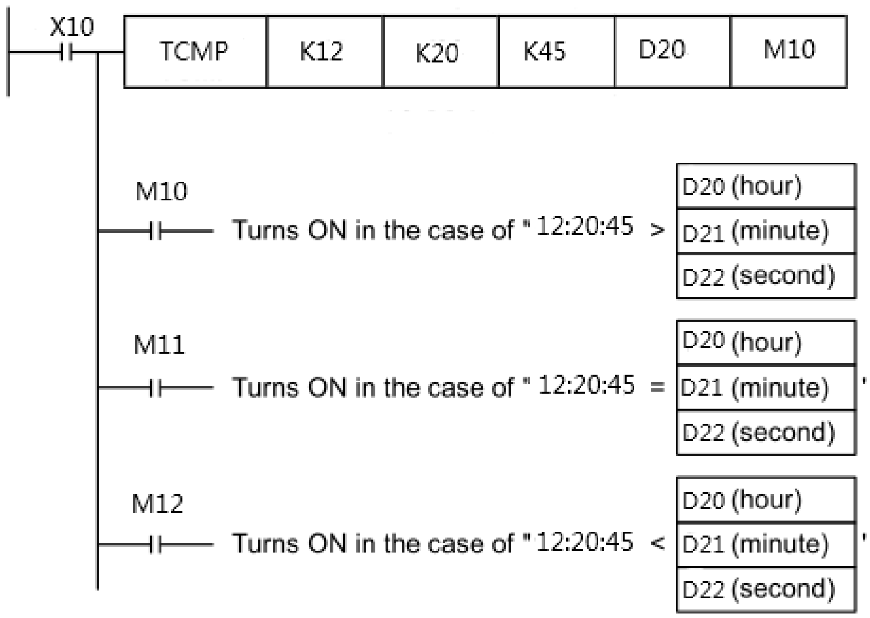
1. 16-bit operation (TCMP, TCMPP)

The comparison time (hour, minute, and second) stored in **S1.**, **S2.**, and **S3.** is compared with the time data (hour, minute, and second) stored in **S.**, **S.+1**, and **S.+2.** Three devices starting from **D.** turn ON or OFF according to the comparison result.



	● When utilizing the time (hour, minute, and second) of the real time clock built in a PLC, please use TRD (FNC 166) Instruction. Read the values of special data registers by TRD (FNC166) instruction, and then specify those word devices as the operands.

- **Program example**



◆ When X10=ON, Instruction is executed, the current time of the D20~D22 calendar is compared with the set value 12:20:45, and the result is displayed to M10~M12. When X10 changes from ON→OFF, the instruction is not executed, but the ON/OFF state before M10~M12 is still maintained.

◆ If you need to get the results of ≧, ≦, ≠, you can get by connecting M10~M12 in series and parallel.

19.2 TZCP/ RTC Data Zone Compare

This instruction compares two comparison time (comparison time zone) with the time data, and turns ON or OFF the specified bit devices according to the comparison results.

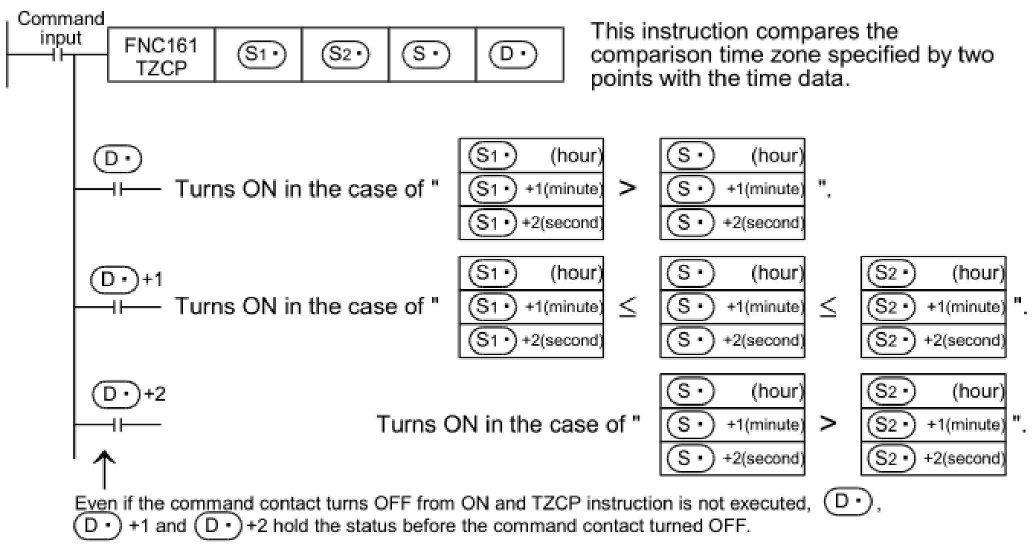
Instruction	Operand type	Functions					
		16-bit instruction	Mnemonic	Operation condition	32-bit instruction	Mnemonic	Operation condition
FNC161 TZCP / **P**	**S1.** **S2.** **S.** **D.**	11 steps	TZCP TZCPP	Continuous operation Pulse operation		—	

Operand			
Operand	**S1.**	Specifies "hour" of the lower limit comparison time (hour, minute, and second). (Three devices are occupied.) **Device: T, C, D, R, decoration**	16-bit binary
	S2.	Specifies "hour" of the upper limit comparison time (hour, minute, and second). (Three devices are occupied.) **Device: T, C, D, R, decoration**	
	S.	Specifies "hour" of the time data (hour, minute, and second). (Three devices are occupied.) **Device: T, C, D, R, decoration**	
	D.	Turns ON or OFF according to the comparison result. (Three devices are occupied.) **Device:** Y, M, S, D □.b, decoration	Bit

Explanation of Instructions

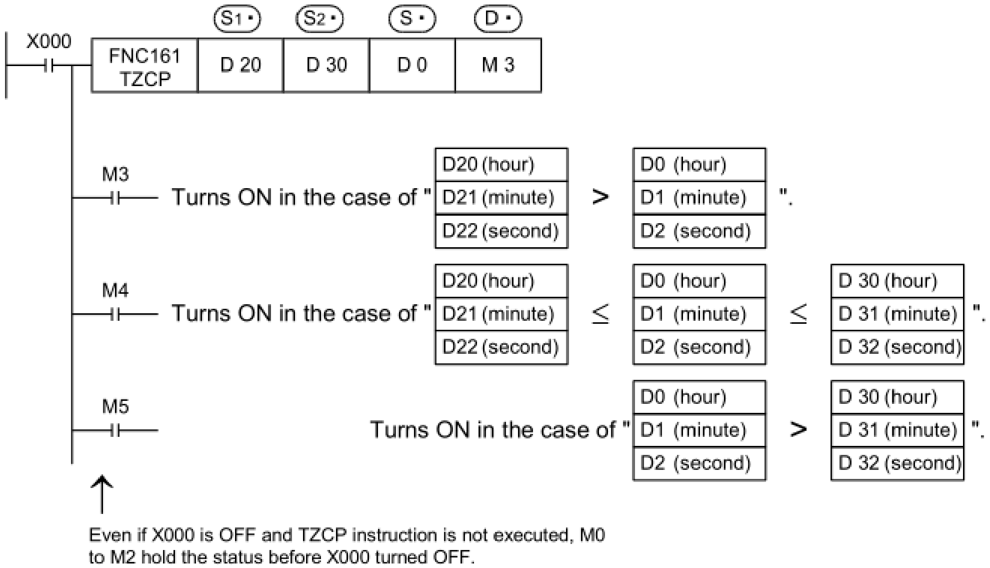
1. 16-bit operation (TZCP, TZCPP)

The lower limit and upper limit comparison time (hour, minute, and second) are compared with the time data (hour, minute, and second) stored in three devices **S.**, **S.** +1, and **S.** +2. Three devices starting from **D.** turn ON or OFF according to the comparison result.



	● When utilizing the time (hour, minute, and second) of the real time clock built in a PLC Read the values of special data registers by TRD (FNC166) instruction, and then specify those word devices as the operands.

● **Program Example**



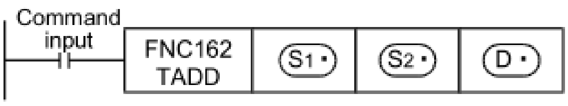
19.3 TADD/ RTC Data Addition

This instruction executes addition of two time data, and stores the addition result to word devices.

Instruction	Operand type	Functions						
FNC162 TADD / **P**	**S1.** **S2.** **D.**	16-bit instruction	Mnemonic	Operation condition		32-bit instruction	Mnemonic	Operation condition
		7 steps	TADD	Continuous operation		—		
			TADDP	Pulse operation				
Operand	**S1.**	Specifies "hour" of the time data (hour, minute, and second) used in addition. (Three devices are occupied.) **Device: T, C, D, R, decoration**				16-bit binary		
	S2.	Specifies "hour" of the time data (hour, minute, and second) used in addition. (Three devices are occupied.) **Device: T, C, D, R, decoration**						
	D.	Stores the addition result (hour, minute, and second) of two time data. (Three devices are occupied.) **Device: T, C, D, R, decoration**						

Explanation of Instructions	**1. 16-bit operation (TADD, TADDP)** The time data (hour, minute, and second) stored in **[S2., S2.+1, S2.+2]** is added to the time data

(hour, minute, and second) stored in **[S1., S1.+1, S1.+2]** , and the addition result (hour, minute, and second) is stored in **[D., D.+1, D.+2]**.



((S1•), (S1•)+1, (S1•)+2)+((S2•), (S2•)+1, (S2•)+2)
→((D•), (D•)+1, (D•)+2)

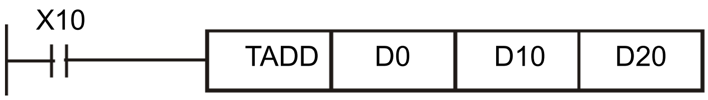
(S1•) (hour)	+	(S2•) (hour)	→	(D•) (hour)	The setting range of "hour" is from 0 to 23.
(S1•) +1(minute)		(S2•) +1(minute)		(D•) +1(minute)	The setting range of "minute" is from 0 to 59.
(S1•) +2(second)		(S2•) +2(second)		(D•) +2(second)	The setting range of "second" is from 0 to 59.

1) When the operation result exceeds 24 hours, the carry flag turns ON, and the value simply acquired by addition subtracted by 24 hours is stored as the operation result.

2) When the operation result becomes "0" (0:0:0), the zero flag turns ON.

- When utilizing the time (hour, minute, and second) of the real time clock built in a PLC, read the values of special data registers using the TRD (FNC166) instruction, and then specify those word devices as the operands.

Program example



◆ When X10=ON, TADD Instruction is executed, and the calendar data hours, minutes, and seconds specified by D0~D2 are added to the calendar data hours, minutes, and seconds specified by D10~D12, and the obtained results are stored in D20~D22. The hours, minutes, and seconds after the summation are obtained in the specified register.

D0 8 (hour)		D10 6 (hour)		D20 14 (hour)
D1 10 (minute)	+	D11 40 (minute)	→	D21 50 (minute)
D2 20 (second)		D12 6 (second)		D22 26 (second)
8:10:20		6:40:6		14:50:26

◆ When the operation result exceeds 24 hours, Carry flag M8022=ON.

18 (hour)		11 (hour)		6 (hour)
40 (minute)	+	30 (minute)	→	10 (minute)
30 (second)		8 (second)		38 (second)
18:40:30		11:30:8		6 :10:38

19.4 TSUB/ RTC Data Subtraction

This instruction executes subtraction of two time data, and stores the subtraction result to word devices.

Instruction	Operand type	Functions					
		16-bit instruction	Mnemonic	Operation condition	32-bit instruction	Mnemonic	Operation condition
FNC163 TSUB **P**	**S1. S2. D.**	7 steps	TSUB	Continuous operation		—	
			TSUBP	Pulse operation			

Operand	**S1.**	Specifies "hour" of the time data (hour, minute, and second) used in subtraction. (Three devices are occupied.) **Device: T, C, D, R, decoration**	16-bit binary
	S2.	Specifies "hour" of the time data (hour, minute, and second) used in subtraction. (Three devices are occupied.) **Device: T, C, D, R, decoration**	
	D.	Stores the subtraction result (hour, minute, and second) of two time data. (Three devices are occupied.) **Device: T, C, D, R, decoration**	

Explanation of Instructions

1. 16-bit operation (TSUB, TSUBP)

The time data (hour, minute, and second) stored in **[S2., S2.+1, S2.+2]** is subtracted from the time data (hour, minute, and second) stored in **[S1., S1.+1, S1.+2]**, and the subtraction result (hour, minute, and second) is stored in **[D., D.+1, D.+2].**

Command input —| |— [FNC163 TSUB | S1• | S2• | D•]

(S1•, S1•+1, S1•+2)−(S2•, S2•+1, S2•+2) → (D•, D•+1, D•+2)

S1		S2		D	
S1• (hour)	−	S2• (hour)	→	D• (hour)	The setting range of "hour" is from 0 to 23.
S1• +1 (minute)		S2• +1 (minute)		D• +1 (minute)	The setting range of "minute" is from 0 to 59.
S1• +2 (second)		S2• +2 (second)		D• +2 (second)	The setting range of "second" is from 0 to 59.

When the operation result is smaller than 0 hour, the borrow flag turns ON, and the value simply acquired by subtraction added by 24 hours is stored as the operation result.
When the operation result becomes "0" (0:0:0), the zero flag turns ON.

- When utilizing the time (hour, minute, and second) of the real time clock built in a PLC, read the values of special data registers using TRD (FNC166) instruction, and then specify those word devices as the operands.

Program example

X10 —| |— [TSUB | D0 | D10 | D20]

- When X10=ON, TSUB Instruction is executed, and the calendar data hours, minutes, and seconds specified by D0~D2 are subtracted from the calendar data hours, minutes, and seconds specified by D10~D12, and the obtained result is stored in the register hours, minutes and seconds specified by D20~D22.

D0 20 (hour)		D10 14 (hour)		D20 5 (hour)
D1 20 (minute)	-	D11 30 (minute)	→	D21 49 (minute)
D2 5 (second)		D12 8 (second)		D22 57 (second)
20:20:5		14:30:8		5:49:57

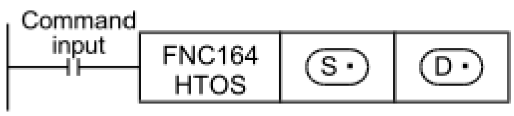
- If the subtraction result is negative, the borrow flag M8021=ON.

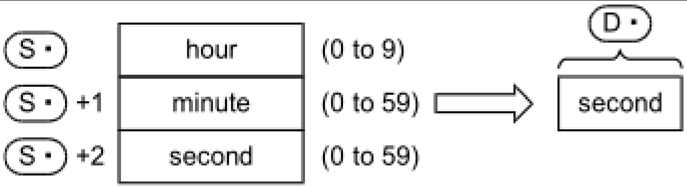
5 (hour)		19 (hour)		10 (hour)
20 (minute)	-	11 (minute)	→	9 (minute)
30 (second)		15 (second)		15 (second)
5:20:30		19:11:15		10:9:15

19.5 HTOS/ Hour to Second Conversion

This instruction converts the time data in units of "hour, minute, and second" into data in units of "second".

<table>
<tr><th>Instruction</th><th>Operand type</th><th colspan="7">Functions</th></tr>
<tr><td rowspan="3">D FNC164 HTOS P</td><td rowspan="3">S.
D.</td><td>16-bit instruction</td><td>Mnemonic</td><td>Operation condition</td><td rowspan="3"></td><td>32-bit instruction</td><td>Mnemonic</td><td>Operation condition</td></tr>
<tr><td rowspan="2">5 steps</td><td>HTOS</td><td>Continuous operation</td><td rowspan="2">9 steps</td><td>DHTOS</td><td>Continuous operation</td></tr>
<tr><td>HTOSP</td><td>Pulse operation</td><td>DHTOSP</td><td>Pulse operation</td></tr>
<tr><td rowspan="2">Operand</td><td>S.</td><td colspan="6">Head device number storing the time data (hour, minute and second) before conversion
Device: KnX, KnY, KnM, KnS, T, C, D, R, decoration</td><td>16-bit binary</td></tr>
<tr><td>D.</td><td colspan="6">Device number storing the time data (second) after conversion
Device: KnY, KnM, KnS, T, C, D, R, decoration</td><td>16- or 32-bit binary</td></tr>
</table>

<table>
<tr><td>Explanation of Instructions</td><td>1. 16-bit operation (HTOS and HTOSP)
The time data (hour, minute, and second) stored in [S., S.+1, S.+2] is converted into data in units of "second", and stored to D..

</td></tr>
</table>

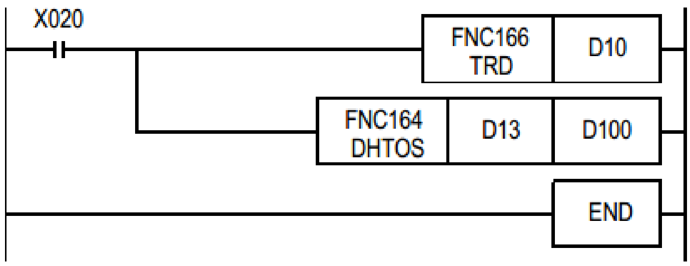


2. 32-bit operation (DHTOS and DHTOSP)

The time data (hour, minute, and second) stored in **[S., S.+1, S.+2]** is converted into data in units of "second", and stored to **[D.+1, D.]**.

- An operation error is caused in the following case; The error flag M8067 turns ON, and the error code is
stored in D8067.
• When the data of **S., S.+1, S.+2** is outside the allowable range (error code: K6706)

Program example



◆ In the program shown above, the time data read from the real time clock built in a PLC is converted into data in units of "second", and stored to D100 and D101 when X020 turns ON.

1) Clock data reading operation by TRD (FNC166) instruction

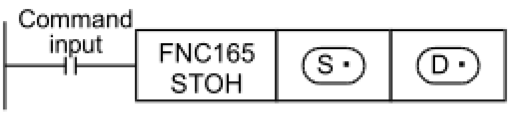
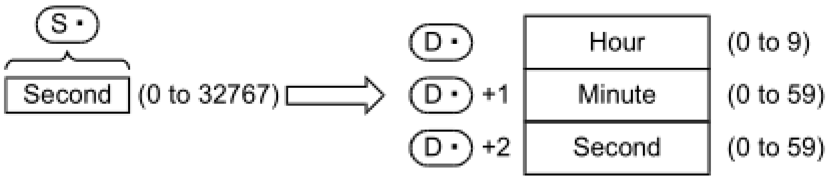
Real time clock	→	D10	2020	Year
		D11	4	Month
		D12	16	Day
		D13	20	Hour
		D14	21	Minute
		D15	23	Second
		D16	4	Day of week

2) Conversion operation into "second" by DHTOS (FNC164) instruction

D13	20	hour		D101, D100
D14	21	minute	→	73,283
D15	23	second		

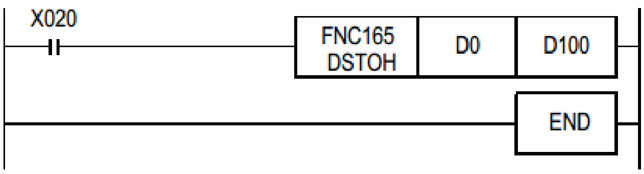
19.6 STOH/ Second to Hour Conversion

This instruction converts the time data in units of "second" into data in units of "hour, minute, and second".

Instruction	Operand type	Functions	
D FNC165 STOH P	S. D.	16-bit instruction: 5 steps; Mnemonic: STOH — Operation condition: Continuous operation; STOHP — Pulse operation 32-bit instruction: 9 steps; Mnemonic: DSTOH — Operation condition: Continuous operation; DSTOHP — Pulse operation	
Operand	S.	Device number storing the time data (second) before conversion **Device:** KnX, KnY, KnM, KnS, T, C, D, R, decoration	16- or 32-bit binary
	D.	Head device number storing the time data (hour, minute and second) after conversion **Device:** KnY, KnM, KnS, T, C, D, R, decoration	16-bit binary
Explanation of Instructions	**1. 16-bit operation (STOH and STOHP)** The time data in units of "second" stored in **S.** is converted into data in units of "hour, minute, and second", and stored to **[D.,D.+1,D.+2]** (hour, minute, and second).   **2. 32-bit operation (DSTOH and DSTOHP)** The time data in units of "second" stored in **[S., S.+1]** is converted into data in units of "hour, minute, and second", and stored to three devices **[D., D.+1, D.+2]** (hour, minute, and second). ● An operation error is caused in the following case; The error flag M8067 turns ON, and the error code is stored in D8067. • When the data of **S.** is outside the allowable range (error code: K6706)		

16-bit instruction	Mnemonic	Operation condition	32-bit instruction	Mnemonic	Operation condition
5 steps	STOH	Continuous operation	9 steps	DSTOH	Continuous operation
	STOHP	Pulse operation		DSTOHP	Pulse operation

Program example



◆ In the program shown above, the time data in units of "second" stored in D0 and D1 is converted into data in units of "hour, minute, and second", and stored to D100, D101, and D102 when X020 turns ON.。

Converting the data in second into the data in hour, minute and second using STOHP instruction (when "40,000 seconds" is specified by D1 and D0)

D1,D0				
		D100	11	hour
40,000	→	D101	6	minute
		D102	40	second

19.7 TRD/ Read RTC data

This instruction reads the clock data of the real time clock built in a PLC.

Instruction	Operand type	Functions					
FNC166 TRD (P)	D.	16-bit instruction	Mnemonic	Operation condition	32-bit instruction	Mnemonic	Operation condition
		3 steps	TRD	Continuous operation		—	
			TRDP	Pulse operation			
Operand	D.	Specifies the head device number storing the clock data. (Seven devices are occupied.) **Device: T, C, D, R, decoration**				16-bit binary	

Explanation of Instructions

1. 16-bit operation (TRD, TRDP)

The clock data stored in D8013 to D8019 of the real time clock built in a PLC is read in the following format,

and stored to **D.~D.+6.**

This instruction reads the real time clock data in a PLC, and transfers it to seven data registers.

	Device	Item	Clock data		Device	Item
Special data register	D8018	Year	0~99 (lower two digits)	→	**D.**	Year
	D8017	Month	1~12	→	**D.+1**	Month
	D8016	Day	1~31	→	**D.+2**	Day
	D8015	Hour	1~23	→	**D.+3**	Hour
	D8014	Minute	1~59	→	**D.+4**	Minute
	D8013	Second	1~59	→	**D.+5**	Second
	D8019	Day of week	0 (Sunday) to 6 (Saturday)	→	**D.+6**	Day of week

19.8 TWR/ Set RTC data

This instruction writes the clock data to the real time clock built in a PLC.

Instruction	Operand type	Functions					
		16-bit instruction	Mnemonic	Operation condition	32-bit instruction	Mnemonic	Operation condition
FNC167 TWR	**S.**	3 steps	TWR	Continuous operation		—	
P			TWRP	Pulse operation			
Operand	**S.**	Specifies the head device number to which the clock data is written. (Seven devices are occupied.) **Device: T, C, D, R, decoration**				16-bit binary	

Explanation of Instructions

16-bit operation (TWR, TWRP)

The clock data stored in **S. to S.+6** is written to D8013 to D8019 for the real time clock built in a PLC.

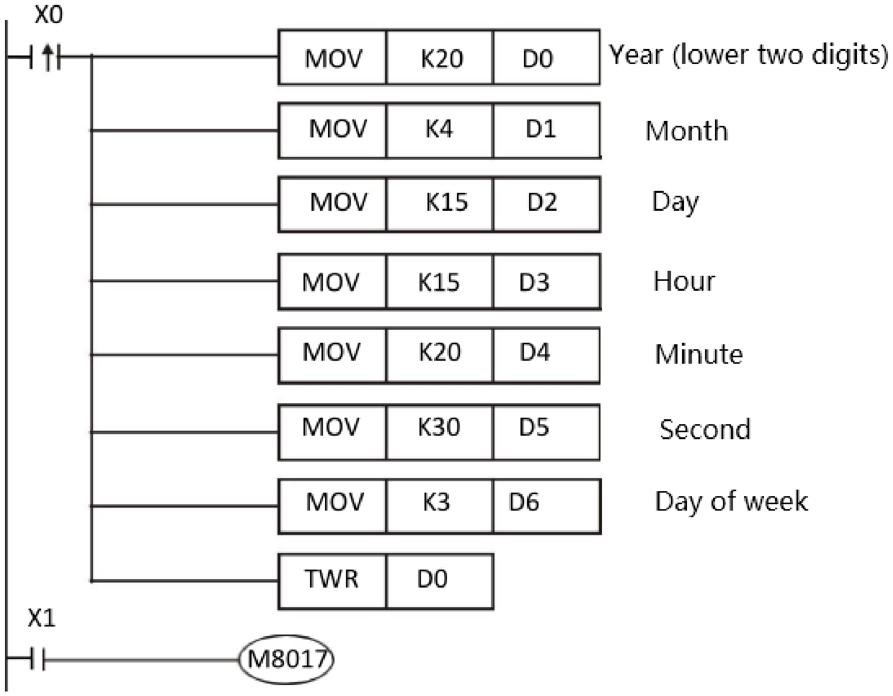
	Device	Item	Clock data		Device	Item	
Time data to be set	**S.**	Year	0~99 (lower two digits)	→	D8018	Year	Special data register
	S.+1	Month	1~12	→	D8017	Month	
	S.+2	Day	1~31	→	D8016	Day	
	S.+3	Hour	1~23	→	D8015	Hour	
	S.+4	Minute	1~59	→	D8014	Minute	
	S.+5	Second	1~59	→	D8013	Second	
	S.+6	Day of week	0 (Sunday) to 6 (Saturday)	→	D8019	Day of week	

- When TWR (FNC167) instruction is executed, the clock data of the real time clock is immediately changed. Accordingly, transfer the clock data several minutes ahead to **S. to S.+6** in advance, and then execute FNC167 instruction when the accurate time has come.

- When setting the clock data (time) using this instruction, it is not necessary to control the special auxiliary relay M8015 (time stop and time setting).

- If a numeric value indicating impossible date/time is set, the clock data is not changed. Set the correct clock data, and then write it.

Program example



- In the program example shown above, the real time clock is set (to 15:20:30 on Wednesday, April 15, 2020).
- The contents of D0~D6 is set as time of the new calendar.
- X0=ON is to change the current time of the calendar clock to the set value.
- Every time X1 is set to ON, the current time can be corrected by ±30 seconds. The correction is that when the second hand is between 1~29, it will be automatically classified as "0" seconds and the minute hand will not change. When it is 30~59, it will also be automatically classified as "0" seconds and the minute hand plus 1 minute.
- D8018 will specify the 4-digit year mode in the second scan and later after the PLC mode is changed to RUN.

— A PLC is normally operating in the 2-digit year mode. When the above instruction is executed and "K2000 (fixed value)" is transferred to D8018 (year) in only one operation cycle after the PLC mode was changed to RUN, the year mode is switched to the 4-digit mode.

— Execute this program every time the PLC mode is changed to RUN. Even if "K2000" is transferred, only the display format is changed to the 4-digit year mode. The current date and time are not affected.。

— In the 4-digit year mode, the set values "80 to 99" correspond to "1980 to 1999", and "00 to 79" correspond to "2000 to 2079".

Examples: "80" indicates 1980. "99" indicates 1999. "00" indicates 2000. "79" indicates 2079.

19.9 HOUR/ Hour Meter

This instruction measures the ON time of the input contact in units of hour.

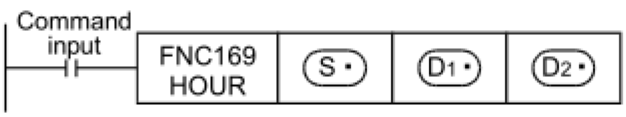
Instruction	Operand type	Functions						
FNC169 HOUR D	**S.** **D1.** **D2.**	16-bit instruction 7 steps	Mnemonic HOUR	Operation condition Continuous operation		32-bit instruction 13 steps	Mnemonic DHOUR	Operation condition Continuous operation
Operand	**S.**	Time after which **D2.** is set to ON (unit: hour)) **Device:** KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, decoration						16- or 32-bit binary
	D1.	Current value (unit: hour) (latched (battery backed) type data register latched (battery backed)) **Device: D, R, decoration**						
	D2.	Head device number to which alarm is output **Device:** Y, M, S, D □.b, decoration						

Explanation of Instructions

1. 16-bit operation (HOUR)

When the accumulated ON time of the command input exceeds the time stored in **S.**, **D2.** is set to ON.

The current value less than one hour is stored in **D1.+1** (unit: second).



S.: Time after which **D2.** is set to ON. Specify a value in units of hour.

D1.: Current value in units of hour

D1.+1: Current value less than one hour (unit: second)

D2.: Alarm output destination. It turns ON when the current value **D1.** exceeds the time specified in **S.**.

2. 32-bit operation (DHOUR)

[S.+1, S.]: Time after which **D2.** is set to ON.

Specify the high-order side in **S.+1**, and the low-order side in **S.**.

[D1.+1, D1.]: Current value in units of hour

The high-order side is stored in **D.+1**, and the low-order side is stored in **D.**.

D1.+2 : Current value less than one hour (unit: second)

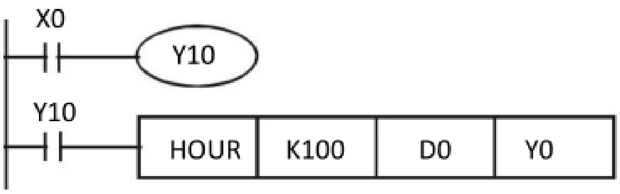
D2.: Alarm output destination

It turns ON when the current value **[D1.+1, D1.]** exceeds the time specified in **S.**.

- Specify a latched (battery backed) type data register as so that the current value data can be

	continuously used even after the PLC turns OFF. ● Even after the alarm output **D2.** turns ON, the measurement is continued. ● When the current value **D1./[D1.+1, D1.]** reaches the maximum value of 16-bit/ 32-not data, the measurement is stopped. For continuing the measurement, clear the current value stored in **D1. to D1.+1/ D1. to D1.+2**. ● Two (16-bit operation) or three (32-bit operation) devices are occupied by **D1.**.

Program example



◆ When X0=ON, Y10 turns on and starts timing. When it reaches 100 hours, Y0 turns on, and D0 will record the current time value (unit: hour) during the measurement, and D1 will record the current time value less than 1 hour during the measurement. 0 ~3599 (unit: second)

20 External Devices

FNC NO.	Instruction	Function	Device		
			3G PLC	2N PLC	MX2N PLC
170	GRY	Decimal to Gray Code Conversio	★		
171	GBIN	Gray Code to Decimal Conversion	★		
172	—				
173	—				
174	—				
175	—				
176	RD3A	Read form Dedicated Analog Block	★ Modbus communication to read slave data (Note: The read function of the original Mitsubishi analog module is not available)		★
177	WR3A	Write to Dedicated Analog Block	★ Modbus communication write data to slave (Note: The original Mitsubishi analog module write function is not available)		★
178	—				
179	—				

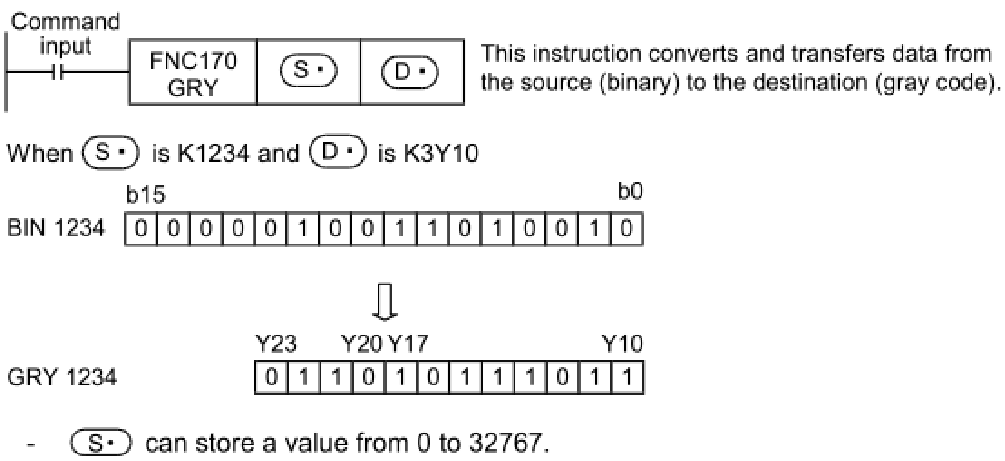
20.1 GRY/ Decimal to Gray Code Conversion

This instruction converts a binary value into a gray code, and transfers it.

Instruction	Operand type	Functions						
D **FNC170 GRY** P	**S.** **D.**	16-bit instruction	Mnemonic	Operation condition		32-bit instruction	Mnemonic	Operation condition
		5 steps	GRY	Continuous operation		9 steps	DGRY	Continuous operation
			GRYP	Pulse operation			DGRYP	Pulse operation
Operand	**S.**	Conversion source data or word device storing conversion source data **Device:** KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, decoration						16- or 32-bit binary
	D.	Word device storing data after conversion **Device:** KnY, KnM, KnS, T, C, D, R, V, Z, decoration						16- or 32-bit binary

Explanation of Instructions

1. 16-bit operation (GRY and GRYP)

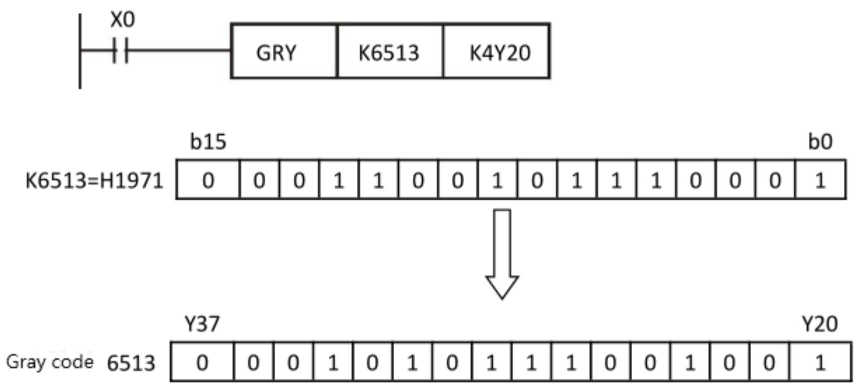


2. 32-bit operation (DGRY and DGRYP)

- A binary value can be converted into a gray code of up to 32 bits.
- S. can store a value from 0 to 2,147,483,647.

- The data conversion speed depends on the scan time of the PLC.

Program example



◆ When X0=ON, convert the constant K6513 to Gray code and store it in K4Y20.

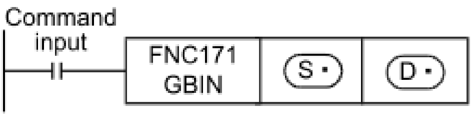
20.2 GBIN/ Gray Code to Decimal Conversion

This instruction converts a gray code into a binary value, and transfers it.

Instruction	Operand type	Functions					
D **FNC171 GBIN** P	**S.** **D.**	16-bit instruction	Mnemonic	Operation condition	32-bit instruction	Mnemonic	Operation condition
		5 steps	GBIN	Continuous operation	9 steps	DGBIN	Continuous operation
			GBINP	Pulse operation		DGBINP	Pulse operation
Operand	**S.**	Conversion source data or word device storing conversion source data **Device:** KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, decoration					16- or 32-bit binary
	D.	Word device storing data after conversion **Device:** KnY, KnM, KnS, T, C, D, R, V, Z, decoration					16- or 32-bit binary

Explanation of Instructions

1. 16-bit operation (GBIN and GBINP)



This instruction converts and transfers data from the source (gray code) to the destination (binary).

When (S·) is K3X000 and (D·) is D10

GRY 1234 — X13 … X10 X7 … X0: 0 1 1 0 1 0 1 1 1 0 1 1

⇩

D10 BIN 1234 — b15 … b0: 0 0 0 0 0 1 0 0 1 1 0 1 0 0 1 0

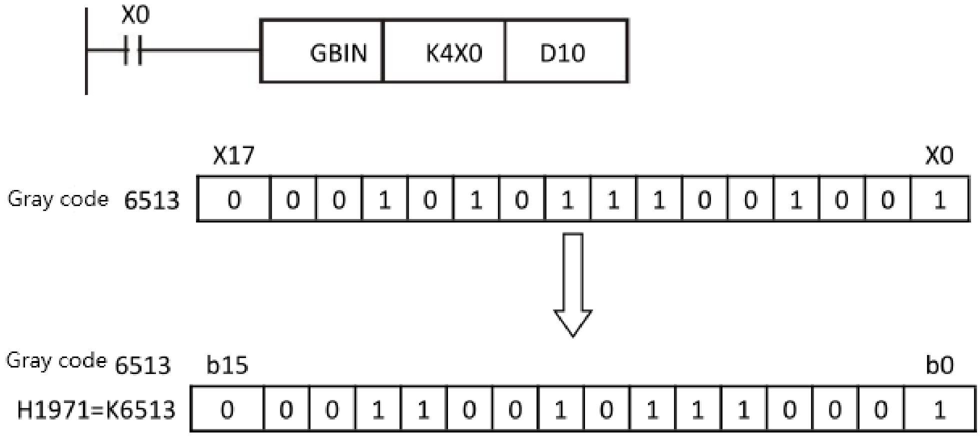
- This instruction can be used for detecting an absolute position by a gray code type encoder.
- (S·) can store a value from 0 to 32,767.

2. 32-bit operation (DGBIN and DGBINP)

- A gray code can be converted into a binary value of up to 32 bits.
- can store a value from 0 to 2,147,483,647.

- When an input relay (X) is specified as **S.**, the response relay will be "Scan time of PLC + Input filter constant".
  The input filter value in X000 to X017 *1 can be converted using the REFF (FNC51) instruction or D8020 (filter adjustment) so that the delay caused by the filter constant is eliminated.

Program example



◆ When X0=ON, convert the gray code of the absolute position encoder connected to the X0~X17 input points into BIN values and store them in D10.

21 Other Instructions

FNC NO.	Instruction	功能	Device		
			3G PLC	2N PLC	MX2N PLC
181	—				
182	COMRD	Read device comment data	★		
183	—				
184	RND	Random Number Generation	★		
185	—				
186	DUTY	Timing pulse generation	★		
187	—				
188	CRC	Cyclic Redundancy Check	★		
189	HCMOV	High speed counter move			

21.1 COMRD/ Read Device Comment Data

This instruction reads the comment data for registered devices written to the PLC by programming software such as GX Developer.

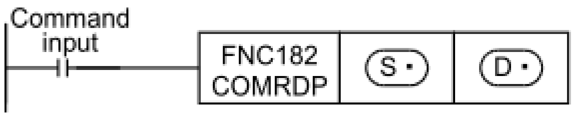
Instruction		Operand type	16-bit instruction	Mnemonic	Operation condition	32-bit instruction	Mnemonic	Operation condition
FNC182 COMRD		S. D.	5 steps	COMRD	Continuous operation		—	
	P			COMRDP	Pulse operation		—	

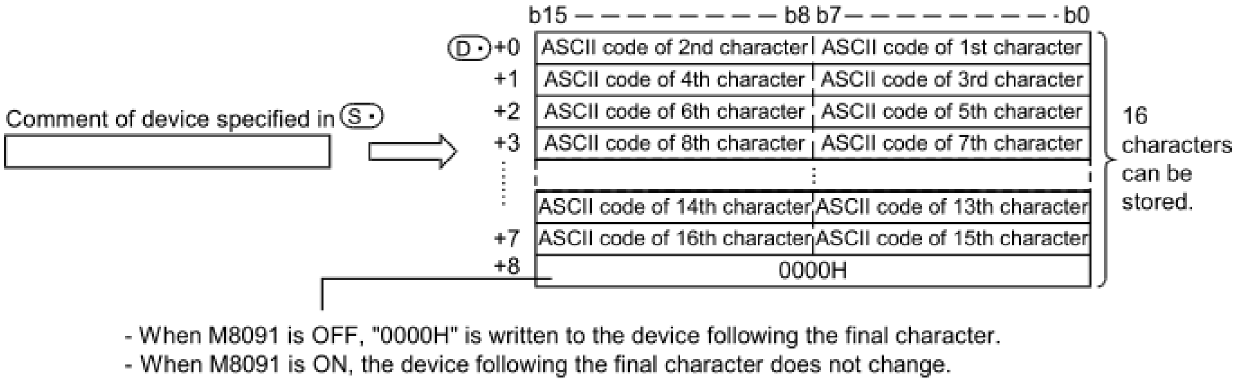
Operand			
Operand	S.	Device number for which comment to be read is registered **Device:** X, Y, M, S, T, C, D, R, decoration	Device name
	D.	Head device number storing read comment **Device: T, C, D, R, decoration**	Character string

Explanation of Instructions

1. 16-bit operation (COMRD and COMRDP)

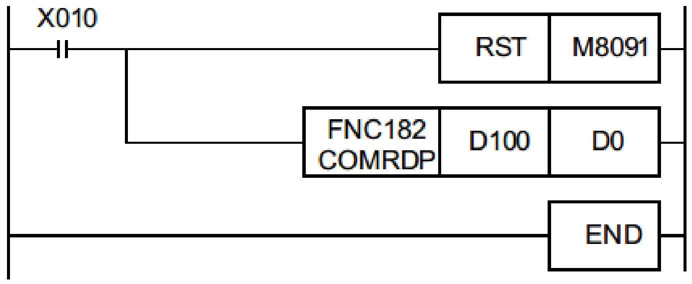
1) The comment registered for device **S.** is read, and stored in ASCII code in **D.** and later.



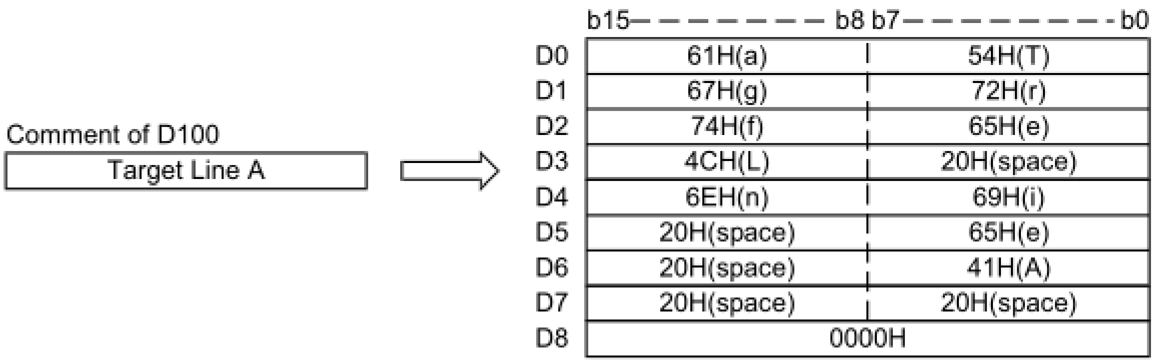


- M8091=ON, **D.+8**=0000H
  M8091=OFF, **D.+8** does not change
- Specify a device number in device **S.** for which a comment is registered in the PLC.
  If a comment is not registered for the device **S.**, "20H" (space) is stored in **D.** and later for the number of characters in the comment (16 half-width characters).
- An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067.
  • When a comment is not registered for the device **S.** (error code: K6706)
  • When the range of points used from **D.** for the comment exceeds the corresponding device range (error code: K6706)

Program example



◆ X10=ON, the comment "Target Line A" registered to D100 is stored in ASCII code in D0 when M8091 is OFF.

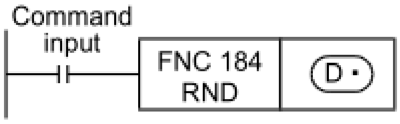


21.2 RND/ Random Number Generation

This instruction generates random numbers.

Instruction	Operand type	Functions						
		16-bit instruction	Mnemonic	Operation condition		32-bit instruction	Mnemonic	Operation condition
FNC184 RND	**D.**	3 steps	RND	Continuous operation			—	
P			RNDP	Pulse operation			—	
Operand	**D.**	Head device number storing a random number **Device:** KnY, KnM, KnS, T, C, D, R, decoration					16-bit binary	

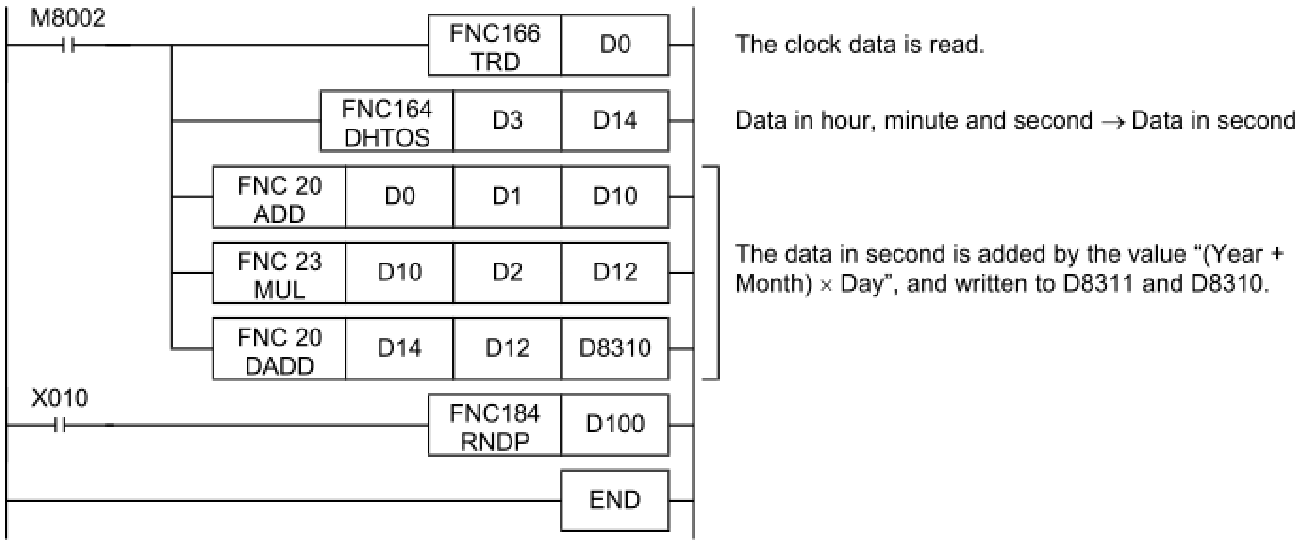
Explanation of Instructions	**1. 16-bit operation (RND and RNDP)** This instruction generates a pseudo-random number within the range from 0 to 32767, and stores it as a random number to **D.**. In the pseudo-random number sequence, the source value of a random number is calculated at every time, and this instruction calculates a pseudo-random number using the source value.

<table>
<tr>
<td></td>
<td>


Pseudo-random number calculation equation:
(D8311, D8310) = (D8311, D8310) * 1 × 1103515245 + 12345.....(1)
D.= “([D8311, D8310]>>16)&<logical product>00007FFFh”
*1. To (D8311, D8310), write a non-negative value (0 to 2,147,483,647) only once when the PLC mode switches from STOP to RUN.
[K1 is written to (D8311, D8310) as the initial value when the power is restored.]
</td>
</tr>
</table>

Program example

◆ In the program example shown below, a random number is stored to D100 every time X010 turns ON. When the PLC mode switches from STOP to RUN, the time data converted into seconds and added by the value “(Year + Month) × Day” is written to D8311 and D8310.



21.3 DUTY/ Timing Pulse Generation

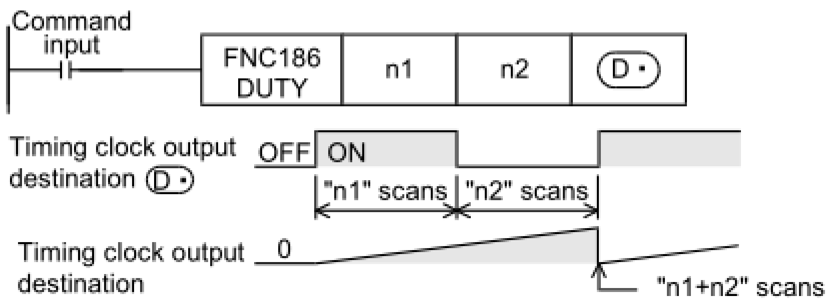
This instruction generates the timing signal whose one cycle corresponds to the specified number of operation cycles.

Instruction	Operand type	Functions						
FNC186 DUTY	**n1 n2 D.**	16-bit instruction	Mnemonic	Operation condition		32-bit instruction	Mnemonic	Operation condition
		7 steps	DUTY	Continuous operation			— —	
Operand	**n1**	Number of scans (operation cycles) to remain ON [n1 > 0] **Device:** T, C, D, R, K, H					16-bit binary	
	n2	Number of scans (operation cycles) to remain OFF [n2 > 0] **Device:** T, C, D, R, K, H					16-bit binary	
	D.	Timing clock output destination **Device:** M(M8330~M8334), decoration					Bit	

Explanation of Instructions

1. 16-bit operation (DUTY)

1) The timing clock output destination D. is set to ON and OFF with the ON duration for "n1" scans and OFF duration for "n2" scans.



2) D8330 ~ D8Specify either one among M8330 to M8334 as the timing clock output destination device **D.**.

The counted number of scans stored in either one among D8330 to D8334 is reset when the counted value reaches "n1+n2" or when the command input (instruction) is set to ON.

Timing clock output destination device **D.**	Scan counting device
M8330	**D8330**
M8331	**D8331**
M8332	**D8332**
M8333	**D8333**
M8334	**D8334**

3) When the command input is set to ON, the operation is started. The timing clock output destination device **D.** is set to ON or OFF by END instruction.

Even if the command input is set to OFF, the operation is not stopped. In the STOP mode, the operation is suspended. When the power of the PLC is turned OFF, the operation is stopped.

4) When "n1" and "n2" are set to "0", the device **D.** is set to the following status:

n1/n2 status	**D.** ON/OFF status
n1 = 0, n2≧0	**D.** Fixed to OFF
n1 = 0, n2≧0	**D.** Fixed to ON

- DUTY (FNC186) instruction can be used up to 5 times (points).

  It is not permitted, however, to use the same timing clock output destination device **D.** for two or more.

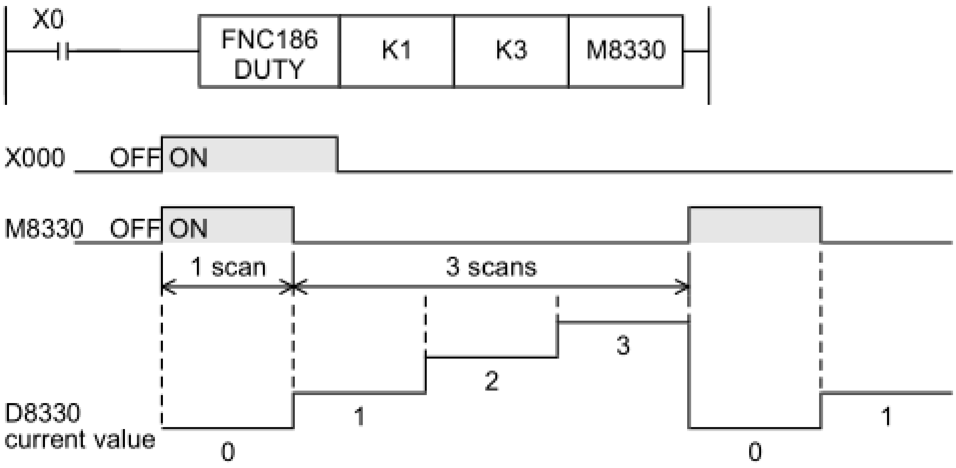
  DUTY (FNC186) instructions.
- An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067.

  • When "n1" and/or "n2" is less than "0" (error code: K6706)

  • When any device other than M8330 to M8334 is set to **D.** (error code: K6705)

Program example

◆ In the program shown below, when X0 is set to ON, M8330 is set to ON for 1 scan and OFF for 3 scans.



21.4 CRC/CRC Cyclic Redundancy Check

This CRC instruction calculates the CRC (cyclic redundancy check) value which is an error check method used in communication.

In addition to CRC value, there are other error check methods such as parity check and sum check. For obtaining the horizontal parity value and sum check value, CCD (FNC 84) instruction is available.

CRC instruction uses [$X^{16}+X^{15}+X^{2}+1$] as a polynomial for generating the CRC value (CRC-16).

Instruction	Operand Type	Function					
		16-bit Instruction	**Mnemonic**	**Operation Condition**	**32-bit Instruction**	**Mnemonic**	**Operation Condition**
FNC188 CRC	**S.** **D.** **n**	7 step	CRC	Continuous Operation		—	
P			CRCP	Pulse (Single) Operation		—	

Operand number	**S.**	Head device number storing data for which the CRC value is generated Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, Modify	BIN16 bit
	D.	Device number storing the generated CRC value **Applicable devices:** KnY, KnM, KnS, T, C, D, R, Modify	
	n	Number of 8-bit (1-byte) data for which the CRC value is generated or the device number storing the number of data **Applicable devices:** D, R, K, H	

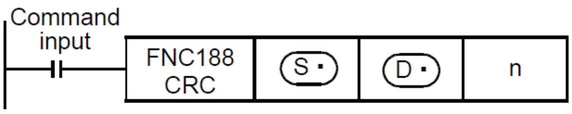
Instruction Explanation

1. 16-bit operation(CRC)

CRC value is generated for “n” 8-bit data (unit: byte) starting from a device specified in **S.**, and stored to **D.**.

The 8-bit conversion mode and 16-bit conversion mode are available in this instruction, and the mode can be switched by turning ON/OFF of M8161. For the operation in each mode, refer to the following pages.

[$X^{16}+X^{15}+X^{2}+1$] is used as a polynomial for generating the CRC value (CRC-16).



2. 16-bit conversion mode [M8161 = OFF]

In this mode, the operation is executed for high-order 8 bits (1 byte) and low-order 8 bits (1 byte) of a device specified in **S.**.

The operation result is stored to one 16-bit device specified in **D.**.

3. 8-bit conversion mode [M8161 = ON]

In this mode, the operation is executed only for low-order 8 bits (low-order 1 byte) of a device specified by **S.**.

With regard to the operation result, low-order 8 bits (1 byte) are stored to a device specified by **D.**, and high-order 8 bits (1 byte) are stored to a device specified by **D.**+1.

- In this instruction, [$X^{16}+X^{15}+X^{2}+1$]" is used as a polynomial for generating the CRC value (CRC-16).

There are many other standard polynomials for generating the CRC value. Note that the CRC value completely differs if an adopted polynomial is different.

Name	**Polynomial**
CRC-12	$X^{12}+X^{11}+X^{3}+X^{2}+X+1$
CRC-16	$X^{16}+X^{15}+X^{2}+1$
CRC-32	$X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^{8}+X^{7}+X^{5}+X^{4}+X^{2}+X+1$
CRC-CCITT	$X^{16}+X^{12}+X^{5}+1$

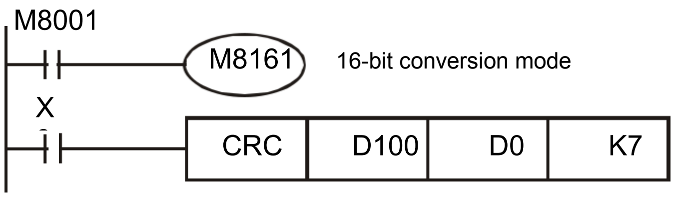
- An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067.

• When any digits other than 4 digits are specified as **S.** or **D.** in digit specification of bit device (error code: K6706)

• **n** is outside the allowable range (1 to 256) (error code: K6706)

• **S.+n-1**、**D.+1** is outside the allowable range (error code: K6706)

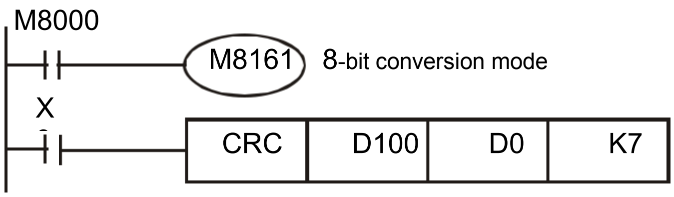
Program Example 1



	Registers	Data content (High/Low byte)
Device storing data for which CRC value is generated	D100	3130H
	D101	3332H
	D102	3534H
	D103	3736H
Device storing generated CRC value	D0	2ACFH

◆ X0=ON, the CRC value of the ASCII code "0123456" stored in D100 to D106 is generated and stored to D0.

Program Example 2



	Registers	Data content (High/Low byte)
Device storing data for which CRC value is generated	D100	30H
	D101	31H
	D102	32H
	D103	33H
	D104	34H
	D105	35H
	D106	36H
Device storing generated CRC value	D0	CFH
	D1	2AH

◆ X0=ON, the CRC value of the ASCII code “0123456” stored in D100 to D106 is generated and stored to D0.

21.5 HCMOV/High Speed Counter Move

This instruction updates the current value of a specified high speed counter or ring counter.

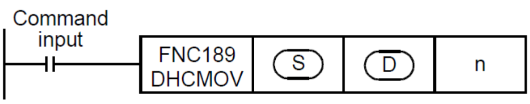
Instruction	Operand Type	Function					
D FNC189 HCMOV	S. D. n	16-bit Instruction	Mnemonic	Operation Condition	32-bitInstruction	Mnemonic	Operation Condition
					13 steps	DHCMOV	Continuous Operation

Operand number	Operand	Description	Data type
Operand number	S.	Device number of high speed counter or ring counter*1 handled as transfer source **Applicable devices:** C, D	BIN32 bit
	D.	Device number handled as transfer destination **Applicable devices:** D, R	BIN32 bit
	n	Specification to clear the current value of high speed counter or ring counter*1 (transfer source) after transfer [clear (K1), no clear (K0)] **Applicable devices:** K, H	BIN16 bit

Remark:*1: Can only specify high speed counter(C235 ~ C255),Ring counter(D8099, D8398)

Instruction Explanation

1. 32-bit operation(DHCMOV)



1) The current value of a high speed counter or ring counter specified in **S.** is transferred to **[D.+1, D.]**.

Device S.		[D.+1, D.] after instruction is executed
High speed counter	C235 ~ C255	Current value of high speed counter **S.→[D.+1, D.]**
Ring counter	D8099	D8099→**D.** "0" is stored in **D.+1**"
	D8398	Current value of [D8399, D8398]→**[D.+1, D.]**

2) After transfer, the current value of the high speed counter or ring counter is processed as shown in the table below depending on the set value of "n":

N **set value**	**Operation**
K0(H0)	Does not clear the current value (no processing).
K1(H1)	Clears the current value to "0".

2. High speed counter current value update timing and the effect of DHCMOV instruction

1) High speed counter current value update timing

When a pulse is input to an input terminal for a high speed counter (C235 to C255), the high speed counter executes up-counting or down-counting. If the current value of a high speed counter is handled in an applied instruction such as the normal MOV instruction, the current value is updated at the timing shown in the table below. As a result, it is affected by the program scan time.

2) Effect of DHCMOV instruction

• By using both input interrupt and DHCMOV instruction, the current value of a high speed counter can be received at the rising edge or falling edge of an external input (at reception of input interrupt).

• When DHCMOV instruction is used just before a comparison instruction (CMP, ZCP or comparison contact instruction), the latest value of a high speed counter is used in comparison.The following points must be kept in mind when using the DHCMOV command.

- When the current value of a high speed counter is compared using CMP, ZCP or comparison contact instruction (not using a designated high speed counter comparison instruction), a hardware counter does not change into a software counter.
- When the number of high speed software counter comparison instructions is reduced, the total frequency limitation is decreased.
- When it is necessary to execute comparison and change an output contact (Y) as soon as the current value of a high speed counter changes, use a designated high speed counter comparison instruction (HSCS, HSCR or HSZ).
- DHCMOV instruction can be used as many times as necessary.

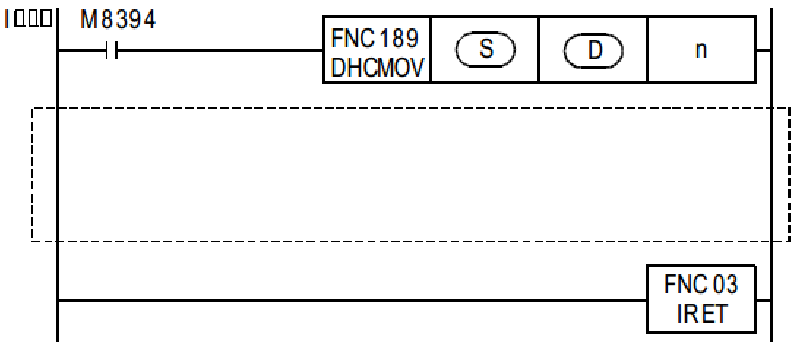
3. Cautions

When programming DHCMOV instruction in an input interrupt program, the following points should be observed.

1) Program EI (FNC 04) and FEND (FNC 06) instructions in the main program. They are necessary to execute an input interrupt program.

2) When programming DHCMOV instruction in the 1st line in an input interrupt program, make sure to use the pattern program shown below.

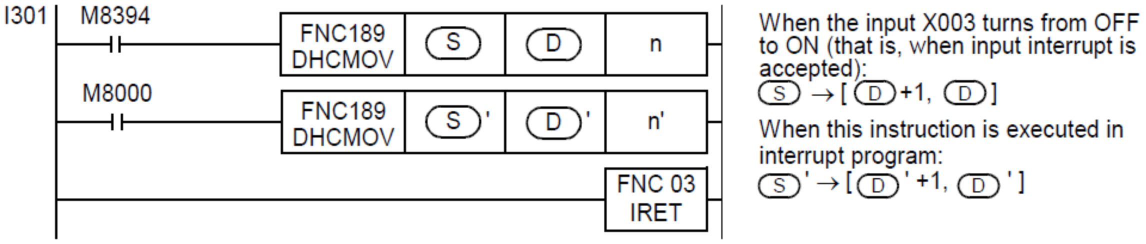
Make sure to use the command contact M8394.



3) If two or more DHCMOV instructions are used in one input interrupt program, only the first instruction (just after the interrupt pointer) is executed when the interrupt is generated.

The rest of the interrupt, including additional DHCMOV instructions, is executed according to normal interrupt processing.

Do not use M8394 as the command contact for the DHCMOV instructions following the first.



4) It is not permitted to use DHCMOV instruction for the same counter in two or more input interrupt programs.

5) While input interrupts are disabled by the interrupt disable flags (shown in the table below), DHCMOV instructions are not executed when they are placed inside a corresponding interrupt.

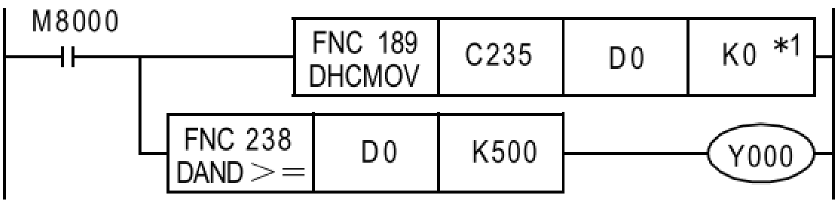
Input	Input interrupt pointer		Interrupt disable flag
	Rising edge interrupt	Falling edge interrupt	
X000	I001	I000	M8050
X001	I101	I100	M8051
X002	I201	I200	M8052
X003	I301	I300	M8053
X004	I401	I400	M8054
X005	I501	I500	M8055

6) If an input interrupt is generated while input interrupts are disabled by something other than the interrupt disable flags M8050 to M8055 (after execution of DI instruction and before execution of EI instruction),DHCMOV instruction is immediately executed, but execution of the interrupt program is held. The interrupt program will be executed after EI instruction is executed and interrupts are enabled.

- An operation error occurs in the following case; The error flag M8067 turns ON, and the error code is stored in D8067.

  • When a device specified in **S.**、 **[D.+1, D.]** is outside the allowable range (error code: K6705)

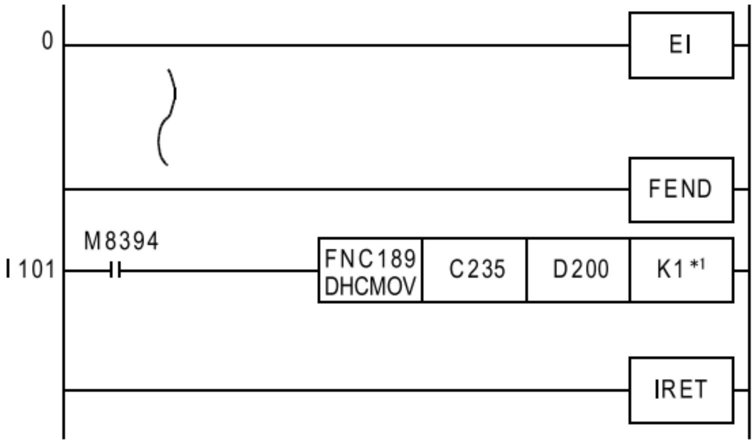
Program Example 1



*1. K0: The current value of the high speed counter is not cleared when DHCMOV instruction is executed.
K1: The current value of the high speed counter is cleared when DHCMOV instruction is executed.

◆ In the program example below, the current value of the high speed counter C235 is compared in each operation cycle, and then the output Y000 is set to ON if the current value is "K500" or more (when the current value of C235 is not cleared).

Program Example 2



*1. K0: K0: The current value of the high speed counter is not cleared when DHCMOV instruction is executed.
K1: The current value of the high speed counter is cleared when DHCMOV instruction is executed.

◆ In the program example shown below, the current value of C235 is transferred to D201 and D200, and the current value of C235 is cleared when X001 turns from OFF to ON.

22 Block Data Operation – FNC190 to FNC199

FNC NO.	Instruction	Function	Supported PLC series		
			3G PLC	2N PLC	MX2N PLC
190	—				
191	—				
192	BK+	Block Data Addition	★		
193	BK-	Block Data Subtraction	★		
194	BKCMP=	Block Data Compare **S1.**=**S2.**	★		
195	BKCMP>	Block Data Compare **S1.** > **S2.**	★		
196	BKCMP<	Block Data Compare **S1.** < **S2.**	★		
197	BKCMP<>	Block Data Compare **S1.**≠**S2.**	★		
198	BKCMP<=	Block Data Compare **S1.**≤**S2.**	★		
199	BKCMP>=	Block Data Compare **S1.**≥**S2.**	★		

22.1 BK+/ Block Data Addition

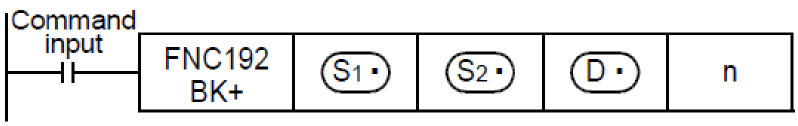
This instruction adds binary block data.

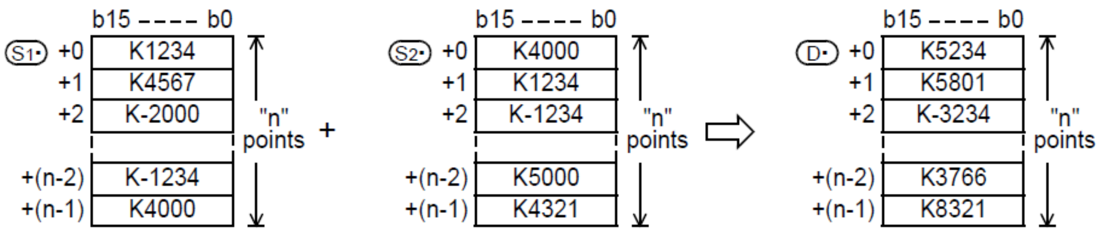
<table>
<tr><th>Instruction</th><th>Operand Type</th><th colspan="7">Function</th></tr>
<tr><td rowspan="4">D FNC192 BK+ P</td><td rowspan="4">S1.
S2.
D.
n</td><td>16-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td><td rowspan="4"></td><td>32-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td></tr>
<tr><td>9 steps</td><td>BK+</td><td>Continuous Operation</td><td>17 steps</td><td>DBK+</td><td>Continuous Operation</td></tr>
<tr><td></td><td>BK+P</td><td>Pulse (Single) Operation</td><td></td><td>DBK+P</td><td>Pulse (Single) Operation</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td rowspan="4">Operand number</td><td>S1.</td><td colspan="6">Head device number storing addition data
Applicable devices: T, C, D, R, Modify</td><td rowspan="4">BIN16/32 Bit</td></tr>
<tr><td>S2.</td><td colspan="6">Added constant or head device number storing addition data
Applicable devices: T, C, D, R, K, H, Modify</td></tr>
<tr><td>D.</td><td colspan="6">Head device number storing operation result
Applicable devices: T, C, D, R, Modify</td></tr>
<tr><td>n</td><td colspan="6">Number of data
Applicable devices: D, R, K, H</td></tr>
</table>

Instruction Explanation

1. 16-bit operation(BK+,BK+P)

1) “n" 16-bit binary data starting from **S2.** are added to "n" 16-bit binary data starting from **S1.** , and the operation result is stored in "n" points starting from **D.**.

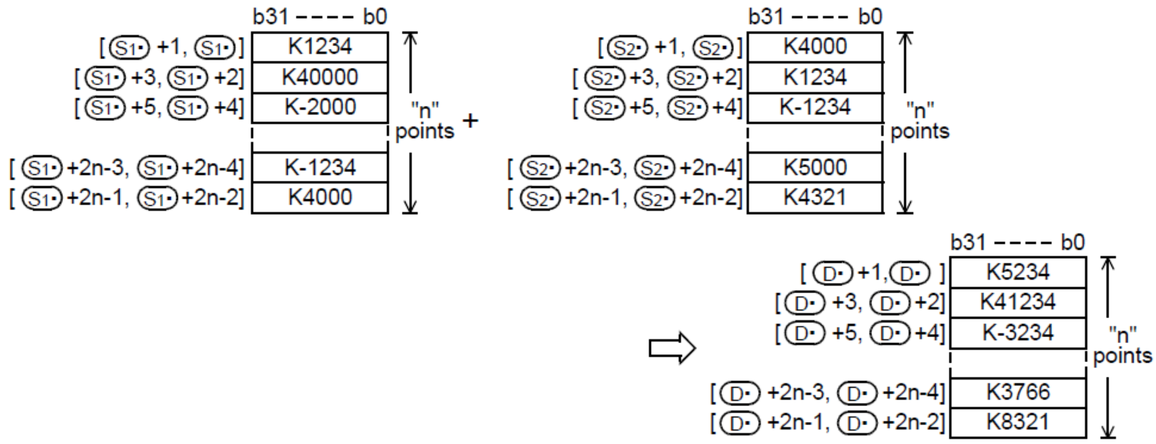




2) A (16-bit) constant from -32768 to +32767 can be directly specified in **S2.**.

2. 32-bit operation(DBK+,DBK+P)

1) "2n" 32-bit binary data starting from **[S2.+1, S2.]** are added to "2n" 32-bit binary data starting from **[S1.+1, S1.]**, and the operation result is stored in "2n" points starting from **[D.+1, D.]**.



2) A (32-bit) constant from -2,147,483,648 to +2,147,483,647 can be directly specified in **[S2.+1, S2.]**.

- When underflow or overflow occurs in the operation result, the following processing is executed. At this time, the carry flag does not turn ON.

  - 16-bit operation

    K32767(H7FFF) + K2(H0002) → K-32767(H8001)
    K-32768(H8000) + K-2(HFFFE) → K32766(H7FFE)

  - 32-bit operation

    K2,147,483,647(H7FFFFFFF) + K2(H00000002) → K-2,147,483,647(H80000001)
    K-2,147,483,648(H80000000) + K-2(HFFFFFFFE) → K2,147,483,646(H7FFFFFFE)

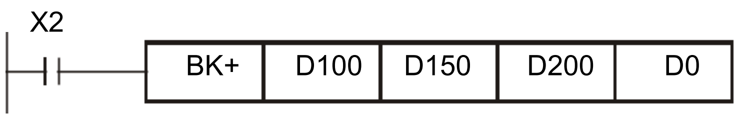
- An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067.

  •)When "n" ("2n" in 32-bit operation) devices starting from **S1.**,**S2.**,**D.** ,exceed the corresponding device range (error code: K6706)

  • When "n" ("2n" in 32-bit operation) devices starting from **S1.** overlap "n" ("2n" in 32-bit operation) devices starting from **D.** (error code: K6706)

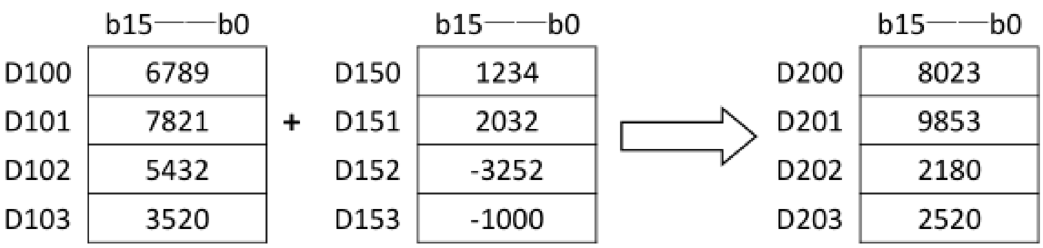
  • When "n" ("2n" in 32-bit operation) devices starting from **S2.** overlap "n" ("2n" in 32-bit operation) devices starting from **D.** (error code: K6706)

Program Example 2



◆ In the program shown below, the specified number of data stored in D150 to D0 are added to the specified number of data stored in D100 to D0 when X020 is set to ON, and the operation result is stored in D200 and later.

D0=4



22.2 BK-/Block Data Subtraction

This instruction subtracts binary block data.

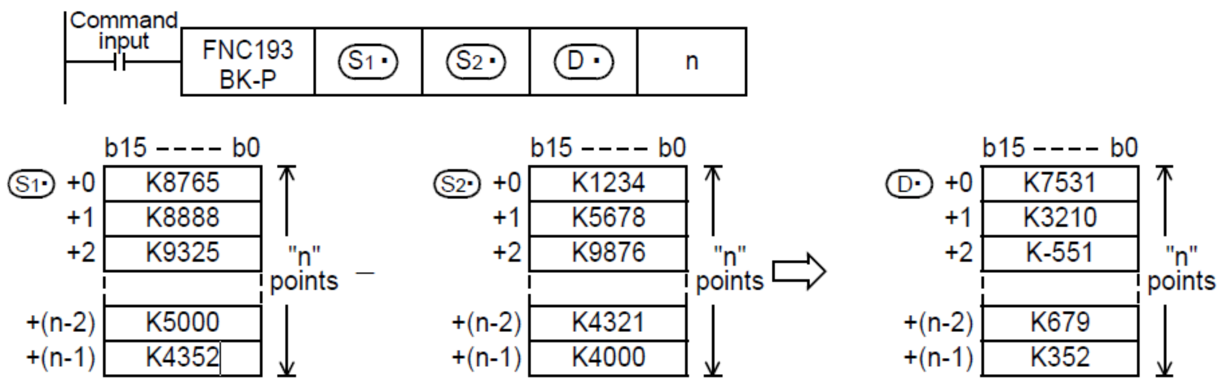
Instruction	Operand Type	Function					
D FNC193 BK- P	S1. S2. D. n	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
		9 steps	BK-	Continuous Operation	17 steps	DBK-	Continuous Operation
			BK-P	Pulse (Single) Operation		DBK-P	Pulse (Single) Operation

Operand number			
Operand number	**S1.**	Head device number storing subtraction data **Applicable devices:** T, C, D, R, Modify	BIN16/32 Bit
	S2.	Subtracted constant or head device number storing subtraction data **Applicable devices:** T, C, D, R, K, H, Modify	
	D.	Head device number storing operation result **Applicable devices:** T, C, D, R, Modify	
	n	Number of data **Applicable devices:** D, R, K, H	

Instruction Explanation

1. 16-bit operation(BK-,BK-P)

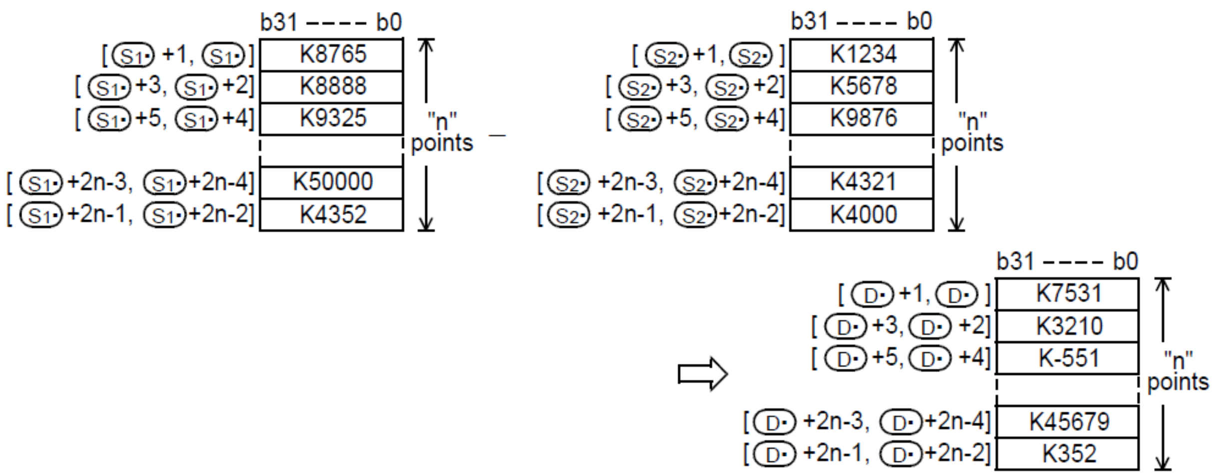
1) "n" 16-bit binary data starting from **S2.** are subtracted from "n" 16-bit binary data starting from **S1.**,and the operation result is stored in "n" points starting from **D.**.



2) A (16-bit) constant from -32768 to +32767 can be directly specified in **S2.**

2. 32-bit operation(DBK-,DBK-P)

1) "2n" 32-bit binary data starting from**[S2.+1, S2.]** are subtracted from "2n" 32-bit binary data starting from **[S1.+1, S1.]**, and the operation result is stored in "2n" points starting from**[D.+1, D.]**.



2) A (32-bit) constant from -2,147,483,648 to +2,147,483,647 can be directly specified in **[S2.+1, S2.]**

- When underflow or overflow occurs in the operation result, the following processing is executed. At this time, the carry flag does not turn ON.

  - 16-bit operation

    K-32768(H8000) - K2(H0002) → K32766(H7FFE)
    K32767(H7FFF) - K-2(HFFFE) → K-32767(H8001)

  - 32-bit operation

    K-2,147,483,648(H80000000) - K2(H00000002) → K2,147,483,646(H7FFFFFFE)
    K2,147,483,647(H7FFFFFFF) - K-2(HFFFFFFFE) → K-2,147,483,647(H80000001)

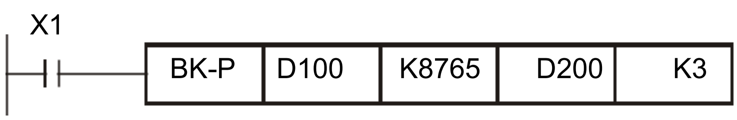
- An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067.

  • When "n" ("2n" in 32-bit operation) devices starting from **S1.**,**S2.**,**D.** exceed the corresponding device range (error code: K6706)

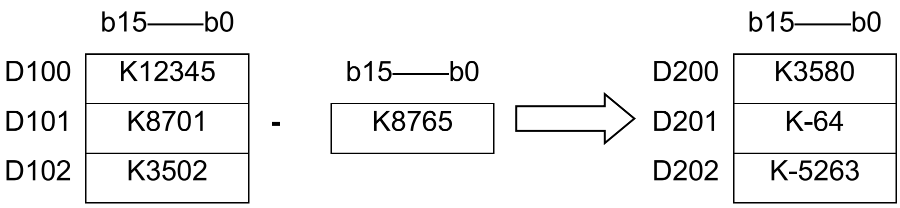
  • When "n" ("2n" in 32-bit operation) devices starting from **S1.** overlap "n" ("2n" in 32-bit operation) devices starting from **D.** (error code: K6706)

  • When "n" ("2n" in 32-bit operation) devices starting from **S2.** overlap "n" ("2n" in 32-bit operation) devices starting from **D.** (error code: K6706)

Program Example



◆ In the program shown below, the constant "8765" is subtracted from the data stored in D100 to D102 when X010 is set to ON, and the operation result is stored in D200 and later.



22.3 BKCMP=,>,<,<>,<=,>=/Block Data Compare

These instructions compare block data in the comparison condition set in each instruction.

	Instruction		Operand Type	Function					
	FNC194 BKCMP=		S1. S2. D. n	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
				9 steps	BKCMP=	Continuous Operation	17 steps	DBKCMP=	Continuous Operation
D		P			BKCMP=P	Pulse (Single) Operation		DBKCMP=P	Pulse (Single) Operation
	FNC195 BKCMP>		S1. S2. D. n	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
				9 steps	BKCMP>	Continuous Operation	17 steps	DBKCMP>	Continuous Operation
D		P			BKCMP>P	Pulse (Single) Operation		DBKCMP>P	Pulse (Single) Operation
	FNC196 BKCMP<		S1. S2. D. n	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
				9 steps	BKCMP<	Continuous Operation	17 steps	DBKCMP<	Continuous Operation
D		P			BKCMP<P	Pulse (Single) Operation		DBKCMP<P	Pulse (Single) Operation
	FNC197 BKCMP≠		S1. S2. D. n	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
				9 steps	BKCMP≠	Continuous Operation	17 steps	DBKCMP≠	Continuous Operation
D		P			BKCMP≠P	Pulse (Single) Operation		DBKCMP≠P	Pulse (Single) Operation
	FNC198 BKCMP≤		S1. S2. D. n	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
				9 steps	BKCMP≤	Continuous Operation	17 steps	DBKCMP≤	Continuous Operation

D		P			BKCMP≤P	Pulse (Single) Operation		DBKCMP≤P	Pulse (Single) Operation

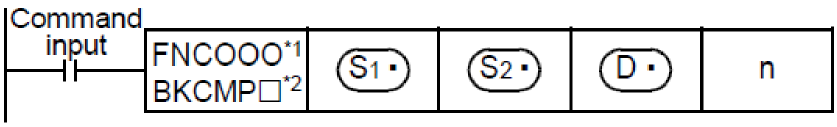
			16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
FNC199 BKCMP≥	D	P	S1. S2. D. n	9 steps	BKCMP≥	Continuous Operation	17 steps	DBKCMP≥	Continuous Operation
					BKCMP≥P	Pulse (Single) Operation		DBKCMP≥P	Pulse (Single) Operation

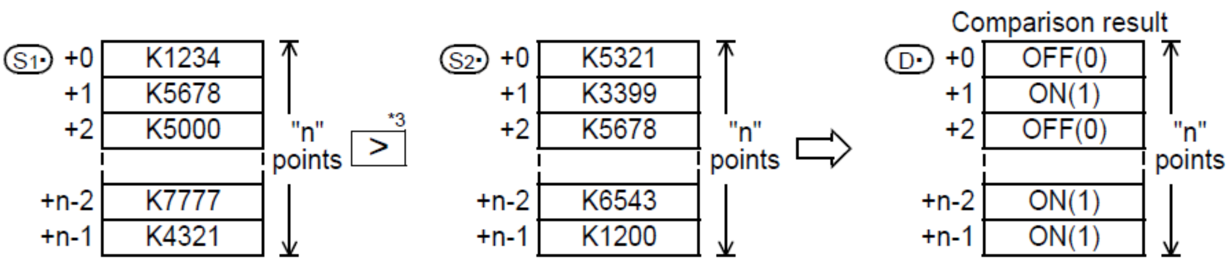
Operand number	S1.	Comparison value of device number storing comparison value **Applicable devices:** T, C, D, R, K, H, Modify	BIN16/32 bit
	S2.	Head device number storing comparison source data **Applicable devices:** T, C, D, R, Modify	
	D.	Head device number storing comparison result **Applicable devices:** Y, M, S, D □.b, Modify	
	n	Number of compared data **Applicable devices:** D, R, K, H	

Instruction Explanation

1. 16-bit operation(BKCMP=,>,<,<>,<=,>=/BKCMP=P,>P,<P,<>P,<=P,>=P)

1) "n" 16-bit binary data starting from **S1.** are compared with "n" 16-bit binary data starting from **S2.**, and the comparison result is stored in "n" points starting from **D.**.





*1.The number out of 194 to 199 is put in "OOO".

*2.The symbol out of "=", ">", "<", "<>", "<=", or ">=" corresponding to the FNC No. is put in □

*3.An operation example of BKCMP >

2) A constant can be directly specified in **S1.**

3) The table below shows the comparison result in each instruction

Instruction	Comparison result ON (1) condition	Comparison result OFF (0) condition
BKCMP=	**S1.** = **S2.**	**S1.≠S2.**
BKCMP>	**S1.** > **S2.**	**S1.≤S2.**
BKCMP<	**S1.** < **S2.**	**S1.≥S2.**
BKCMP<>	**S1.≠S2.**	**S1.** = **S2.**
BKCMP<=	**S1.≤S2.**	**S1.** > **S2.**
BKCMP>=	**S1.≥S2.**	**S1.** < **S2.**

4) When the comparison result is ON (1) in all of "n" points starting from **D.**, M8090 (block comparison signal) turns ON.

2. 32-bit operation((DBKCMP=,>,<,<>,<=,>=/DBKCMP=P,>P,<P,<>P,<=P,>=P)

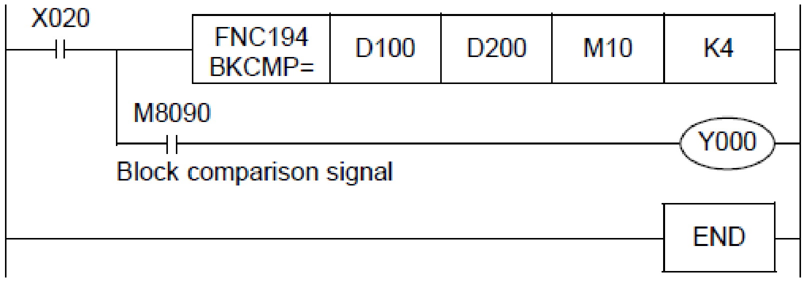
1) "n" 32-bit binary data starting from **[S1.+1, S1.]** are compared with "n" 32-bit binary data starting from **[S2.+1, S2.]**, and the comparison result is stored in "n" points starting from **D.**

2) A constant can be directly specified in **[S1.+1, S1.].**

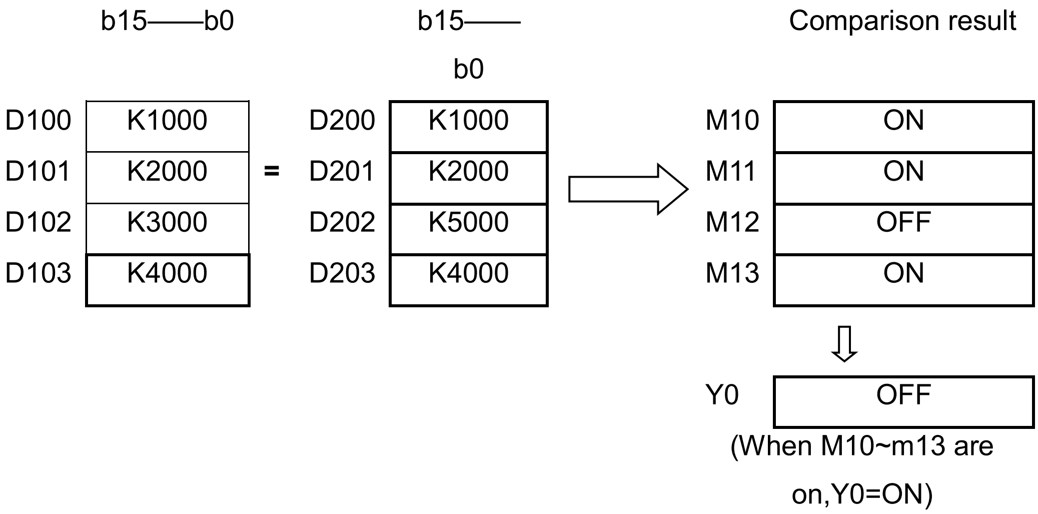
3. When the comparison result is ON (1) in all "block compare signal, M8090, data block instruction" turns ON.

- When using 32-bit counters (including 32-bit high speed counters)
  For comparing 32-bit counters and 32-bit high speed counters (C200 to C255), make sure to use an instruction for 32-bit operation (DBKCMP=, DBKCMP>, DBKCMP<, DBKCMP<>, DBKCMP<=, or DBKCMP>=).
  If an instruction for 16-bit operation (BKCMP=, BKCMP>, BKCMP<, BKCMP<>, BKCMP<=, or BKCMP>=) is used, an operation error is caused (error code: K6705).
- An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067.
  • When the range of "n" ("2n" in 32-bit operation) points starting from **S1.**,**S2.** exceeds the corresponding device range (error code: K6706)
  • When the range of "n" points starting from **D.** exceeds the corresponding device range (error code: K6706)
  • When data registers starting from **D.** specified as "D□.b" overlap "n" ("2n" in 32-bit operation) points starting from **S1.** (error code: K6706)
  • When data registers starting from **D.** specified as "D□.b" overlap "n" ("2n" in 32-bit operation) points starting from **S2.** (error code: K6706)
  • When a 32-bit counter (C200 to C255) is specified in **S1.**,**S2.** in 16-bit operation (error code: K6705)
  For comparing 32-bit counters, make sure to use an instruction for 32-bit operation (DBKCMP=,DBKCMP>, DBKCMP<, DBKCMP<>, DBKCMP<=, or DBKCMP>=).

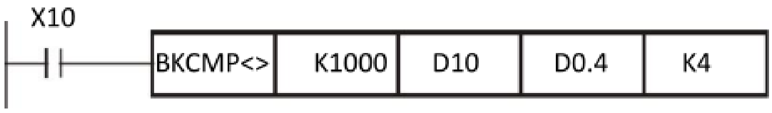
Program Example 1



◆ In the program shown below, four 16-bit binary data starting from D100 are compared with four 16-bit binary data starting from D200 by BKCMP= (FNC194) instruction when X020 is set to ON, and the comparison result is stored in four points starting from M10.
When the comparison result is "ON (1)" in all of the four points starting from M10, Y000 is set to ON.



Program Example 2



◆ In the program shown below, the constant K1000 is compared with four data starting from D10 when X010 is set to ON, and the comparison result is stored in b4 to b7 of D0.

23 Character String Control – FNC200 to FNC209

FNC NO.	Instruction	Function	Supported PLC series		
			3G PLC	2N PLC	MX2N PLC
200	STR	BIN to Character String Conversion	★		
201	VAL	Character String to BIN Conversion	★		
202	$+	Link Character Strings	★		
203	LEN	Character String Length Detection	★		
204	RIGHT	Extracting Character String Data from the Right	★		
205	LEFT	Extracting Character String Data from the Left	★		
206	MIDR	Random Selection of Character Strings	★		
207	MIDW	Random Replacement of Character Strings	★		
208	INSTR	Character string search	★		
209	$MOV	Character String Transfer	★		

23.1 STR/BIN to Character String Conversion

This instruction converts binary data into character strings (ASCII codes).

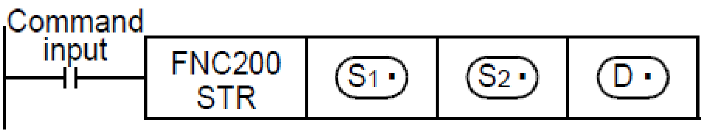
On the other hand, the ESTR (FNC116) instruction converts floating point data into character strings.

<table>
<tr><th>Instruction</th><th>Operand Type</th><th colspan="8">Function</th></tr>
<tr><td rowspan="3">D FNC200 STR P</td><td rowspan="3">S1.
S2.
D.</td><td>16-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td><td></td><td>32-bit Instruction</td><td>Mnemonic</td><td colspan="2">Operation Condition</td></tr>
<tr><td rowspan="2">7steps</td><td>STR</td><td>Continuous Operation</td><td></td><td rowspan="2">17steps</td><td>DSTR</td><td colspan="2">Continuous Operation</td></tr>
<tr><td>STRP</td><td>Pulse (Single) Operation</td><td></td><td>DSTRP</td><td colspan="2">Pulse (Single) Operation</td></tr>
<tr><td rowspan="3">Operand number</td><td>S1.</td><td colspan="7">Head device number storing the number of digits of a numeric value to be converted
Applicable devices: T, C, D, R, Modify</td><td>BIN16-bit</td></tr>
<tr><td>S2.</td><td colspan="7">Device number storing binary data to be converted
Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, Modify</td><td>BIN16/32-bit</td></tr>
<tr><td>D.</td><td colspan="7">Head device number storing converted character string
Applicable devices: T, C, D, R, Modify</td><td>Character string</td></tr>
</table>

Instruction Explanation

1. 16-bit operation(STR,STRP)

1) All digits (specified by **S1.**) of 16-bit binary data stored in are converted into ASCII codes while the decimal point is added to the position specified by the device storing the number of digits of the decimal part **S1.+1**, and stored in **D.** and later.



2) Set the number of all digits **S1.** in the range from 2 to 8.

3) Set the number of digits of the decimal part **S1.+1** in the range from 0 to 5.

Make sure to satisfy "Number of digits of decimal part <= (Number of all digits -3)".

4) 16-bit binary data to be converted stored in **S2.** should be within the range from -32768 to +32767.

5) Converted character string data is stored in **D.** and later as shown below.

- As the sign, "space" (20H) is stored when the 16-bit binary data stored in **S2.** is positive, and "-"(2DH) is stored when the 16-bit binary data stored in **S2.** is negative.
- When the number of digits of the decimal part **S1.+1** is set to any value other than "0", the decimal point ".(2EH)" is automatically added in "number of digits of decimal part **S1.+1**"the digit. When the number of digits of the decimal part +1 is set to "0", the decimal point is not

added.

- When the number of digits of the decimal part **S1.+1** is larger than the number of digits of 16-bit binary data stored in **S2.**, "0" (30H) is automatically added, and the data is shifted to the right end during conversion.
- When the number of all digits stored in **S1.** excluding the sign and decimal point is larger than the number of digits of 16-bit binary data stored in **S2.**, "space" (20H) is stored in each digit between the sign and the numeric value.

When the number of all digits stored in **S1.** excluding the sign and decimal point is smaller than the number of digits of 16-bit binary data stored in **S2.**, an error is caused.

- "00H" indicating the end of a character string is automatically stored at the end of a converted character string. When the number of all digits is even, "0000H" is stored in the device after the last character. When the number of all digits is odd, "00H" is stored in the high-order byte (8 bits) of the device storing the final character.

2. 32-bit operation(DSTR、DSTRP)

1) All digits (specified by **S1.**) of 32-bit binary data stored in **[S2.+1, S2.]** are converted into ASCII codes while the decimal point is added to the position specified by the device storing the number of digits of the decimal part (**S1.+1**), and stored in **D.** and later.

2) Set the number of all digits **S1.** in the range from 2 to 13.

3) Set the number of digits of the decimal part **S1.+1** in the range from 0 to 10.

Make sure to satisfy "Number of digits of decimal part <= (Number of all digits -3)".

4) 32-bit binary data (BIN)**S2.** to be converted ,should be within the range from -2,147,483,648 to +2,147,483,647.

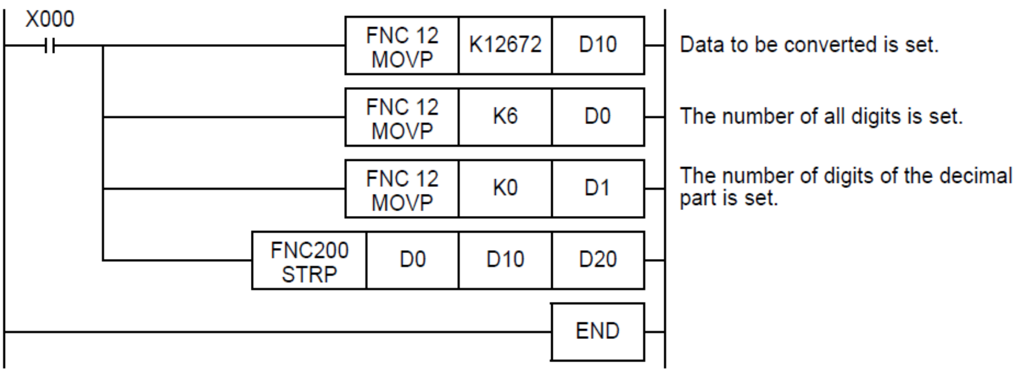
● An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067.

• When the range of "n" ("2n" in 32-bit operation) points starting from **S1.**,**S2.**,**D.** exceeds the corresponding device range (error code: K6706)

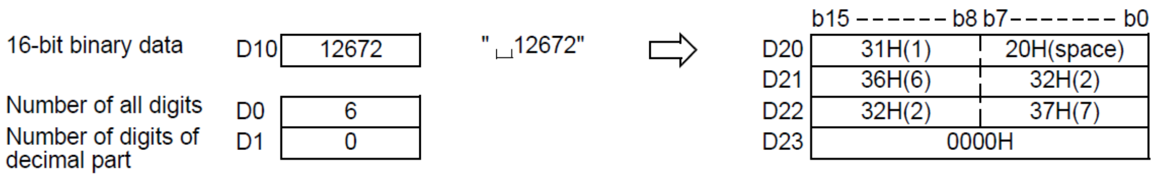
• When "n" points starting from **S1. and** "n" points starting from **D.** repeat ("2n" in 32-bit operation). (error code: K6706)

• When "n" points starting from **S2. and** "n" points starting from **D.** repeat ("2n" in 32-bit operation). (error code: K6706)

Program Example



◆ When X000=ON, Convert the BIN data (16 bits) saved in D10 to character string according to the number of digits specified in D0 and D1, and then store it to D20～D23 Program.



23.2 VAL/Character String to BIN Conversion

This instruction converts a character string (ASCII codes) into binary data.

On the other hand, EVAL (FNC117) instruction converts a character string (ASCII codes) into floating point data.

Instruction	Operand Type	Function						
D FNC201 VAL P	S. D1. D2.	16-bit Instruction	Mnemonic	Operation Condition		32-bit Instruction	Mnemonic	Operation Condition
		7 steps	VAL	Continuous Operation		13 steps	DVAL	Continuous Operation
			VALP	Pulse (Single) Operation			DVALP	Pulse (Single) Operation

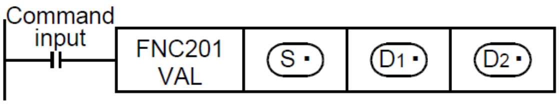
Operand number			
Operand number	S.	Head device number storing a character string to be converted into binary data **Applicable devices:** T, C, D, R, Modify	Character string
	D1.	Head device number storing the number of all digits of the binary data acquired by conversion **Applicable devices:** T, C, D, R, Modify	BIN16 bit
	D2.	Head device number storing the binary data acquired by conversion **Applicable devices:** KnY, KnM, KnS, T, C, D, R, Modify	BIN16/32 bit

Instruction Explanation

1. 16-bit operation(VAL,VALP)

A character string stored in **S.** and later is converted into 16-bit binary data. The number of all digits of the binary data acquired for conversion is stored in **D1.** , the number of digits of the decimal part is stored in **D1.+1**, and the converted binary data is stored in **D2.**.

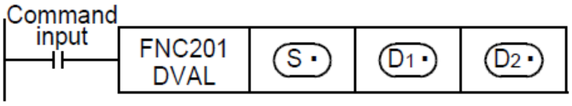
In converting a character string into binary data, the data from **S.** to a device number storing "00H" is handled as a character string in byte units.



2. 32-bit operation(DVAL,DVALP)

A character string stored in **S.** and later is converted into 32-bit binary data. The number of all digits of the binary data acquired for conversion is stored in **D1.**, the number of digits of the decimal part is stored in **D1.+1**, and the binary data is stored in **[D2.+1, D2.]**.

In conversion from a character string into binary data, the data from **S.** to a device number storing "00H" is handled as a character string in byte units.



1) Character string to be converted

a) Number of characters of character string and the numeric range when the decimal point is ignored

	Description	
	16-bit operation	**32-bit operation**
Number of all characters (digits)	2 ~ 8	2 ~ 13
Number of characters (digits) of decimal part	0 ~ 5,smaller than "number of all digits −3"	0 ~ 10,smaller than "number of all digits −3"
Numeric range when decimal point is ignored	-32768 ~ 32767 Example:“123.45”→“12345”	-2,147,483,648 ~ 2,147,483,647 Example:“12345.678”→“12345678”

b) Character types used in characters to be converted

		Character type
Sign	Positive numeric value	“Space(20H)”
	Negative numeric value	“-(2DH)”
Decimal point		“ . (2EH)”
Number		“0(30H)” ~ “9(39H)”

2) **D1.** stores the number of all digits. The number of all digits indicates the number of all characters (including the number, sign and decimal point).

3) **D1.+1** stores the number of digits of the decimal part. The number of digits of the decimal part indicates the number of all characters after the decimal point "." (2EH).

4) **D2./[D2.+1, D2.]** stores 16-bit data (bin) converted from a character string with the decimal point ignored.

For the character string located in and later, the "space" (20H) and "0" (30H) characters between the sign and the first number other than "0" are ignored in the conversion to 32-bit binary data.

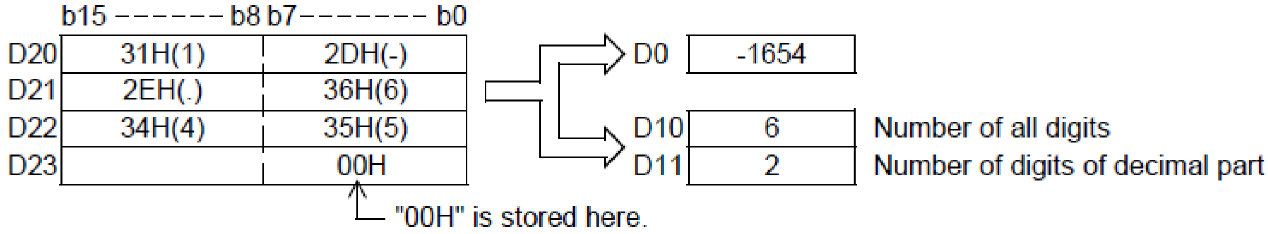
- Store sign data, "space (20H)" or "- (2DH)", must be stored in the 1st byte (lower order 8 bits of the head device set in **S.**).

  Only the ASCII code data "0 (30H)" to "9 (39H)", "space (20H)" and "decimal point (2EH)" can be stored from the 2nd byte to the "00H" at the end of the character string in **S.**.

  If "- (2DH)" is stored in the 2nd byte or later, an operation error (error code: K6706) occurs.

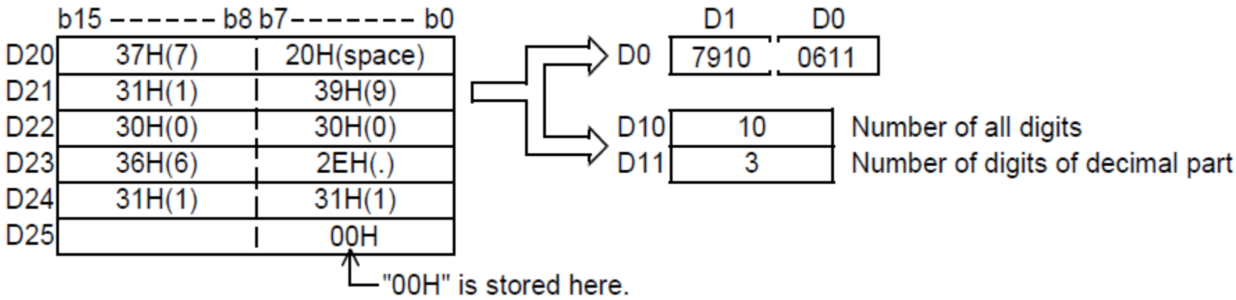
Program Example 1

- X0=ON, the character string data stored in D20 to D22 is regarded as an integer value, converted into a binary value, and stored in D0



Program Example 2

- X0=ON,the character string data stored in D20 to D24 is regarded as an integer value,converted into a binary value, and stored in D0



23.3 $+/Link Character Strings

This instruction links a character string to another character string

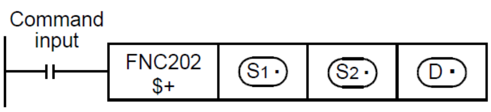
Instruction	Operand Type	Function						
		16-bit Instruction	Mnemonic	Operation Condition		32-Bit Instruction	Mnemonic	Operation Condition
FNC202 $+	S1. S2. D.	7 steps	$+	Continuous Operation			—	
P			$+P	Pulse (Single) Operation			—	

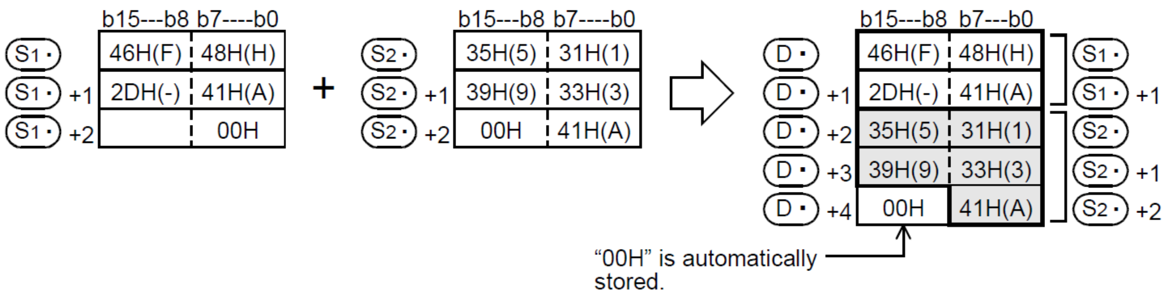
Operand number			
Operand number	S1.	Head device number storing the link source data (character string) or directly specified character string **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, Modify	Character string
	S2.	Head device number storing the link data (character string) or directly specified Character string **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, Modify	
	D.	Head device number storing the linked data (character string) **Applicable devices:** KnY, KnM, KnS, T, C, D, R, Modify	

Instruction Explanation	
Instruction Explanation	**1. 16-bit operation($+,$+P)**

The character string data stored in **S2.** and later is linked to the end of the character string data stored in **S1.** and later, and the linked data is stored to devices starting from **D.**.

A character string stored in **S1.** or **S2.** or later indicates the data from the specified device to the first "00H" in units of byte.





1) In linking, "00H" indicating the end of a character string specified in **S1.** is ignored, and a character string specified in **S2.** is linked to the last character specified in **S1.** .When a character string is linked, "00H" is automatically added at the end.

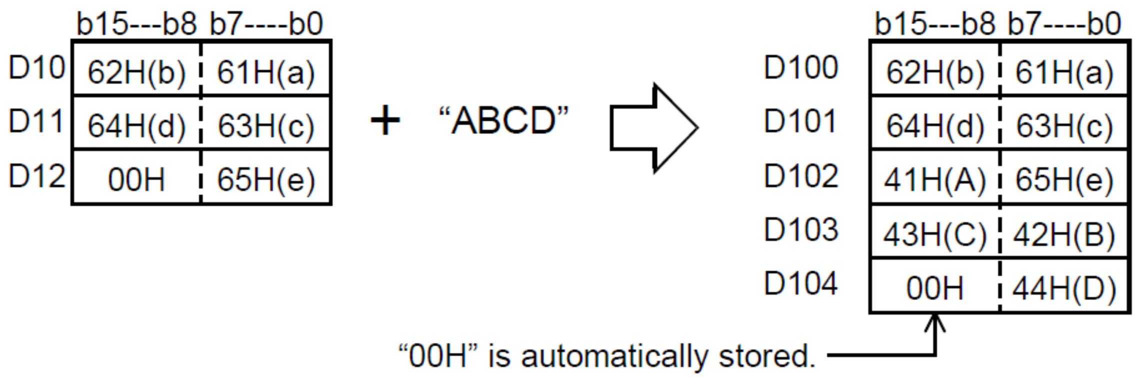
- When the number of characters after linking is odd, "00H" is stored in the high-order byte of the device storing the last character.
- When the number of characters after linking is even, "0000H" is stored in the device after the

last character.

- When directly specifying a character string, up to 32 characters can be specified (input). However, this limitation in the number of characters is not applied when a word device is specified in **S1.** or **S2.**.
- When the values in both **S1.** and **S2.** start from "00H" (that is, when the number of characters is “0”), "0000H" is stored in **D.**.
- An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067.
  • When the number of devices after a device number specified by **D.** is smaller than the number of devices required to store all linked character strings (that is, when “00H” cannot be stored after all character strings and the last character) (error code: K6706)
  • When the same device is specified in **S1.** ,**S2.** and **D.** as a device for storing a character string (error code: K6706)
  • When “00H” is not set within the corresponding device range after the device specified by **S1.** or **S2.** (error code: K6706)

Program Example

◆ X0=ON, a character string stored in D10 to D12 (abcde) is linked to the character string “ABCD”, and the result is stored to D100



23.4 LEN/Character String Length Detection

This instruction detects the number of characters (bytes) of a specified character string.

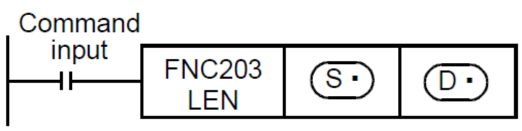
Instruction	Operand Type	Function						
		16-bit Instruction	Mnemonic	Operation Condition		32-bit Instruction	Mnemonic	Operation Condition
FNC203 LEN	**S.** **D.**	5 steps	LEN	Continuous Operation			—	
P			LENP	Pulse (Single) Operation			—	

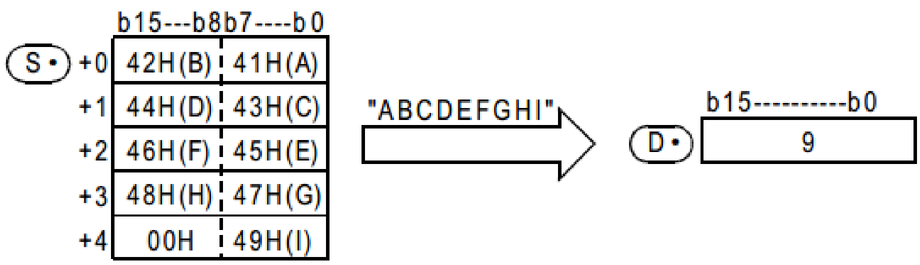
Operand number	**S.**	Head device number storing a character string whose length is to be detected **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, Modify	Character string
	D.	Device number storing the detected character string length (number of bytes) **Applicable devices:** KnY, KnM, KnS, T, C, D, R, Modify	BIN16 bit

Instruction Explanation

1. 16-bit operation(LEN,LENP)

The length of a character string stored in **S.** and later is detected, and stored to **D.**. Data starting from **S.** until the first device storing "00H" is handled as a character string in units of byte.

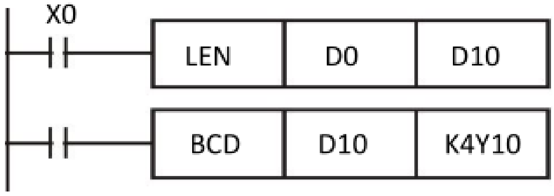




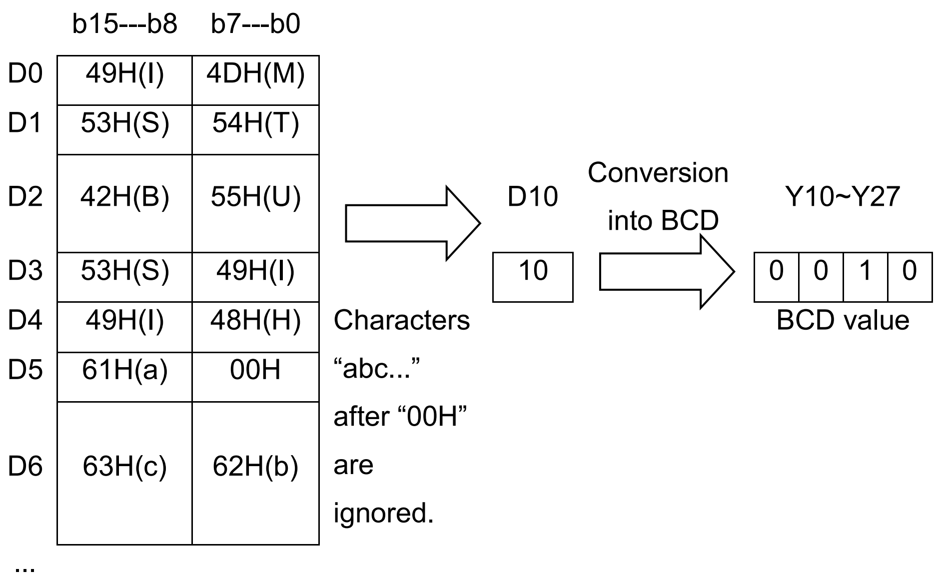
- This instruction can handle character codes other than ASCII codes, but the character string length is handled in byte units (8 bits). Accordingly, in the case of character codes in which 2 bytes express 1 character such as shift JIS codes, the length of 1 character is detected as “2”.

- An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067.
  • When “00H” is not set within the corresponding device range after a device specified by **S.** (error code: K6706)
  • When the detected number of characters is “32768” or more (error code: K6706)

Program Example 2



- X0=ON, the length of a character string stored in D0 and later is output in 4-digit BCD to Y10 to Y27.



23.5 RIGHT/Extracting Character String Data from the Right

This instruction extracts a specified number of characters from the right end of a specified character string.

Instruction	Operand Type	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
FNC204 RIGHT	**S.** **D.** **n**	5 steps	RIGHT	Continuous Operation		—	
P			RIGHTP	Pulse (Single) Operation		—	

Operand number			
Operand number	**S.**	Head device number storing a character string **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, Modify	Character string
	D.	Head device number storing extracted character string **Applicable devices:** KnY, KnM, KnS, T, C, D, R, Modify	
	n	Number of characters to be extracted **Applicable devices:** D, R, K, H	BIN16 bit

Instruction Explanation

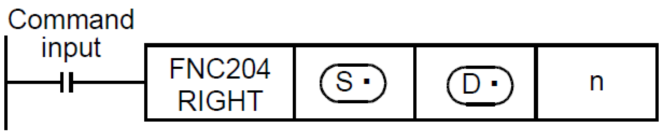
1. 16-bit operation(RIGHT,RIGHTP)

"n" characters are extracted from the right end (that is, from the end) of the character string data stored in **S.** and later, and stored to **D.** and later.

If the number of characters specified by "n" is "0", the NULL code (0000H) is stored to **D.**.

When characters are extracted from a character string, "00H" is automatically added at the end of the extracted characters.

- When the number of extracted characters is odd, "00H" is stored in the high-order byte of a device storing the last character.
- When the number of extracted characters is even, "0000H" is stored in the device after the last character.



- When handling character codes other than ASCII codes, note the following contents:

  • The number of characters is handled in byte units (8 bits). Accordingly, in the case of character codes in which 2 bytes express 1 character such as shift JIS codes, the length of 1 character is detected as "2".

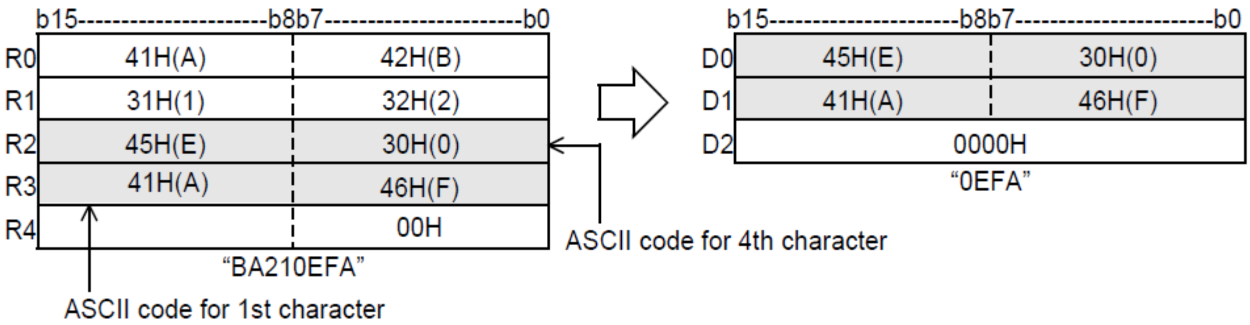
  • When extracting characters from a character string including character codes in which 2 bytes express 1 character such as shift JIS codes, consider the number of characters to be extracted in units of character codes for 1 character.

  Note that the expected character code is not given if only 1 byte is executed out of a 2-byte

	character code. ● An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067. • When “00H” is not set within the corresponding device range after a device specified by **S.** (error code:K6706) • When “n” exceeds the number of characters specified by **S.** (error code: K6706) • When the number of devices after a device number specified by **D.** is smaller than the number of devices required to store extracted “n” characters (that is, when “00H” cannot be stored after all character strings and the last character) (error code: K6706) • When “n” is a negative value (error code: K6706)

Program Example

◆ X0=ON, 4 characters are extracted from the right end of the character string data stored in R0 and later, and stored to D0 and later



23.6 LEFT/Extracting Character String Data from the Left

This instruction extracts a specified number of characters from the left end of a specified character string.

Instruction	Operand Type	Function					
		16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
FNC205 LEFT	**S.** **D.** **n**	7 steps	LEFT	Continuous Operation		—	
P			LEFTP	Pulse (Single) Operation		—	

Operand number	**S.**	Head device number storing a character string **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, Modify	Character string
	D.	Head device number storing extracted character string **Applicable devices:** KnY, KnM, KnS, T, C, D, R, Modify	
	n	Number of characters to be extracted **Applicable devices:** D, R, K, H	BIN16 bit

Instruction Explanation

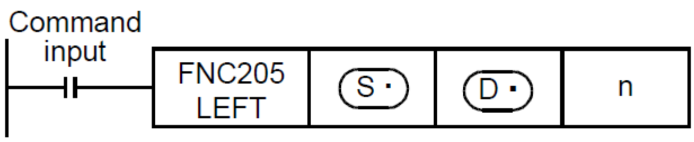
1. 16-bit operation(LEFT,LEFTP)

"n" characters are extracted from the left end (that is, from the head) of the character string data stored in **S.** and later and stored to **D.** and later.

If the number of characters specified by "n" is "0", the NULL code (0000H) is stored to **D.**.

When characters are extracted from a character string, "00H" is automatically added at the end of the extracted characters.

- When the number of extracted characters is odd, "00H" is stored in the high-order byte of a device storing the last character.
- When the number of extracted characters is even, "0000H" is stored in the device after the last character.



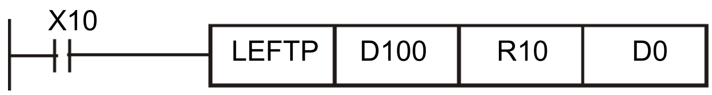
● When handling character codes other than ASCII codes, note the following contents:

• The number of characters is handled in byte units (8 bits). Accordingly, in the case of character codes in which 2 bytes express 1 character such as shift JIS codes, the length of 1 character is detected as "2".

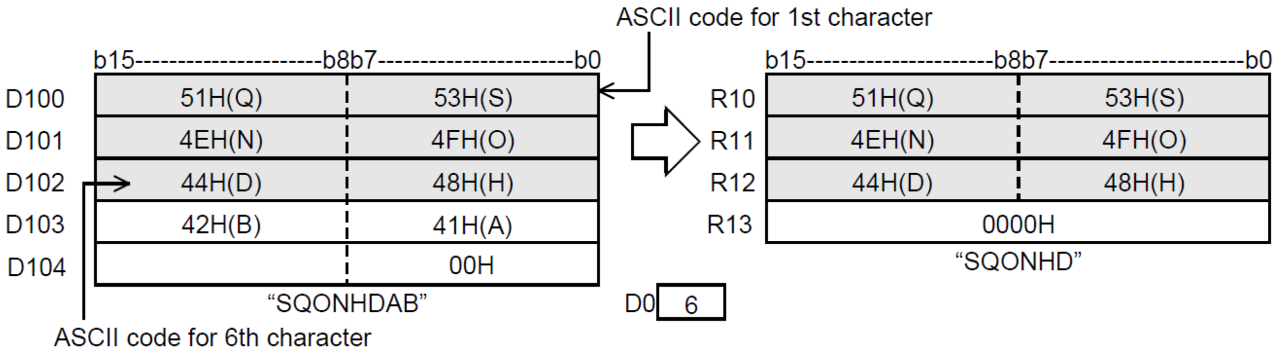
• When extracting characters from a character string including character codes in which 2 bytes express 1 character such as shift JIS codes, consider the number of characters to be extracted in units of character codes for 1 character.

	Note that the expected character code is not given if only 1 byte is executed out of a 2-byte character code. ● An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067. • When “00H” is not set within the corresponding device range after a device specified by **S.** (error code: K6706) • When “n” exceeds the number of characters specified by **S.** (error code: K6706) • When the number of devices after a device number specified by **D.** is smaller than the number of devices required to store extracted “n” characters (that is, when “00H” cannot be stored after all character strings and the last character) (error code: K6706) • When “n” is a negative value (error code: K6706)

Program Example 2



◆ X10=ON,the number of characters which is equivalent to the number stored in D0 are extracted from the left end of the character string data stored in D100 and later, and stored to R10 and later.



23.7 MIDR/Random Selection of Character Strings

This instruction extracts a specified number of characters from arbitrary positions of a specified character string.

Instruction	Operand Type	16 -bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
FNC206 MIDR	S1. D. S2.	7 s	MIDR	Continuous Operation		—	
P			MIDRP	Pulse (Single) Operation		—	

Operand number	S1.	Head device number storing a character string **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, Modify	Character string
	D.	Head device number storing extracted character string **Applicable devices:** KnY, KnM, KnS, T, C, D, R, Modify	
	S2.	Head device number specifying the head position and number of characters to be Extracted **S2.** : Head character position **S2.+1** : Number of characters **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, Modify	BIN16 bit

Instruction Explanation

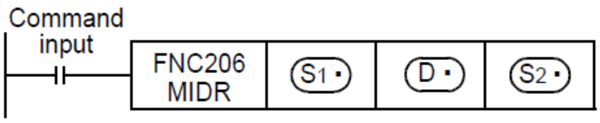
1. 16-bit operation(MIDR,MIDRP)

" **S2.+1**" characters are extracted leftward from the position specified by **S2.** of the character string data stored in **S1.** and later, and stored to **D.** and later.

When characters are extracted from a character string, "00H" is automatically added at the end of the extracted characters.

- When the number of extracted characters specified by +1 is odd, "00H" is stored in the high-order byte of a device storing the last character.

- When the number of extracted characters specified by +1 is even, "0000H" is stored in the device after the last character.



- When handling character codes other than ASCII codes, note the following contents:

  • The number of characters is handled in byte units (8 bits). Accordingly, in the case of character codes in which 2 bytes express 1 character such as shift JIS code, the length of 1 character is regarded as 2 characters.

  • When extracting characters from a character string including character codes in which 2 bytes express 1 character such as shift JIS codes, consider the number of characters to be extracted in units of character codes for 1 character.

Note that the expected character code is not given if only 1 byte is executed out of a 2-byte character code

- An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067.

  • When "00H" is not set within the corresponding device range after a device specified by **S1.** (error code: K6706)

  • When the value specified by **S2.** exceeds the number of characters specified by **S1.** (error code: K6706)

  • When the number of characters specified by **[S2.+1]** from the position specified by **D.** exceeds the device range specified by **D.** (error code: K6706)

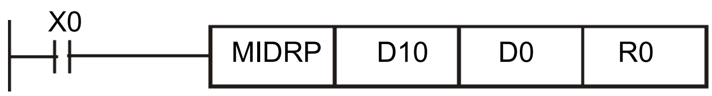
  • When the number of devices after a device number specified by **D.** is smaller than the number of devices required to store extracted characters as many as the number specified by **[S2.+1]** (that is, when "00H" cannot be stored after all character strings and the last character) (error code: K6706)

  • When "n" is a negative value (error code: K6706)

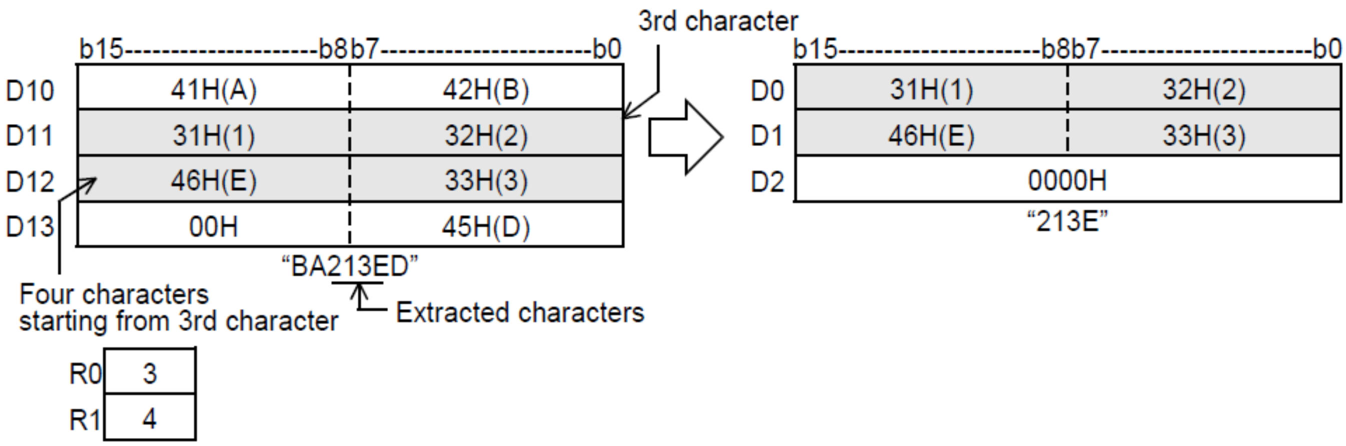
  • When **S2.+1** specifies "-2" or less (error code: K6706)

  • When **S2.+1** specifies a number larger than the number of characters specified by **S1.** (error code: K6706)

Program Example



◆ X0=ON,four characters are extracted from the 3rd character from the left end of the character string data stored in D10 and later, and then stored to D0.



23.8 MIDW/Random Replacement of Character Strings

This instruction replaces the characters in arbitrary positions inside designated character string with a specified character string.

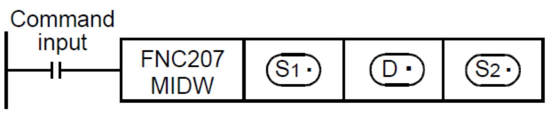
Instruction	Operand Type	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
FNC207 MIDW P	**S1. D. S2.**	7 steps	MIDW	Continuous Operation		—	
			MIDWP	Pulse (Single) Operation		—	

Operand number			
Operand number	**S1.**	Head device number storing a character string used in overwriting **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, Modify	Character string
	D.	Head device number storing character string to be overwritten **Applicable devices:** KnY, KnM, KnS, T, C, D, R, Modify	
	S2.	Head device number specifying the head position and number of characters to be Overwritten **S2.** : Head character position to be overwritten **S2.+1** : Number of characters to be overwritten **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, Modify	BIN16 bit

Instruction Explanation

1. 16-bit operation(MIDW,MIDWP)

" **S2.+1**" characters are extracted from the left end (that is, the head) of the character string data stored in **S1.** and later, and stored to the position specified by **S2.** and later of the character string data stored in **D.** and later.



- This instruction can handle character codes other than ASCII codes, but please note the following:

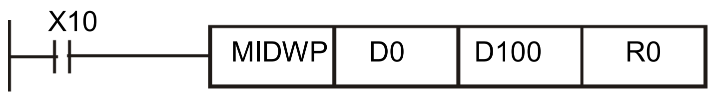
  • The number of characters is handled in byte units (8 bits). Accordingly, in the case of character codes in which 2 bytes express 1 character such as shift JIS code, the length of 1 character is regarded as 2 characters.

  • When overwriting a character string including character codes in which 2 bytes express 1 character such as shift JIS codes, consider the number of characters to be extracted in units of character codes for 1 character.

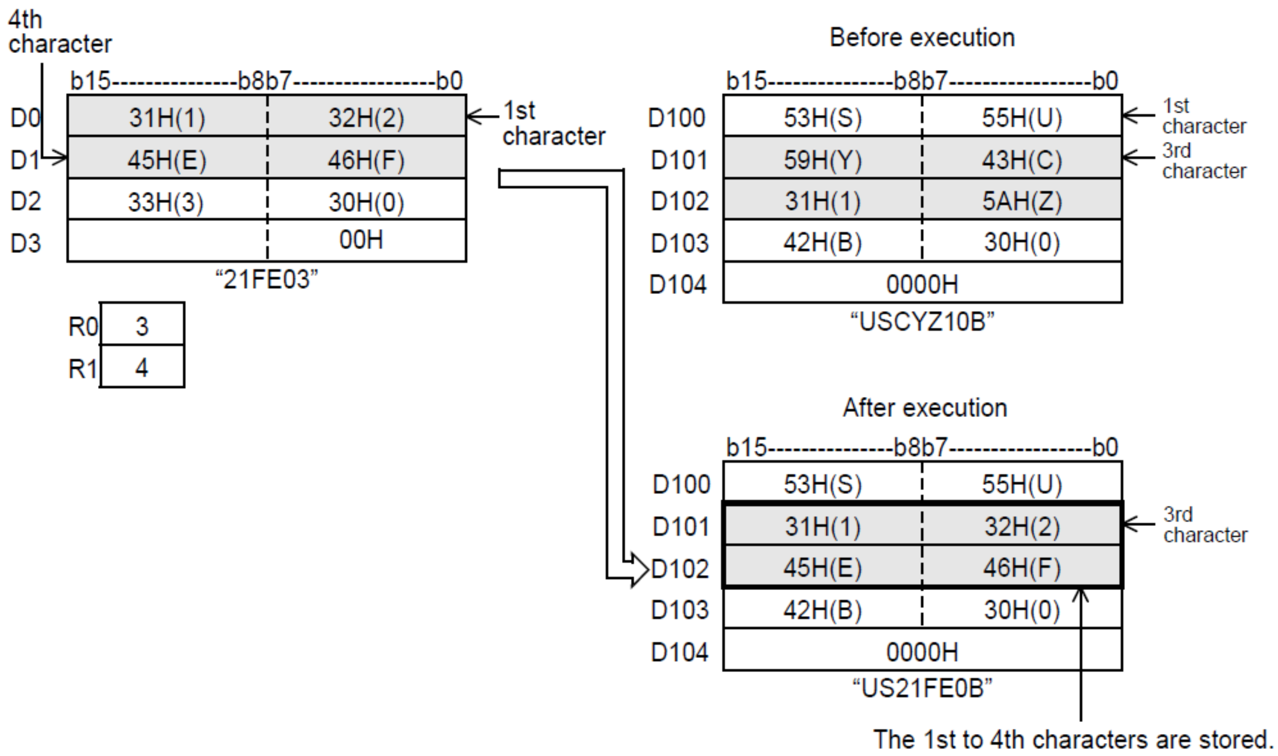
  Note that the expected character code is not given if only 1 byte is overwritten out of a 2-byte character code.

	● An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067. • When "00H" is not set within the corresponding device range after a device specified by **S1.** or **D.** (error code: K6706) •When the value specified by **S2.** exceeds the number of characters of the character string stored in **D.** and later (error code: K6706) • When **S2. n** specifies a negative value (error code: K6706) • When **S2.+1** specifies "-2" or less (error code: K6706) • When the number of characters specified by **S2.+1** exceeds the number of characters specified by **S1.** (error code: K6706)

Program Example



◆ X10=ON,4 characters are extracted from the character string data stored in D0 and later, and stored to the 3rd character (from the left end) and later for the character string data stored in D100 and later



23.9 INSTR/Character string search

This instruction searches a specified character string within another character string.

Instruction	Operand Type	Function					
FNC208 INSTR P	**S1. S2. D. n**	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
		9 steps	INSTR	Continuous Operation		—	
			INSTRP	Pulse (Single) Operation		—	

Operand number			
	S1.	Head device number storing a character string **Applicable devices:** T, C, D, R, Modify	Character string
	S2.	Head device number storing a character string to be searched **Applicable devices:** T, C, D, R, Modify	
	D.	Head device number storing search result **Applicable devices:** T, C, D, R, Modify	BIN16 bit
	n	Search start position **Applicable devices:** D, R, K, H	

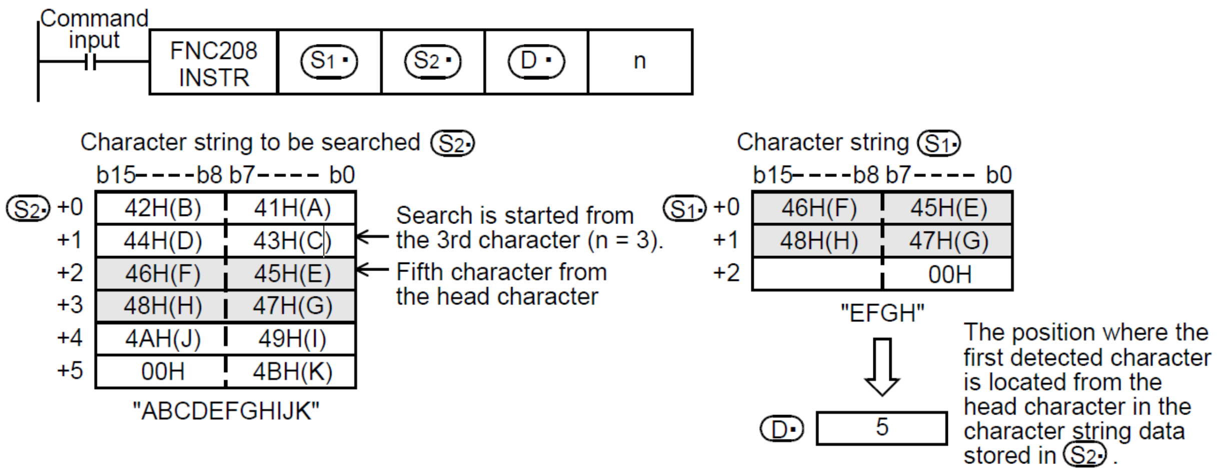
Instruction Explanation

1. 16-bit operation(INSTR,INSTRP)

1) The character string stored in **S1.** and higher is searched for within the character string **S2.** and higher.

The search begins at the "n"th character from the left end (head character) of **S2.** and the search result is stored in **D.**.

The search result provides the first matching character (located from the left end (head character)) in **S2.** .



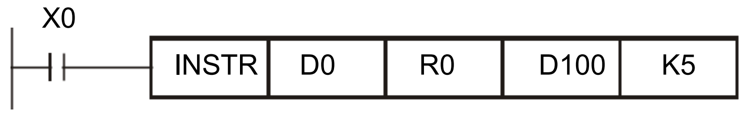
2) When the searched character string is not detected, "0" is stored in **D.**.

3) When the search start position "n" is a negative number or "0", search processing is not executed.

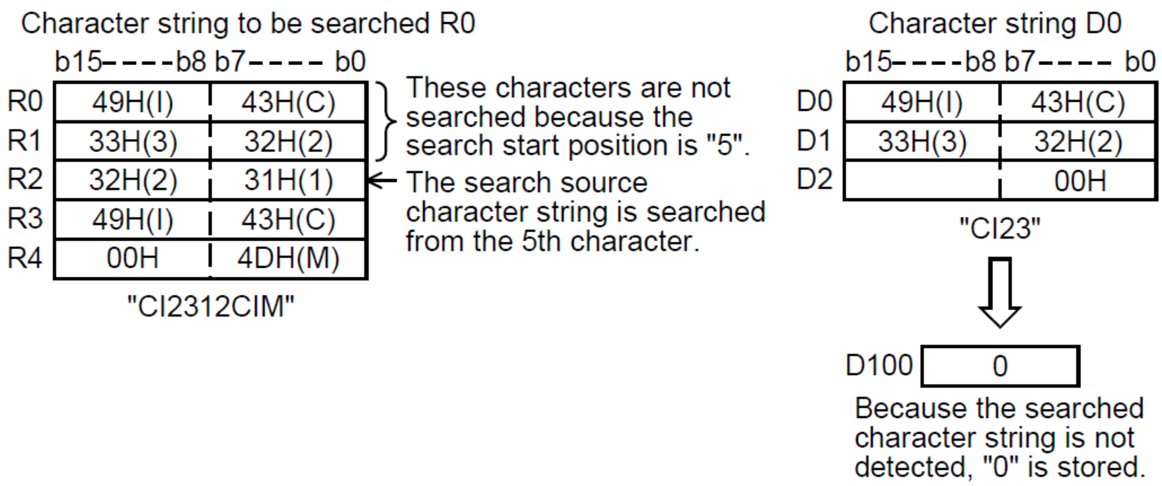
4) character string can be directly specified in the character string **S1.**.

- An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067.
  • When the search start position "n" exceeds the number of characters stored in **S2.**(error code: K6706).
  • When "00H (NULL)" is not located within the corresponding device range starting from **S1.**/**S2.** (error code: K6706)

Program Example



◆ X0=ON,the character string "CI23" (D0 and later) is searched from the 5th character from the left end (head character) of the character string "CI2312CIM" (R0 and later),The search result is stored in D100.



23.10 $MOV/Character String Transfer

This instruction transfers character string data.

Instruction	Operand Type	Function					
FNC209 $MOV **P**	**S.** **D.**	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
		5 steps	$MOV	Continuous Operation		—	
			$MOVP	Pulse (Single) Operation		—	

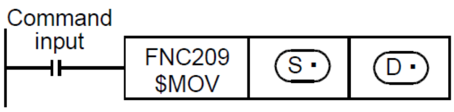
Operand number	**S.**	Directly specified character string (up to 32 characters) or head device number storing character string which is handled as the transfer source **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, Modify	Character string
	D.	Head device number storing transferred character string **Applicable devices:** KnY, KnM, KnS, T, C, D, R, Modify	

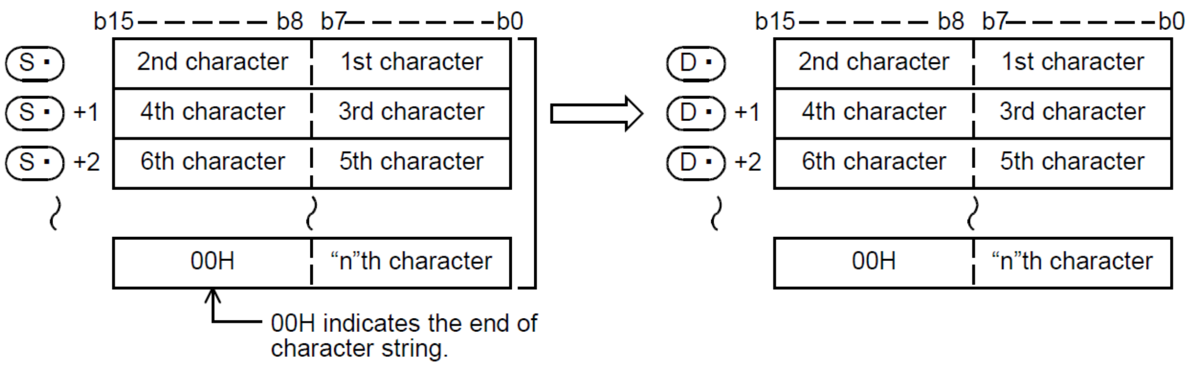
Instruction Explanation

1. 16-bit operation($MOV,$MOVP)

The character string data stored in the device specified by **S.** and later is transferred to the device specified by **D.** and later.

From the device number specified by **S.** to a device after that which stores “00H” in its high-order or low-order byte are transferred at one time.



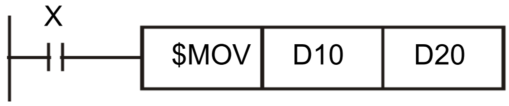


Even if the device range **[S. ~ S.+n]** storing the transfer source character string data overlaps the device range **[S. ~ S.+n/2]** storing the transferred character string data, transfer is executed.

- When “00H” is stored in the low-order byte of **S.+n**, “00H” is stored to both the high-order byte and low-order byte of **D.+n**.

	● An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067. • When “00H” does not exist in the range specified from device **S.** (error code: K6706) • When the specified character string cannot be stored in devices from the device specified by **D.** to the last device (error code: K6706)

Program Example



◆ X0=ON,character string data stored in D10～D12 is transferred to D20～D22 program.

	b15---b8	b7---b0
D10	4DH(M)	2AH(*)
D11	45H(E)	45H(E)
D12		00H

	b15---b8	b7---b0
D20	4DH(M)	2AH(*)
D21	45H(E)	45H(E)
D22	0000H	

24 Data Operation 3 – FNC210 to FNC219

FNC NO.	Instruction	Function	Supported PLC series		
			3G PLC	2N PLC	MX2N PLC
210	FDEL	Deleting Data from Tables	★		
211	FINS	Inserting Data to Tables	★		
212	POP	Shift Last Data Read [FILO Control]	★		
213	SFR	16-bit data n Bit Shift Right with Carry	★		
214	SFL	16-bit data n Bit Shift Left with Carry	★		
215	—				
216	—				
217	—				
218	—				
219	—				

24.1 FDEL/Deleting Data from Tables

This instruction deletes an arbitrary data from a data table.

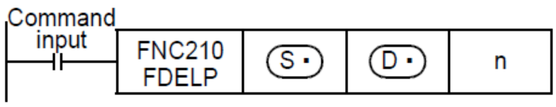
Instruction	Operand Type	Function						
FNC210 FDEL P	S. D. n	16-bit Instruction	Mnemonic	Operation Condition		32-bit Instruction	Mnemonic	Operation Condition
		5steps	FDEL	Continuous Operation			—	
			FDELP	Pulse (Single) Operation			—	

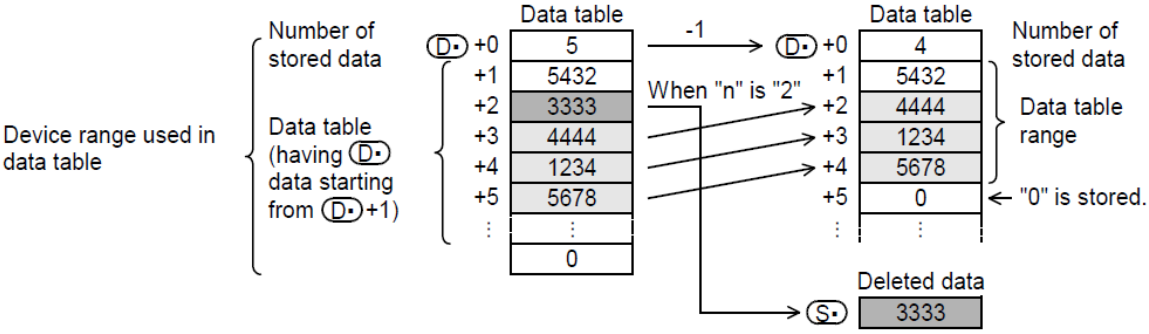
Operand number			
Operand number	S.	Device number storing deleted data **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, Modify	BIN16 bit
	D.	Head device number in data table **Applicable devices:** KnY, KnM, KnS, T, C, D, R, Modify	
	n	Position of deleted data in table **Applicable devices:** KnY, KnM, KnS, T, C, D, R, Modify	

Instruction Explanation

1. 16-bit operation(FDEL,FDELP)

"n"th data is deleted from a data table (stored in **D.** and later), and the deleted data is stored in **S.**. "n+1"th data and later in the data table are shifted forward one by one, and the number of stored data is subtracted by "−1".



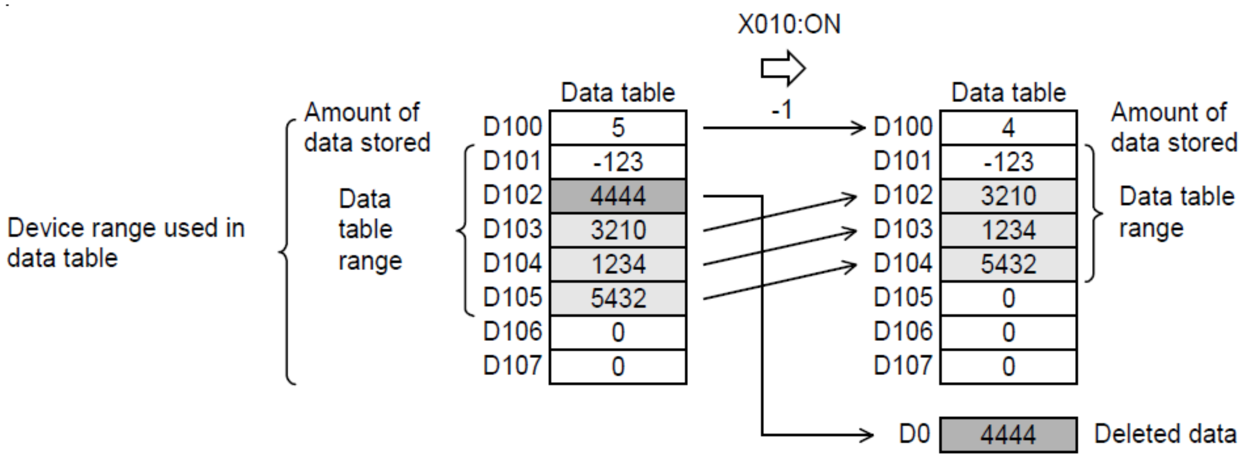


- The device range used in a data table should be controlled by the user.
  The data table has **D.** data starting from the next device (**D.**+1) after **D.** indicating the number of stored data.
- An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067.
  • When the table position "n" of the data to be deleted exceeds the amount of data stored (error code: K6706)
  • When the value "n" exceeds the device range of the data table **D.** (error code: K6706)

	• When the FNC210 instruction is executed under the condition "n ≤ 0" (error code: K6706) • When the amount of data stored specified in **D.** is "0" (error code: K6706) • When the data table range exceeds the corresponding device range (error code: K6706)

Program Example

◆ The 2nd data entry is deleted from the data table stored in D100 to D105, and the deleted data is stored in D0. When the amount of data stored is "0", however, the FDEL (FNC210) instruction is not executed. (The device range used in the data table is D100～D107).



24.2 FINS/Inserting Data to Tables

This instruction inserts data into an arbitrary position in a data table.

Instruction	Operand Type	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
FNC211 FINS **P**	**S.** **D.** **n**	7 steps	FINS	Continuous Operation		—	
			FINSP	Pulse (Single) Operation		—	

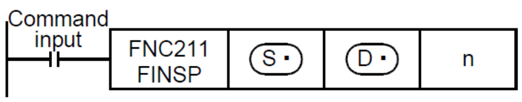
Operand number			
Operand number	**S.**	Device number storing inserted data **Applicable devices:** T, C, D, R, K, H, Modify	BIN16 bit
	D.	Head device number in data table **Applicable devices:** T, C, D, R, K, H, Modify	
	n	Data insertion position in table **Applicable devices:** D, R, K, H	

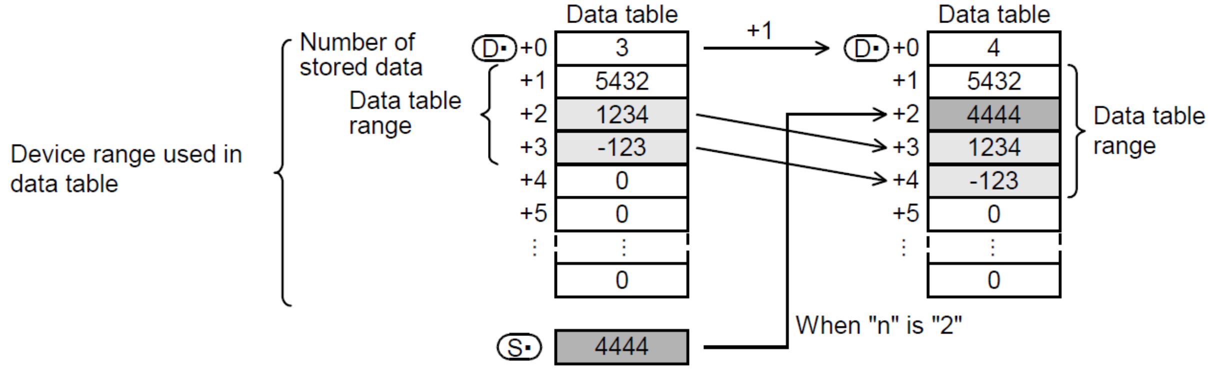
Instruction Explanation

1. 16-bit operation(FINS,FINSP)

16-bit data **S.** is inserted in "n"th position in a data table (stored in **D.** and later).

"n"th data and later in the data table are shifted backward one by one, and the number of stored data is added by "1".

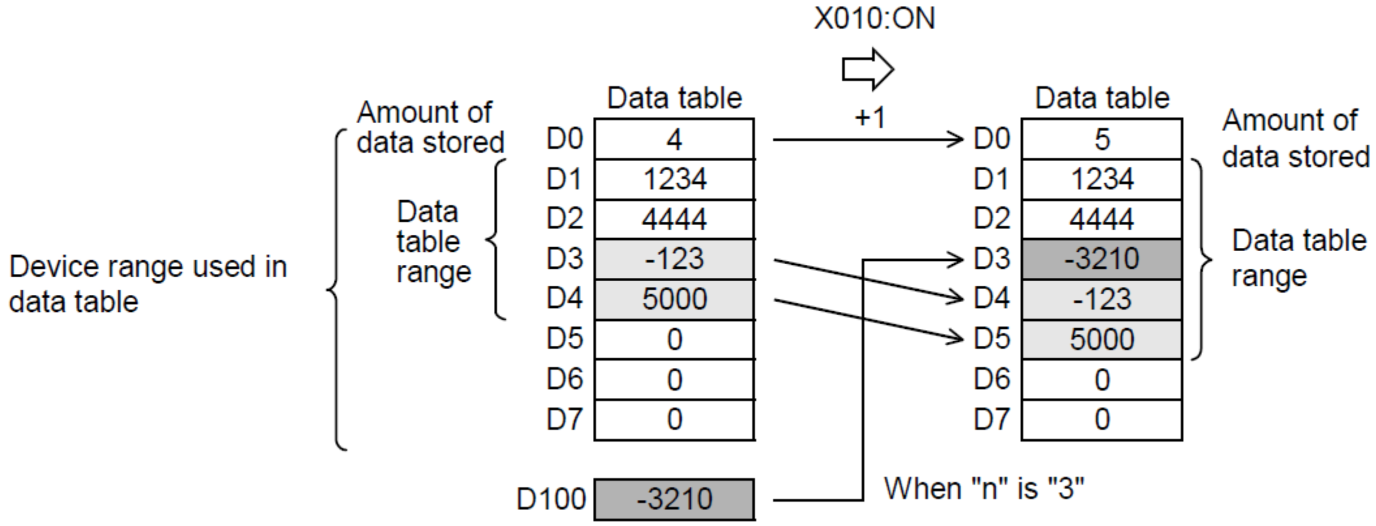




- The device range used in a data table should be controlled by the user.
  The data table has **D.** data starting from the next device (**D.+1**) after **D.** indicating the number of stored data.
- An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067.
  - When the table position "n" for data insertion exceeds the amount of stored data plus 1(error code: K6706)
  - When the value "n" exceeds the device range of the data table **D.** (error code: K6706)
  - When FNC211 instruction is executed under the condition "n ≤ 0" (error code: K6706)
  - When the data table range exceeds the corresponding device range (error code: K6706)

Program Example

- X010=ON,data stored in D100 is inserted into the 3rd position of the data table stored in D0 to D4. When the amount of data stored exceeds "7", however, the FINS (FNC211) instruction is not executed. (The device range used in the data table is D0 to D7)



24.3 POP/Shift Last Data Read [FILO Control]

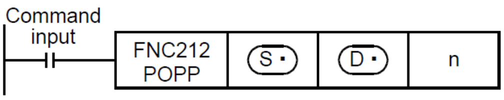
This instruction reads the last data written by shift write (SFWR) instruction for FILO control.

<table>
<tr><th rowspan="2">Instruction</th><th rowspan="2">Operand Type</th><th colspan="7">Function</th></tr>
<tr><td>16-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td><td></td><td>32-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td></tr>
<tr><td rowspan="2">FNC212 POP
P</td><td rowspan="2">S.
D.
n</td><td rowspan="2">7 steps</td><td>POP</td><td>Continuous Operation</td><td rowspan="2"></td><td rowspan="2"></td><td>—</td><td></td></tr>
<tr><td>POPP</td><td>Pulse (Single) Operation</td><td>—</td><td></td></tr>
<tr><td rowspan="3">Operand number</td><td>S.</td><td colspan="5">Head device number storing first-in data (including pointer data)
Applicable devices: KnY, KnM, KnS, T, C, D, R, Modify</td><td colspan="2" rowspan="3">BIN16 bit</td></tr>
<tr><td>D.</td><td colspan="5">Device number storing last-out data
Applicable devices: KnY, KnM, KnS, T, C, D, R, Modify</td></tr>
<tr><td>n</td><td colspan="5">Length of data array
(Add "1" because pointer data is also included.) 2≦n≦512
Applicable devices: K, H</td></tr>
</table>

<table>
<tr><td rowspan="2">Instruction Explanation</td><td>

1. 16-bit operation(POP,POPP)

16-bit data **S.** is inserted into the n^th of the data table (D. and later). The data after the n^th data table is shifted back one by one, the data Add 1 to the saved number.



	Description
S.	Pointer data (amount of data stored)
S.+1	Data area (First-in data written by shift write (SFWR) instruction)
S.+2	
S.+3	
...	
S.+n-3	
S.+n-2	
S.+n-1	

</td></tr>
<tr><td>

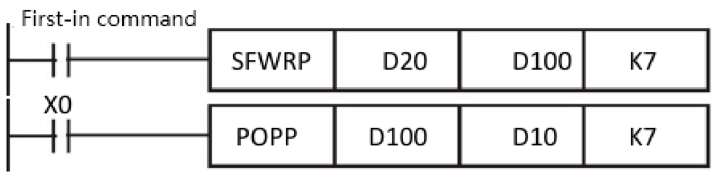
1) Every time the instruction is executed for the word devices [**S.** ~ **S.+n-1**], a device "**S.**+ Pointer data **S.**" is read to **D.**. (The last data entry written by the shift write (SFWR) instruction for first-in first-out control is read to **D.**.) Specify "n" in the range 2 ~ 512.

2) Subtract "1" from the value of the pointer data **S.**

</td></tr>
</table>

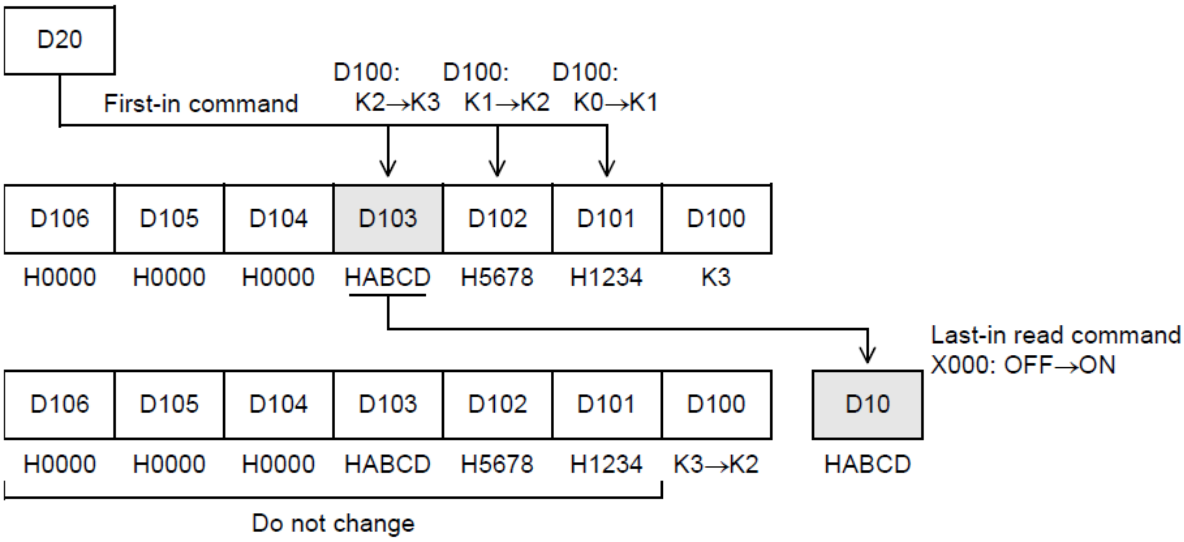
- the zero flag M8020, when current value of the pointer **S.=0**, It turns ON after executing Instruction.
- When this instruction is programmed in the continuous operation type, the instruction is executed in every operation cycle. As a result, an expected operation may not be achieved.
  Usually, program this instruction in the “pulse operation type”, or let this instruction be executed by a “pulsed command contact”.
- An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067.
  - **S.>**n-1, (error code: K6706)
  - **S.<**0,(error code: K6706)

Program Example



- X0=ON, Among values stored in D20 input first to D101 to D106, the last value input is stored to D10, and “1” is subtracted from the number of stored data (pointer D100)
- When the first-in data are as shown in the table below

Pointer	**D100**	**K3**
Data	**D101**	**H1234**
	D102	**H5678**
	D103	**HABCD**
	D104	**H0000**
	D105	**H0000**
	D106	**H0000**



24.4 SFR/Bit Shift Right with Carry

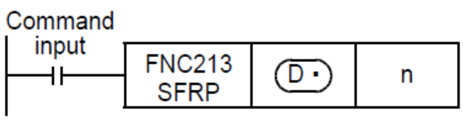
This instruction shifts 16 bits stored in a word device rightward by “n” bits.

Instruction		Operand Type	Function					
			16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
FNC213 SFR		**D.** **n**	5 steps	SFR	Continuous Operation		—	
	P			SFRP	Pulse (Single) Operation		—	

Operand number			
Operand number	**D.**	Device number storing data to be shifted **Applicable devices:** KnY, KnM, KnS, T, C, D, R, Modify	BIN16 bot
	n	Number of times of shift (0 ≤ n ≤ 15) **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, Modify	

Instruction Explanation

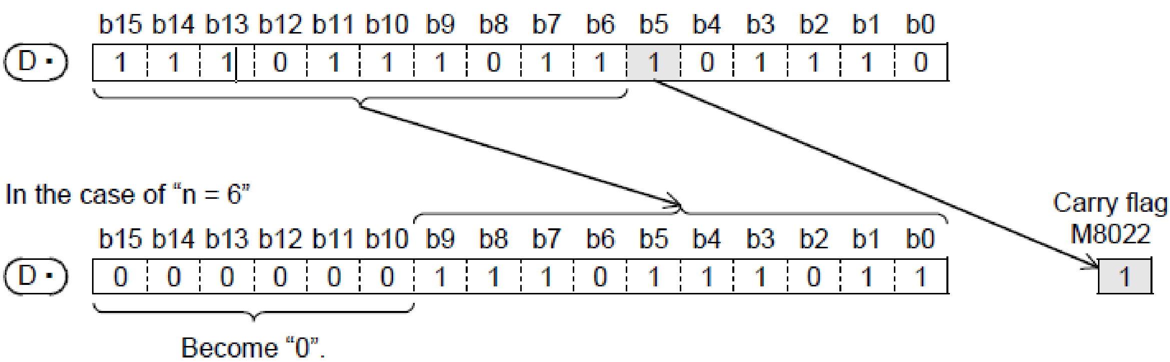
1. 16-bit operation(SFR,SFRP)



1) 16 bits stored in a word device **D.** are shifted rightward by “n” bits.
Specify a value in the range from “0” to “15” as “n”.
If “16” or larger value is specified as “n”, 16 bits are shifted rightward by the remainder of “n/16”.
For example, when “n” is set to “18”, 16 bits are shifted rightward by 2 bits (18/16 = 1 ... 2).

2) The ON (1)/OFF (0) status of the “n”th bit (bit “n-1”) in the word device **D.** is transferred to the carry flag M8022.

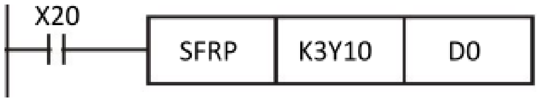
3) “0” is set to “n” bits from the most significant bit.



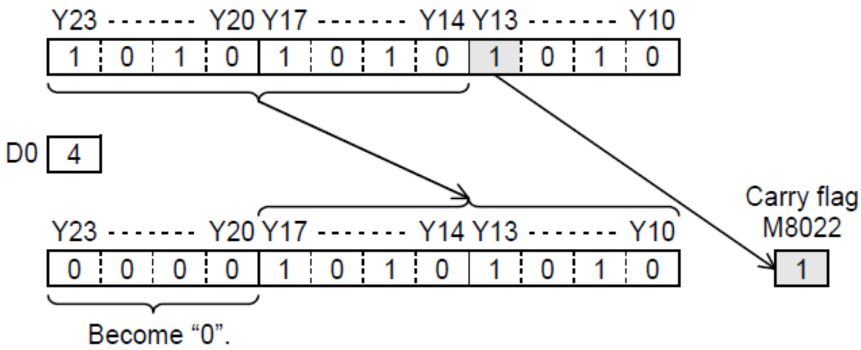
Shifts the ON/OFF status of bit “n-1”.

- An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067.
  - When a negative value is set to “n” (error code: K6706)

Program Example 2



◆ X020=ON,the contents of Y010 to Y023 are shifted rightward by the number of bits specified by D0



24.5 SFL/Bit Shift Left with Carry

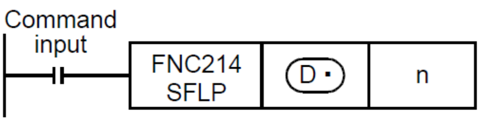
This instruction shifts 16 bits stored in a word device leftward by "n" bits.

Instruction		Operand Type	16-bit Instruction	Mnemonic	Operation Condition	32-bt Instruction	Mnemonic	Operation Condition
FNC214 SFL		**D. n**	5 steps	SFL	Continuous Operation		—	
	P			SFLP	Pulse (Single) Operation		—	

Operand number	Operand	Description	
Operand number	**D.**	Device number storing data to be shifted **Applicable devices:** KnY, KnM, KnS, T, C, D, R, V, Z, Modify	BIN16 bit
	n	Number of times of shift (0 ≤ n ≤ 15) **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, Modify	

Instruction Explanation

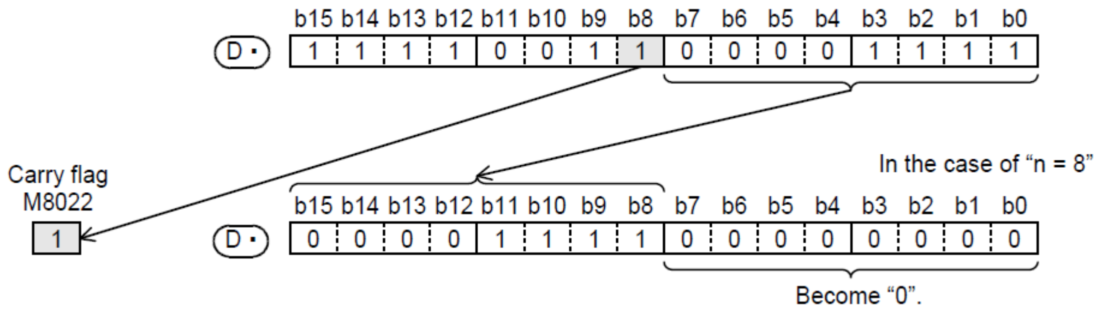
1. 16-bit operation(SFL,SFLP)



1) 16 bits stored in a word device **D.** are shifted leftward by "n" bits.
Specify a value in the range from "0" to "15" as "n".
If "16" or larger value is specified as "n", 16 bits are shifted leftward by the remainder of "n/16".
For example, when "n" is set to "18", 16 bits are shifted leftward by 2 bits (18/16 = 1 ... 2).

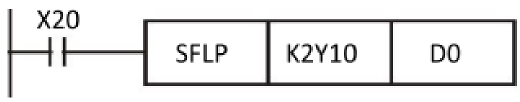
2) The ON (1)/OFF (0) status of the “n+1”th bit (bit “n”) in the word device is transferred to the carry flag M8022.

3) “0” is set to “n” bits from the least significant bit.

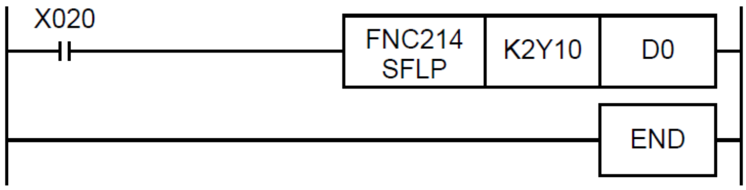


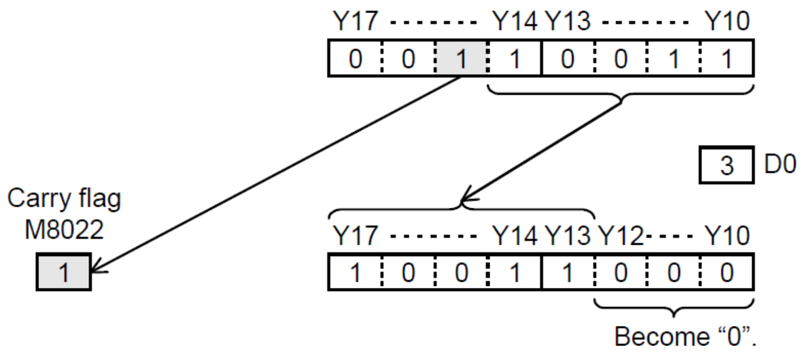
- the carry flag M8022,Shifts the ON/OFF status of bit “n”.
- An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067.
  - When a negative value is set to “n” (error code: K6706)

Program Example 2



◆ X020=ON, the contents of Y010 to Y017 are shifted leftward by the number of bits specified by D0





25 Data Comparison – FNC220 to FNC249

FNC NO.	Instruction	Function	Supported PLC series		
			3G PLC	2N PLC	MX2N PLC
220	—				
221	—				
222	—				
223	—				
224	LD=	Load Compare **S1.=S2.**	★	★	★
225	LD>	Load Compare **S1.>S2.**	★	★	★
226	LD<	Load Compare **S1.<S2.**	★	★	★
227	—				
228	LD<>	Load Compare **S1.<>S2.**	★	★	★
229	LD<=	Load Compare **S1.<=S2.**	★	★	★
230	LD>=	Load Compare **S1.>=S2.**	★	★	★
231	—				
232	AND=	AND Compare **S1.=S2.**	★	★	★
233	AND>	AND Compare **S1.>S2.**	★	★	★
234	AND<	AND Compare **S1.<S2.**	★	★	★
225	—				
236	AND<>	AND Compare **S1.<>S2.**	★	★	★
237	AND<=	AND Compare **S1.<=S2.**	★	★	★
238	AND>=	AND Compare **S1.>=S2.**	★	★	★
239	—				
240	OR=	OR Compare **S1.=S2.**	★	★	★
241	OR>	OR Compare **S1.>S2.**	★	★	★
242	OR<	OR Compare **S1.<S2.**	★	★	★
243	—				
244	OR<>	OR Compare **S1.<>S2.**	★	★	★
245	OR<=	OR Compare **S1.<=S2.**	★	★	★
246	OR>=	OR Compare **S1.>=S2.**	★	★	★
247	—				
248	—				
249	—				

25.1 LD=, >, <, <>, <=, >=/Data Comparison

These instructions compare numeric values, and set a contact to ON when the condition agrees so that an operation is started.

Instruction	Operand Type	Function						
D FNC224 LD=	S1. S2.	16-bit Instruction	Mnemonic	Operation Condition		32-bit Instruction	Mnemonic	Operation Condition
		5 steps	LD=	Continuous Operation		9 steps	LDD=	Continuous Operation
D FNC225 LD>	S1. S2.	16 -bit Instruction	Mnemonic	Operation Condition		32-bit Instruction	Mnemonic	Operation Condition
		5 steps	LD>	Continuous Operation		9 steps	LDD>	Continuous Operation
D FNC226 LD<	S1. S2.	16-bit Instruction	Mnemonic	Operation Condition		32-bit Instruction	Mnemonic	Operation Condition
		5 steps	LD<	Continuous Operation		9 steps	LDD<	Continuous Operation
D FNC227 LD<>	S1. S2.	16-bit Instruction	Mnemonic	Operation Condition		32-bit Instruction	Mnemonic	Operation Condition
		5 steps	LD<>	Continuous Operation		9 steps	LDD<>	Continuous Operation
D FNC228 LD<=	S1. S2.	16-bit Instruction	Mnemonic	Operation Condition		32-bit Instruction	Mnemonic	Operation Condition
		5 steps	LD<=	Continuous Operation		9 steps	LDD<=	Continuous Operation

		16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
FNC229 LD>= D	S1. S2.	5 steps	LD>=	Continuous Operation	9 steps	LDD>=	Continuous Operation

Operand number			
Operand number	S1.	Device number storing comparison data **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, Modify	BIN16/32-bit
	S2.	Device number storing comparison data **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, Modify	

Instruction Explanation

data comparison instructions connected to bus lines.

The contents of **S1.** are compared with the contents of **S2.** in the binary format, and a contact becomes conductive (ON) or non-conductive (OFF) depending on the comparison result.

16-bit instruction	32-bit instruction	ON condition	OFF condition
LD=	LDD=	**S1.＝S2.**	**S1.≠S2.**
LD>	LDD>	**S1.＞S2.**	**S1.≤S2.**
LD<	LDD<	**S1.＜S2.**	**S1.≥S2.**
LD<>	LDD<>	**S1.≠S2.**	**S1.＝S2.**
LD<=	LDD<=	**S1.≤S2.**	**S1.＞S2.**
LD>=	LDD>=	**S1.≥S2.**	**S1.＜S2.**

1. Negative value

When the most significant bit is "1" in the data stored in **S1.** or **S2.** , it is regarded as a negative value in comparison.

•In the 16-bit operation: bit 15

• In the 32-bit operation: bit 31

2. When using 32-bit counters (including 32-bit high speed counters)

Make sure to execute the 32-bit operation (such as "LDD=", "LDD>" and "LDD<") when comparing 32-bit counters (C200 to C255).

If a 32-bit counter is specified in the 16-bit operation (such as "LD=", "LD>" and "LD<"), a program error or operation error will occur.

3. Programming of data comparison instructions

When programming in GX Developer, symbols "≤" and "≥" cannot be input.

Separate "≤" into "<" and "=", and separate "≥" into ">" and "=" in input.

The input procedure is described below:

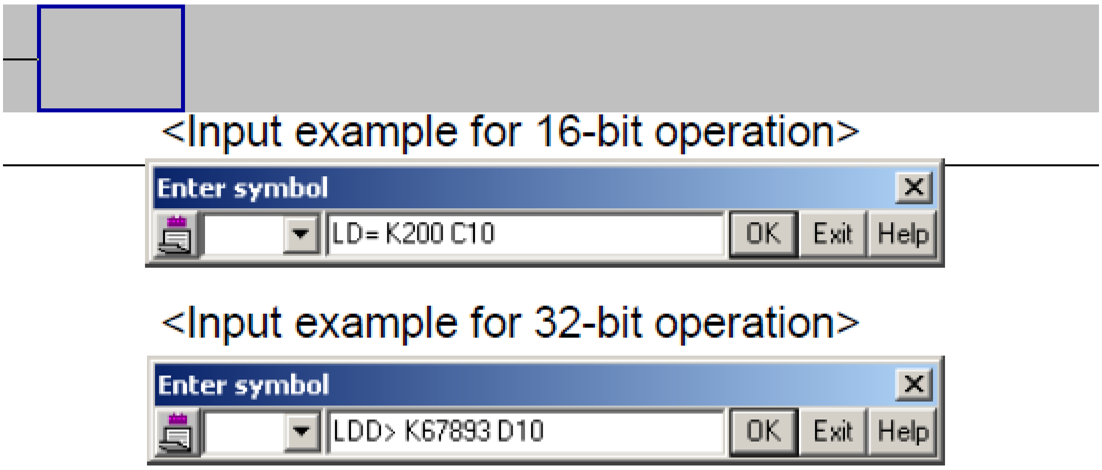
a) Display the circuit program edit window, and put the cursor in a position where a data comparison instruction is to be used.

b) Input "Instruction" → "space" → "value or device" → "space" → "value or device".

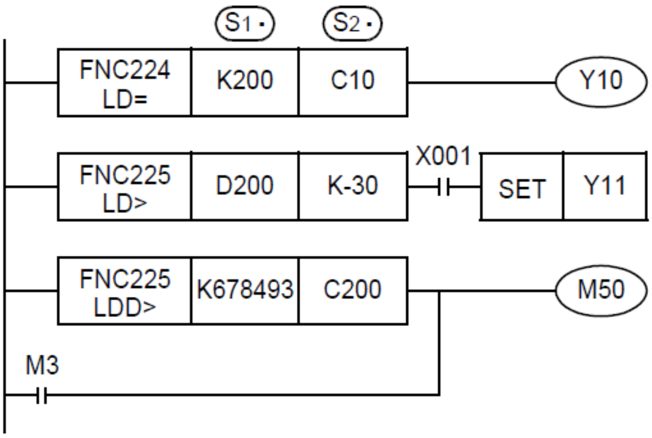
For an input example, refer to "Instruction input window in GX Developer" shown below.

c) Click the [OK] button.

d) Input other contacts and coil drive units consecutively.



Program Example



When the current value of the counter C10 is "200", Y010 is driven.

When the contents of D200 are "-29" or more and X001 is ON, Y011 is set.

When the contents of the counter C200 are less than "K678,493" or when M3 turns ON, M50 is driven.

25.2 AND=,>,<,<>,<=,>=/Data Comparison

These instructions compare numeric values, and set a contact to ON when the condition agrees.

Instruction		Operand Type	Function					
FNC232 AND=	D	S1. S2.	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
			5 steps	AND=	Continuous Operation	9 steps	ANDD=	Continuous Operation
FNC233 AND>	D	S1. S2.	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
			5 steps	AND>	Continuous Operation	9 steps	ANDD>	Continuous Operation
FNC234 AND<	D	S1. S2.	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
			5 steps	AND<	Continuous Operation	9 steps	ANDD<	Continuous Operation
FNC235 AND<>	D	S1. S2.	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
			5 steps	AND<>	Continuous Operation	9 steps	ANDD<>	Continuous Operation
FNC236 AND<=	D	S1. S2.	16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
			5 steps	AND<=	Continuous Operation	9 steps	ANDD<=	Continuous Operation
FNC237		S1.	16-bit	Mnemonic	Operation	32-bit	Mnemonic	Operation

D	AND>=	S2.	Instruction		Condition		Instruction		Condition
			5 steps	AND>=	Continuous Operation		9 steps	ANDD>=	Continuous Operation

Operand number	S1.	Device number storing comparison data **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, Modify	BIN16/32-bit
	S2.	Device number storing comparison data **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, Modify	

Instruction Explanation

Data comparison instructions connected to other contacts in series

The contents of **S1.** are compared with the contents of **S2.**in binary format, and a contact becomes conductive (ON) or non-conductive (OFF) depending on the comparison result.

16-bit instruction	32-bit instruction	ON condition	OFF condition
AND=	ANDD=	**S1. = S2.**	**S1.≠S2.**
AND>	ANDD>	**S1. > S2.**	**S1.≤S2.**
AND<	ANDD<	**S1. < S2.**	**S1.≥S2.**
AND<>	ANDD<>	**S1.≠S2.**	**S1. = S2.**
AND<=	ANDD<=	**S1.≤S2.**	**S1. > S2.**
AND>=	ANDD>=	**S1.≥S2.**	**S1. < S2.**

1. Negative value

When the most significant bit is "1" in the data stored in **S1.**or **S2.** , it is regarded as a negative value in comparison.

• In the 16-bit operation: bit 15

• In the 32-bit operation: bit 31

2. When using 32-bit counters (including 32-bit high speed counters)

Make sure to execute the 32-bit operation (such as "ANDD=", "ANDD>" and "ANDD<") when comparing 32- bit counters (C200 to C255).

If 16-bit operation (such as "AND=", "AND>" and "AND<") is specified , a program error or operation error will occur.

3. Programming of data comparison instructions

When programming in GX Developer, symbols "≤" and "≥" cannot be input.

Separate "≤" into "<" and "=", and separate "≥" into ">" and "=".

The input procedure is described below:

Operating procedure

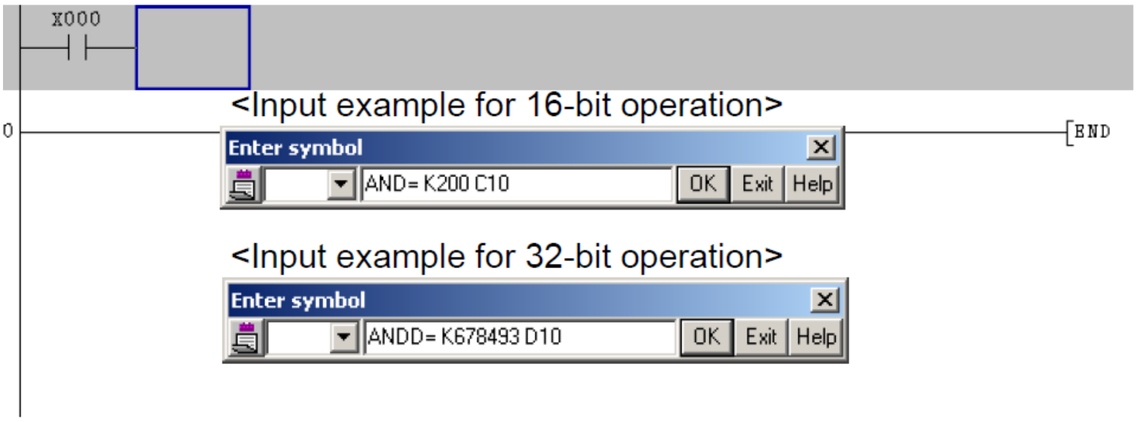
a) Display the circuit program edit window, and put the cursor in a position where a data comparison instruction is to be used.

b) Input "Instruction" → "space" → "value or device" → "space" → "value or device".

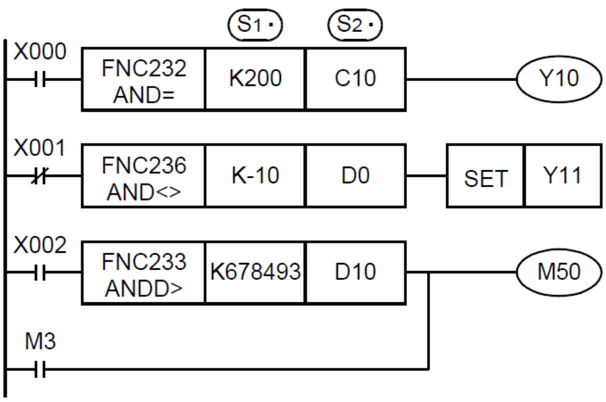
For an input example, refer to "Instruction input window in GX Developer" shown below.

c) Click the [OK] button.

d) Input other contacts and coil drive units consecutively.



Program Example 2



When X000 is ON and the current value of the counter C10 is "200" Y010 is driven.

When X001 is OFF and the contents of the data register D0 are not "-10", Y011 is set.

When X002 is ON, when the contents of the data registers D11 and D10 are less than "K678,493", or when M3 turns ON, M50 is driven.

25.3 OR=,>,<,<>,<=,>=/Data Comparison

These instructions compare numeric values, and set a contact to ON when the condition agrees.

Instruction	Operand Type	Function					
		16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
D **FNC240 OR=**	**S1. S2.**	5 steps	OR=	Continuous Operation	9 steps	ORD=	Continuous Operation
D **FNC241 OR>**	**S1. S2.**	5 steps	OR>	Continuous Operation	9 steps	ORD>	Continuous Operation
D **FNC242 OR<**	**S1. S2.**	5 steps	OR<	Continuous Operation	9 steps	ORD<	Continuous Operation
D **FNC244 OR<>**	**S1. S2.**	5 steps	OR<>	Continuous Operation	9 steps	ORD<>	Continuous Operation
D **FNC245 OR<=**	**S1. S2.**	5 steps	OR<=	Continuous Operation	9 steps	ORD<=	Continuous Operation
FNC246	**S1.**	16-bit	Mnemonic	Operation	32-bit	Mnemonic	Operation

OR>=		S2.	Instruction		Condition		Instruction		Condition
D			5 steps	OR>=	Continuous Operation		9 steps	ORD>=	Continuous Operation

Operand number	S1.	Device number storing comparison data **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, Modify	BIN16/32-bit
	S2.	Device number storing comparison data **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, Modify	

Instruction Explanation

Data comparison instructions connected to other contacts in parallel.

The contents of **S1.** are compared with the contents of **S2.** in binary format, and a contact becomes conductive (ON) or non-conductive (OFF) depending on the comparison result.

16-bit instruction	32-bit instruction	ON condition	OFF condition
OR=	ORD=	**S1. = S2.**	**S1.≠S2.**
OR>	ORD>	**S1. > S2.**	**S1.≤S2.**
OR<	ORD<	**S1. < S2.**	**S1.≥S2.**
OR<>	ORD<>	**S1.≠S2.**	**S1. = S2.**
OR<=	ORD<=	**S1.≤S2.**	**S1. > S2.**
OR>=	ORD>=	**S1.≥S2.**	**S1. < S2.**

1. Negative value

When the most significant bit is "1" in the data stored in **S1.** or **S2.** , it is regarded as a negative value in comparison.

• In the 16-bit operation: bit 15

• In the 32-bit operation: bit 31

2. When using 32-bit counters (including 32-bit high speed counters)

Make sure to execute the 32-bit operation (such as "ORD=", "ORD>" and "ORD<") when comparing 32-bit counters (C200 to C255).

If a 32-bit counter is specified in the 16-bit operation (such as "ORD=", "OR>" and "OR<"), a program error or operation error will occur.

3. Programming of data comparison instructions

When programming in GX Developer, symbols "≤" and "≥" cannot be input.

Separate "≤" into "<" and "=", and separate "≥" into ">" and "=".

The input procedure is described below:

Operating procedure

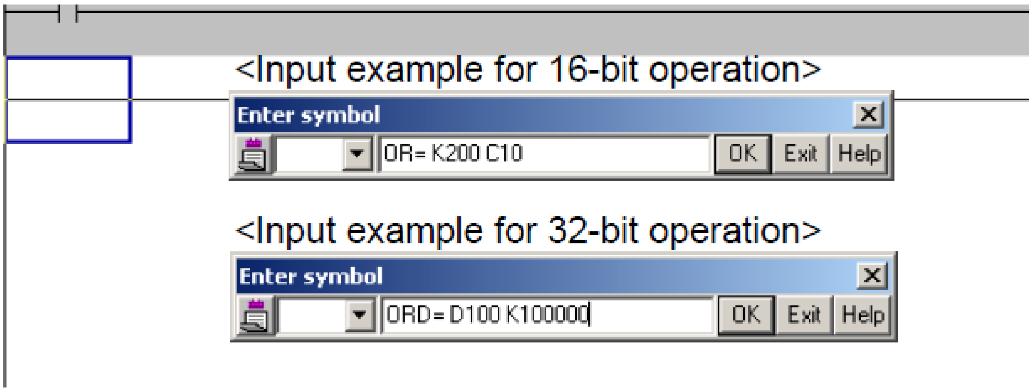
a) Display the circuit program edit window, and put the cursor in a position where a data comparison instruction is to be used.

b) Input "Instruction" → "space" → "value or device" → "space" → "value or device".

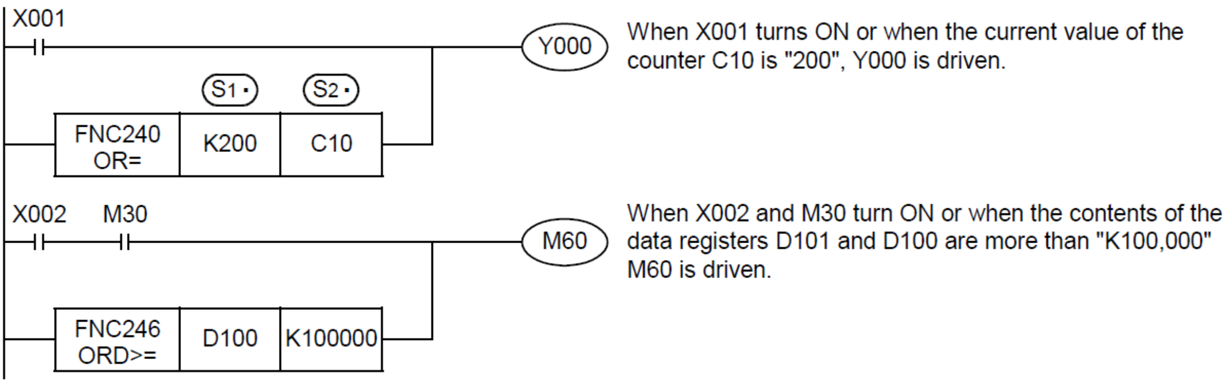
For an input example, refer to "Instruction input window in GX Developer" shown below.

c) Click the [OK] button.

d) Input other contacts and coil drive units consecutively.



Program Example



26 Data table operation

FNC NO.	Instruction	Function	Supported PLC series		
			3G PLC	2N PLC	MX2N PLC
250	—				
251	—				
252	—				
253	—				
254	—				
225	—				
256	LIMIT	Limit Control	★		
257	BAND	Dead Band Control	★		
258	ZONE	Zone Control	★		
259	SCL	Scaling (Coordinate by Point Data)	★		
260	DABIN	Decimal ASCII to BIN Conversion 换	★		
261	BINDA	BIN to Decimal ASCII Conversion	★		
262	—				
263	—				
264	—				
265	—				
266	—				
267	—				
268	—				
269	SCL2	Scaling 2 (Coordinate by X/Y Data)	★		

26.1 LIMIT/Limit Control

This instruction provides the upper limit value and lower limit value for an input numeric value, and controls the output value using these limit values.

Instruction	Operand Type	Function						
D FNC256 LIMIT P	S1. S2. S3. D.	16-bit Instruction	Mnemonic	Operation Condition		32-bit Instruction	Mnemonic	Operation Condition
		9 steps	LIMIT	Continuous Operation		17 steps	DLIMIT	Continuous Operation
			LIMITP	Pulse (Single) Operation			DLIMITP	Pulse (Single) Operation

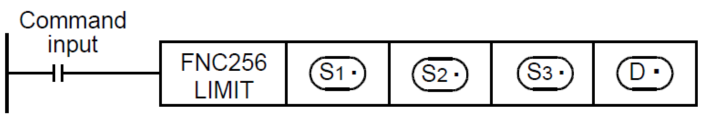
Operand number	S1.	Lower limit value (minimum output value) **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, K, H, Modify	BIN16/32 bit
	S2.	Upper limit value (maximum output value) **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, K, H, Modify	
	S3.	Input value controlled by the upper and lower limit values **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, Modify	
	D.	Head device number storing the output value controlled by the upper and lower limit values **Applicable devices:** KnY, KnM, KnS, T, C, D, R, K, H, Modify	

Instruction Explanation

1. 16-bit operation(LIMIT,LIMITP)

Depending on how the input value (16-bit binary value) specified by **S3.** compares to the range between **S1.** and **S2.**, the output value **D.** is controlled.

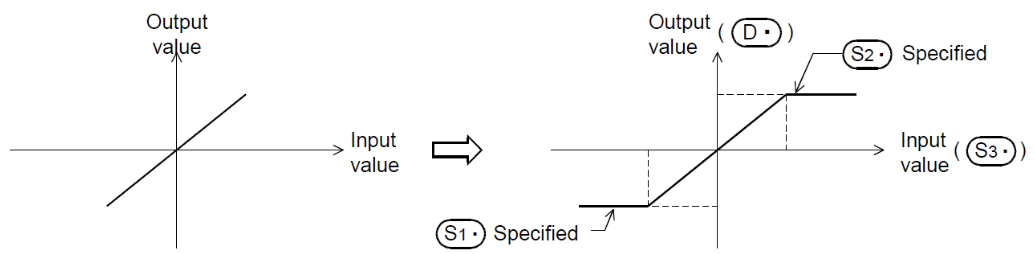
The output value is controlled as shown below:



·In the case of " S1· Lower limit value > S3· Input value" .... S1· Lower limit value → D· Output value

·In the case of " S2· Upper limit value < S3· Input value" .... S2· Upper limit value → D· Output value

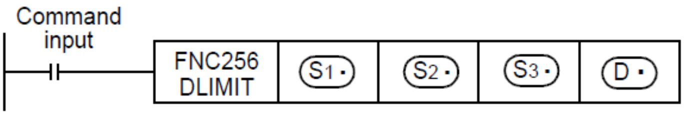
·In the case of " S1· Lower limit value ≤ S3· Input value ≤ S2· Upper limit value" .... S3· Input value → D· Output value



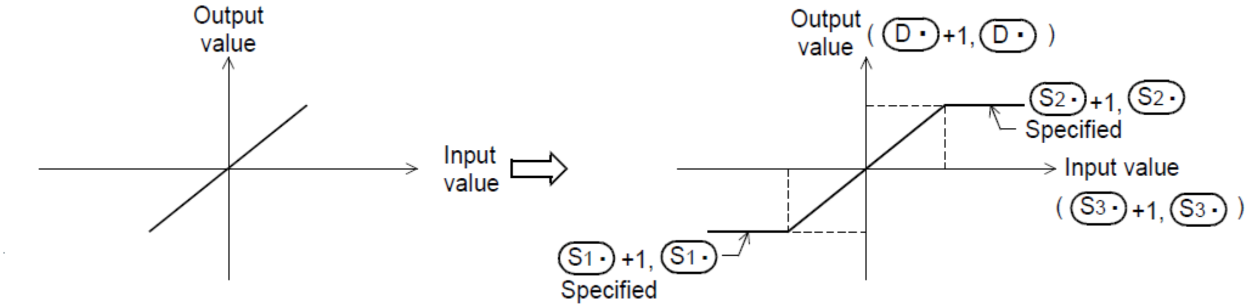
- When controlling the output value using only the upper limit value, set "-32768" to the lower limit value specified in S1· .
- When controlling the output value using only the lower limit value, set "32767" to the upper limit value specified in S2· .

2. 32-bit operation(DLIMIT,DLIMITP)

Depending on how the input value (32-bit binary value) specified by **[S3.+1, S3.]** compares to the range between **[S1.+1, S1.]** and **[S2.+1, S2.]**, the output value **[D.+1, D.]** is controlled.



- In the case of "Lower limit value ([S1·]+1, [S1·]) > Input value ([S3·]+1, [S3·])" .... Lower limit value ([S1·]+1, [S1·]) → Output value ([D·]+1, [D·])
- In the case of "Upper limit value ([S2·]+1, [S2·]) < Input value ([S3·]+1, [S3·])" .... Upper limit value ([S2·]+1, [S2·]) → Output value ([D·]+1, [D·])
- In the case of "Lower limit value ([S1·]+1, [S1·]) ≤ Input value ([S3·]+1, [S3·]) ≤ Upper limit value ([S2·]+1, [S2·])" .... Input value ([S3·]+1, [S3·]) → Output value ([D·]+1, [D·])



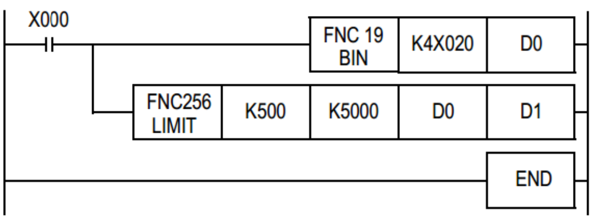
- When controlling the output value using only the upper limit value, set "-2,147,483,648" to the lower limit value specified in [S1·+1, S1·].
- When controlling the output value using only the lower limit value, set "2,147,483,647" to the upper limit value specified in [S2·+1, S2·].

- An operation error is caused when the instruction is executed in the setting status shown below; The error flag M8067 turns ON, and the error code (K6706) is stored in D8067.

  —16-bit operation, **S1.>S2.**

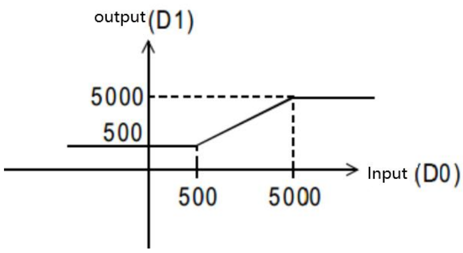
  —32-bit operation, **[S1.+1, S1.]>[S2.+1, S2.]**

Program Example



◆ X0=ON, the BCD data set in X020 to X037 is controlled by the limit values 500～5000,and the controlled value is stored in D1.

- D0 ＜ 500, D1=500.
- 500 ≦ D0 ≦ 5000, D1=D0.
- 5000 ＜ D0, D1=5000.



26.2 BAND/Dead Band Control

This instruction provides the upper limit value and lower limit value.

Instruction	Operand Type	Function						
D / **FNC257 BAND** / P	**S1. S2. S3. D.**	16-bit Instruction	Mnemonic	Operation Condition		32bit Instruction	Mnemonic	Operation Condition
		9 steps	BAND	Continuous Operation		17 steps	DBAND	Continuous Operation
			BANDP	Pulse (Single) Operation			DBANDP	Pulse (Single) Operation

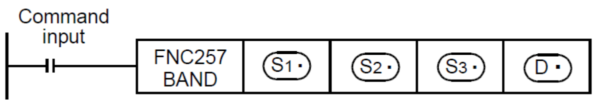
Operand number			
Operand number	**S1.**	Lower limit value of the dead band (no-output band) **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, K, H, Modify	BIN16/32 bit
	S2.	Upper limit value of the dead band (no-output band) **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, K, H, Modify	
	S3.	Input value controlled by the dead band **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, Modify	
	D.	Device number storing the output value controlled by the dead band **Applicable devices:** KnY, KnM, KnS, T, C, D, R, K, H, Modify	

Instruction Explanation

1. 16-bit operation(BAND,BANDP)

Depending on how the input value (16-bit binary value) specified by **S3.** compares to the dead band range between **S1.** and **S2.**, the output value **D.** is controlled.

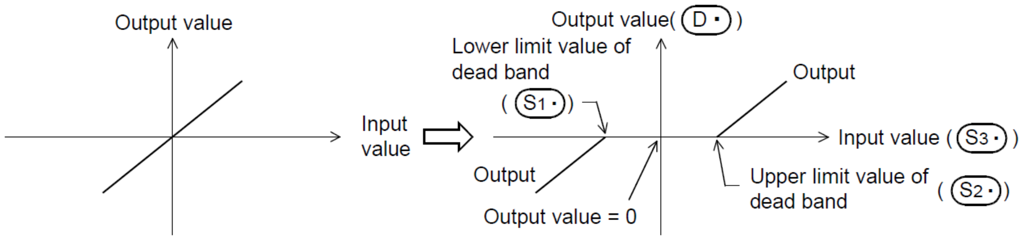
The output value is controlled as shown below:



•In the case of " S1· Lower limit value > S3· Input value" .... S3· Input value – S1· Lower limit value → D· Output value

•In the case of " S2· Upper limit value < S3· Input value" .... S3· Input value – S2· Upper limit value → D· Output value

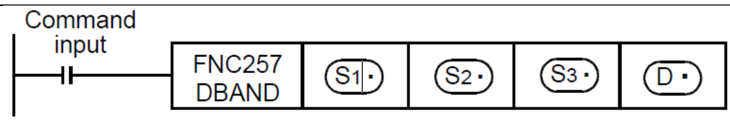
•In the case of " S1· Lower limit value ≤ S3· Input value ≤ S2· Upper limit value" ....0 → D· Output value



2. 32-bit operation(DBAND,DBANDP)

Depending on how the input value (32-bit binary value) specified by **[S3.+1, S3.]** compares to the dead band range between **[S1.+1, S1.]** and **[S2.+1, S2.]**, the output value **[D.+1, D.]** is controlled.

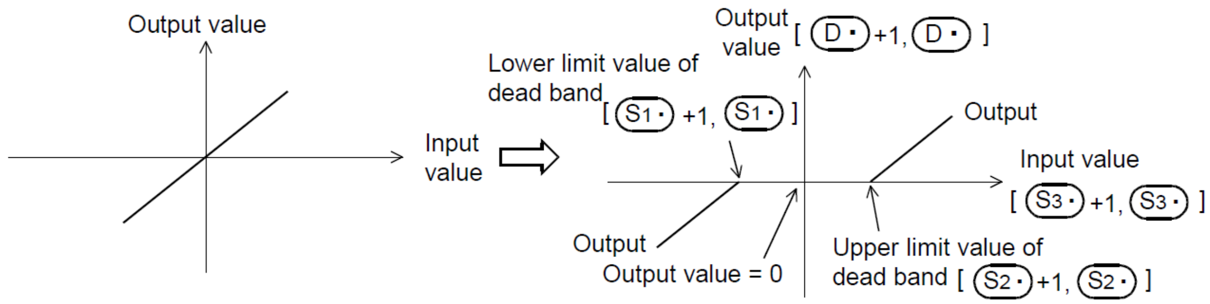
The output value is controlled as shown below:



S1· +1, S1·   S3· +1, S3·   S3· +1, S3·   S1· +1, S1·   D· +1, D·
•In the case of "Lower limit value > Input value" ....Input value – Lower limit value → Output value

S2· +1, S2·   S3· +1, S3·   S3· +1, S3·   S2· +1, S2·   D· +1, D·
•In the case of "Upper limit value < Input value" ....Input value – Upper limit value → Output value

S1· +1, S1·   S3· +1, S3·   S2· +1, S2·   D· +1, D·
•In the case of "Lower limit value ≤ Input value ≤ Upper limit value" .... 0 → Output value



- When the output value overflows, it is handled as follows

  - 16-bit operation

  The output value is a 16-bit binary value with sign. Accordingly, if the operation result is outside the range from -32768 to +32767, it is handled as follows:

  Lower limit value of dead band **S1.**=10
  Input value **S3.** =-32,768
  ⇨
  Output value= -32, 768 - 10
  =8000H-AH
  =7FF6H
  =32,758

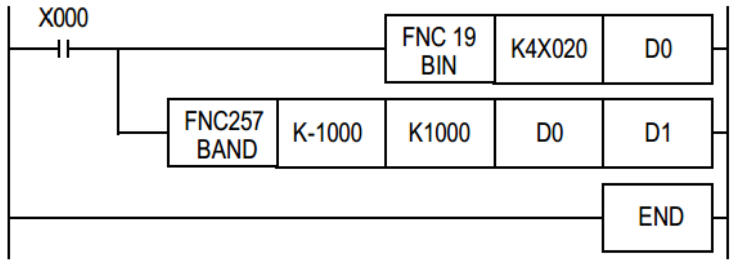
  - 32-bit operation

  The output value is a 32-bit binary value with sign. Accordingly, if the operation result is outside the range from -2,147,483,648 to +2,147,483,647, it is handled as follows:

  Lower limit value of dead band **[S1.+1, S1.]**=1000
  Input value **[S3.+1, S3.]** =-2,147,483,648
  ⇨
  Output value= -2,147,483,648 - 10
  =80000000H-000003E8H
  =7FFFFC18H
  =2,147,483,648

- An operation error is caused when the instruction is executed in the setting status shown below; The error flag M8067 turns ON, and the error code (K6706) is stored in D8067.
  —16-bit operation, **S1.>S2.**
  —32-bit operation, **[S1.+1, S1.]>[S2.+1, S2.]**

Program Example

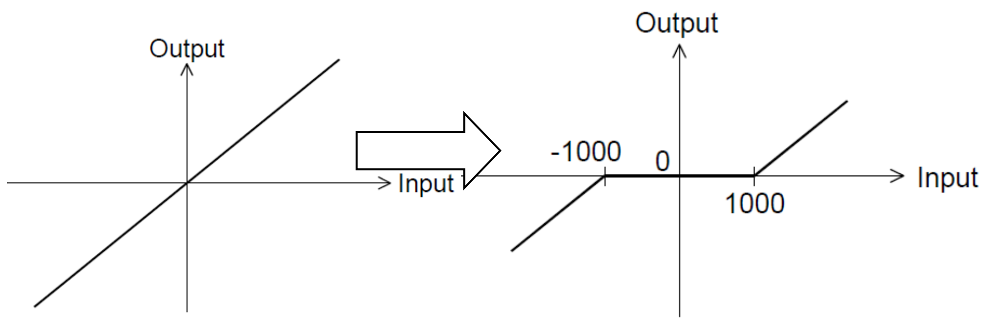


◆ X000=ON, the BCD data set in X020 to X037 is controlled by the dead band from “-1000” to “+1000”, and a controlled value is output to D1

• D0＜(-1000),“D0 - (-1000)” is output to D1.

• -1000≦D0≦1000,“0” is output to D1.

• 1000＜D0,“D0 - 1000” is output to D1.



26.3 ZONE/Zone Control

Depending on how the input value compares to positive or negative, the output value is controlled by the bias value specified.

Instruction	Operand Type	Function					
		16-bit Instruction	Mnemonic	Operation Condition	32-bit Instruction	Mnemonic	Operation Condition
D **FNC258 ZONE** P	**S1.** **S2.** **S3.** **D.**	9 steps	ZONE	Continuous Operation	17 steps	DZONE	Continuous Operation
			ZONEP	Pulse (Single) Operation		DZONEP	Pulse (Single) Operation

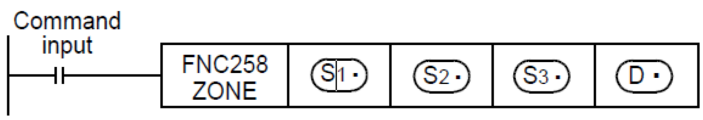
Operand number			
Operand number	**S1.**	Negative bias value to be added to the input value **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, K, H, Modify	BIN16/32 bit
	S2.	Positive bias value to be added to the input value **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, K, H, Modify	
	S3.	Input value controlled by the zone **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, Modify	
	D.	Head device number storing the output value controlled by the zone **Applicable devices:** KnY, KnM, KnS, T, C, D, R, K, H, Modify	

Instruction Explanation

1. 16-bit operation(ZONE,ZONEP)

The bias value specified by **S1.** or **S2.** is added to the input value specified by **S3.**, and output to the device specified by **D.**.

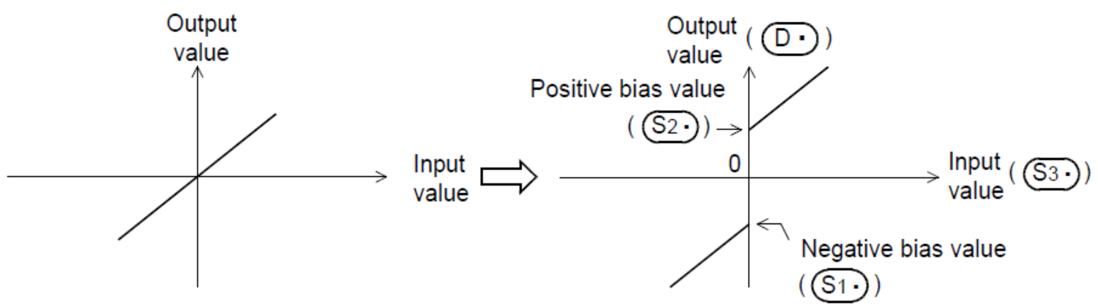
The bias value is added as shown below:



·In the case of "S3· Input value < 0" ..... S3· Input value + S1· Negative bias value → D· Output value

·In the case of "S3· Input value = 0" ....0 → D· Output value

·In the case of "S3· Input value > 0" ..... S3· Input value + S2· Positive bias value → D· Output value

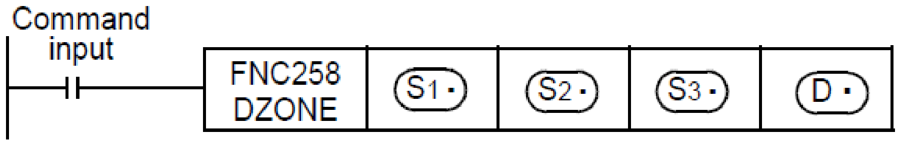


2. **32-bit operation(DZONE,DZONEP)**

The bias value specified by **[S1.+1, S1.]** or **[S2.+1, S2.]** is added to the input value specified by

[S3.+1, S3.], and output to the device specified by **[D.+1, D.]**.

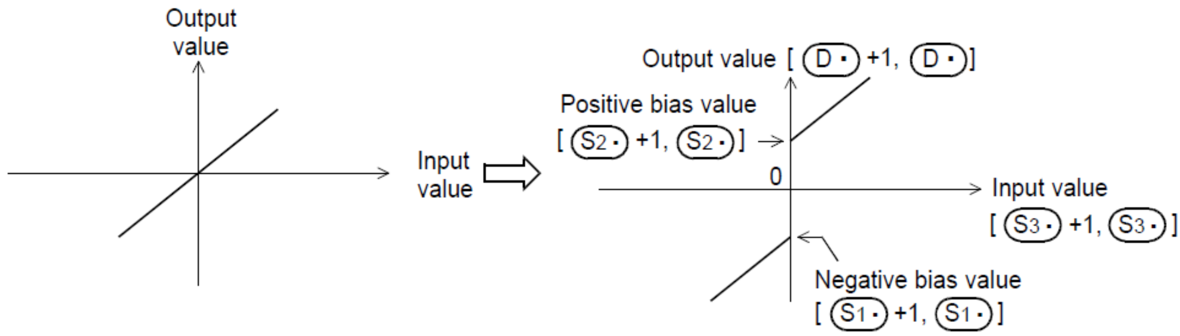
The bias value is added as shown below:



S3·+1, S3·   S3·+1, S3·   S1·+1, S1·   D·+1, D·
•In the case of "Input value < 0" ....Input value + Negative bias value → Output value

S3·+1, S3·   D·+1, D·
•In the case of "Input value = 0" ....0 → Output value

S3·+1, S3·   S3·+1, S3·   S2·+1, S2·   D·+1, D·
•In the case of "Input value > 0" ....Input value + Positive bias value → Output value



- When the output value overflows, it is handled as follows:
  - 16-bit operation

    The operation result is a 16-bit binary value with sign. Accordingly, if the output value is outside the range from -32768 to +32767, it is handled as follows:

Negative bias value **S1.**=-100
Input value **S3.** =-32,768

Output value = -32, 768 +(-100)
=8000H+FF9CH
=7F9CH
=32,668

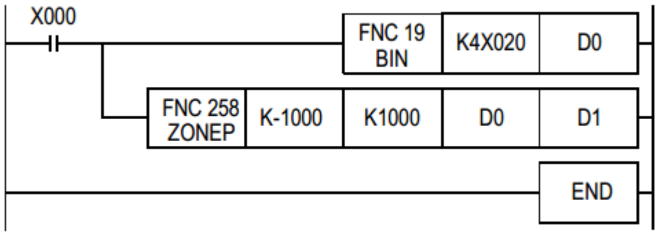
  - 32-bit operation

    The output value is a 32-bit binary value with sign. Accordingly, if the operation result is outside the range from -2,147,483,648 to +2,147,483,647, it is handled as follows:

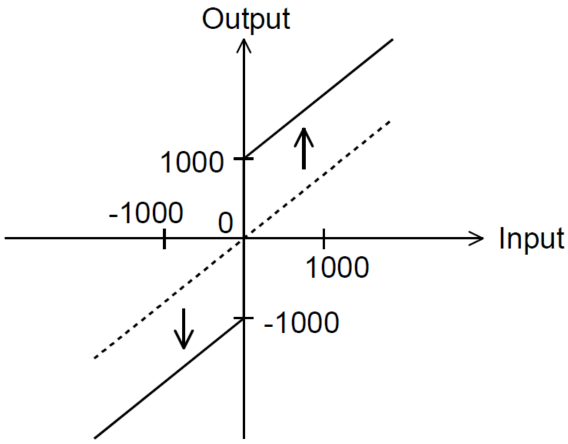
Negative bias value **[S1.+1, S1.]**=-1000
Input value **[S3.+1, S3.]** =-2,147,483,648

Output value= -2,147,483,648 +(-1000)
=80000000H-FFFFFC18H
=7FFFFC18
=2,147,483,648

Program Example



- X000=ON, the BCD data set in X020 to X037 is controlled by the zone from “-1000” to “+1000”, and the controlled value is output to D1
  - D0<0,“D0 + (-1000)” is output to D1.
  - D0=0,“0” is output to D1.
  - 0<D0,“D0 + 1000” is output to D1.



26.4 SCL/Scaling (Coordinate by Point Data)

This instruction executes scaling of the input value using a specified data table, and outputs the result.

SCL2 (FNC269) is also available with a different data table configuration for scaling.

Instruction	Operand Type	Function						
D FNC259 SCL P	S1. S2. D.	16-bit Instruction	Mnemonic	Operation Condition		33-bit Instruction	Mnemonic	Operation Condition
		7 steps	SCL	Continuous Operation		13 steps	DSCL	Continuous Operation
			SCLP	Pulse (Single) Operation			DSCLP	Pulse (Single) Operation

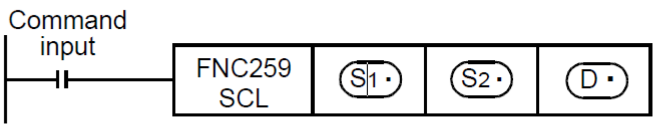
Operand number			
Operand number	S1.	Input value used in scaling or device number storing the input value **Applicable devices:** KnX, KnY, KnM, KnS, T, C, D, R, K, H, Modify	BIN16/32 bit
	S2.	Head device number storing the conversion table used in scaling **Applicable devices:** D, R, Modify	
	D.	Device number storing the output value controlled by scaling **Applicable devices:** KnY, KnM, KnS, T, C, D, R, Modify	

Instruction Explanation

1. 16-bit operation(SCL,SCLP)

The input value specified in **S1.** is processed by scaling for the specified conversion characteristics, and stored to a device number specified in **D.**. Conversion for scaling is executed based on the data table stored in a device specified in **S2.** and later.

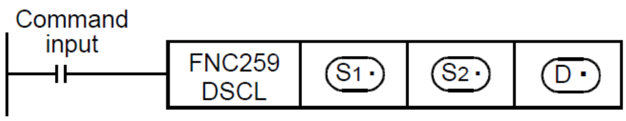
If the output data is not an integer, however, the number in the first decimal place is rounded.



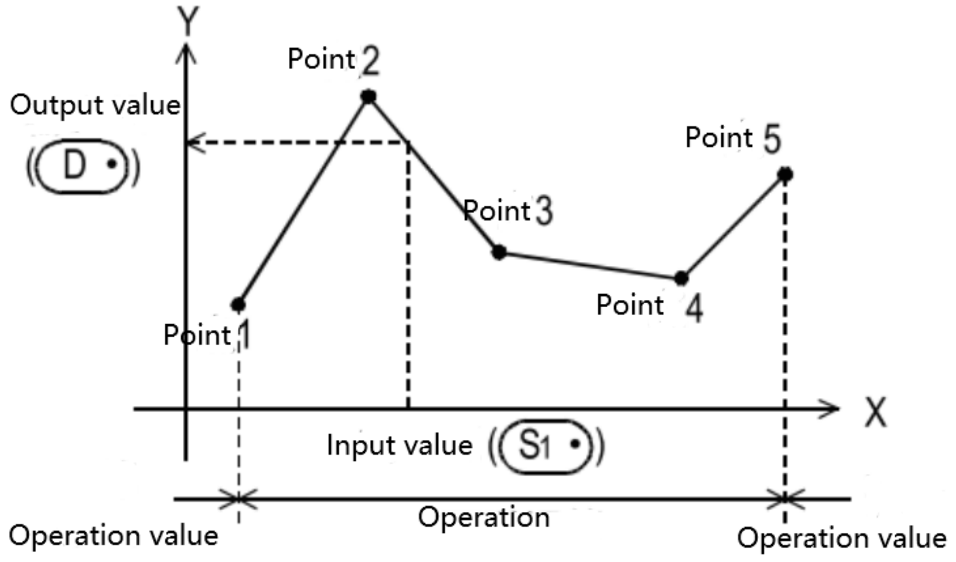
2. 32-bit operation(DSCL,DSCLP)

The input value specified in **[S1.+1, S1.]** is processed by scaling for the specified conversion characteristics, and stored to a device number specified in **[D.+1, D.]**. Conversion for scaling is executed based on the data table stored in a device specified in **[S2.+1, S2.]** and later.

If the output data is not an integer, however, the number in the first decimal place is rounded.



3. Setting the conversion table for scaling



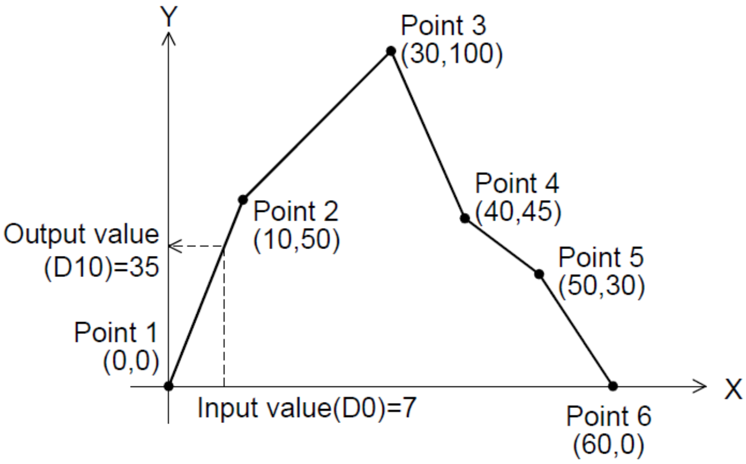
Set item		Device assignment in setting data table	
		16-bit operation	32-bit operation
Number of coordinate points (As above picture, it is“5”)		**S2.**	**[S2.+1, S2.]**
Point 1	X coordinate	**S2.+1**	**[S2.+3, S2.+2]**
	Y coordinate	**S2.+2**	**[S2.+5, S2.+4]**
Point 2	X coordinate	**S2.+3**	**[S2.+7, S2.+6]**
	Y coordinate	**S2.+4**	**[S2.+9, S2.+8]**
Point 3	X coordinate	**S2.+5**	**[S2.+11, S2.+10]**
	Y coordinate	**S2.+6**	**[S2.+13, S2.+12]**
Point 4	X coordinate	**S2.+7**	**[S2.+15, S2.+14]**
	Y coordinate	**S2.+8**	**[S2.+17, S2.+16]**
Point 5	X coordinate	**S2.+9**	**[S2.+19, S2.+18]**
	Y coordinate	**S2.+10**	**[S2.+21, S2.+20]**
...	...	**...**	**...**
Point n (last)	X coordinate	**S2.+2n-1**	**[S2.+4n-1、S2.+4n-2]**
	Y coordinate	**S2.+2n**	**[S2.+4n+1、S2.+4n]**

- An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067.
  • When the Xn data is not set in the ascending order in the data table (error code: K6706)
  The data table is searched from the low-order side of device numbers in the data table in the operation. Accordingly, even if only some Xn data is set in the ascending order in the data table, the instruction is executed without operation error up to the area of the data table in which the Xn data is set in the ascending order.
  • When **S1.** is outside the data table (error code: K6706)
  • When the value exceeds the 32-bit data range in the middle of operation (error code: K6706)
  In this case, check whether the distance between points is not “65535” or more.

	If the distance is “65535” or more, reduce the distance between points.

Program Example

◆ the value input to D0 is processed by scaling based on the conversion table for scaling set in R0 and later, and output to D10.



Set item		Device	Setting contents
Number of coordinate points		R0	K6
Point 1	X coordinate	R1	K0
	Y coordinate	R2	K0
Point 2	X coordinate	R3	K10
	Y coordinate	R4	K50
Point 3	X coordinate	R5	K30
	Y coordinate	R6	K100
Point 4	X coordinate	R7	K40
	Y coordinate	R8	K45
Point 5	X coordinate	R9	K50
	Y coordinate	R10	K30
Point 6	X coordinate	R11	K60
	Y coordinate	R12	K0

26.5 DABIN/Decimal ASCII to BIN Conversion

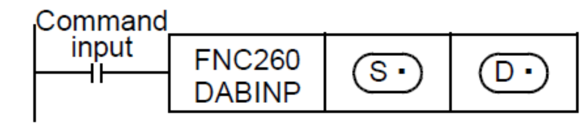
This instruction converts numeric data expressed in decimal ASCII codes (30H to 39H) into binary data.

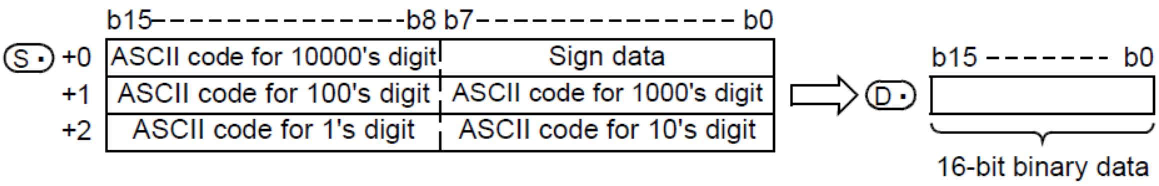
<table>
<tr><th colspan="3">Instruction</th><th>Operand Type</th><th colspan="7">Function</th></tr>
<tr><td rowspan="3">D</td><td rowspan="3">FNC260
DABIN</td><td rowspan="3">P</td><td rowspan="3">S
D.</td><td>16-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td><td></td><td>32-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td></tr>
<tr><td rowspan="2">5 steps</td><td>DABIN</td><td>Continuous Operation</td><td rowspan="2"></td><td rowspan="2">9steps</td><td>DDABIN</td><td>Continuous Operation</td></tr>
<tr><td>DABINP</td><td>Pulse (Single) Operation</td><td>DDABINP</td><td>Pulse (Single) Operation</td></tr>
<tr><td colspan="3" rowspan="2">Operand number</td><td>S.</td><td colspan="6">Head device number storing data (ASCII codes) to be converted into binary data
Applicable devices: T, C, D, R, Modify</td><td>Character string</td></tr>
<tr><td>D.</td><td colspan="6">Device number storing conversion result
Applicable devices: KnY, KnM, KnS, T, C, D, R, V, Z, Modify</td><td>BIN16/32 bit</td></tr>
</table>

<table>
<tr><td>Instruction Explanation</td><td>

1. 16-bit operation(DABIN,DABINP)

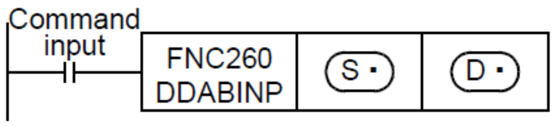
Data stored in **S.~S.+2** expressed in decimal ASCII codes (30H to 39H) is converted into 16-bit binary data, and stored in **D.**.

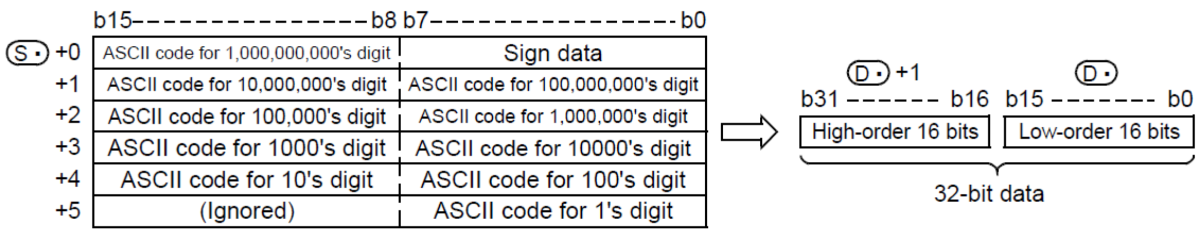




2. 32-bit operation(DDABIN,DDABINP)

Data stored in **S.~S.+5**5 expressed in decimal ASCII codes (30H to 39H) is converted into 32-bit binary data, and stored in **[D.+1, D.]**.





3. Data explanation

1) 16-bit operation, numeric range of data stored in **S.~S.+2** is -32768 ~ 32767 ;

</td></tr>
</table>

32-bit operation, numeric range of data stored in **S.~S.+5** is -2,147,483,648 ~ 2, 147,483,647.The high-order byte of +5 is ignored.

2) As "sign data" (low-order byte of), "20H (space)" is set when the data to be converted is positive, and "2DH (-)" is set when the data to be converted is negative.

3) An ASCII code for each digit is within the range from 30H to 39H.

4) When an ASCII code for each digit is "20H (space)" or "00H (NULL)", it is handled as "30H".

- An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067.
  - When the sign data stored in **S.** (low byte) is any value other than "20H (space)" or "2DH (-)" (error code: K6706)
  - When an ASCII code for each digit stored in **S.~S.+2(5)** is any value other than "30H" to "39H","20H (space)", or "00H (NULL)" (error code: K6706)
  - When the numeric range of **S.~S.+2(5)** exceeds the device range (error code: K6706)

Program Example

◆ X0=ON, the sign and decimal ASCII codes in five digits stored in D20 to D22 are converted into a binary value and stored in D0

	b15---b8	b7---b0
D20	20H(space)	2DH(-)
D21	32H(2)	20H(space)
D22	36H(6)	37H(7)

(regarded as -00276)

D0
-276

BIN value

26.6 BINDA/BIN to Decimal ASCII Conversion

This instruction converts binary data into decimal ASCII codes (30H to 39H).

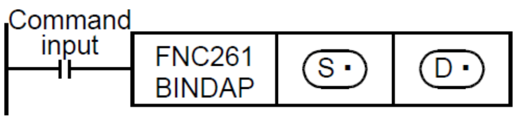
<table>
<tr><th>Instruction</th><th>Operand Type</th><th colspan="6">Function</th></tr>
<tr><td rowspan="4">D
FNC261
BINDA
P</td><td rowspan="4">S.
D.</td><td>16-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td><td>32-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td></tr>
<tr><td rowspan="2">5 steps</td><td>BINDA</td><td>Continuous Operation</td><td rowspan="2">9 steps</td><td>DBINDA</td><td>Continuous Operation</td></tr>
<tr><td>BINDAP</td><td>Pulse (Single) Operation</td><td>DBINDAP</td><td>Pulse (Single) Operation</td></tr>
</table>

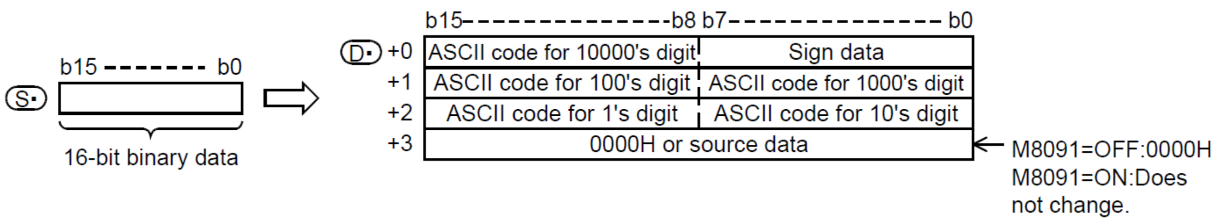
<table>
<tr><td rowspan="2">Operand number</td><td>S.</td><td>Device number storing binary data to be converted into ASCII codes
Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, V, Z, K, H, Modify</td><td>Character string</td></tr>
<tr><td>D.</td><td>Head device number storing conversion result
Applicable devices: T, C, D, R, Modify</td><td>BIN16/32 bit</td></tr>
</table>

Instruction Explanation

1. 16-bit operation(BINDA,BINDAP)

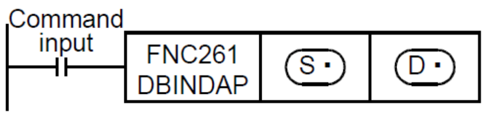
Each digit of 16-bit binary data stored in **S.** is converted into an ASCII code (30H to 39H), and stored in **D.** and later.

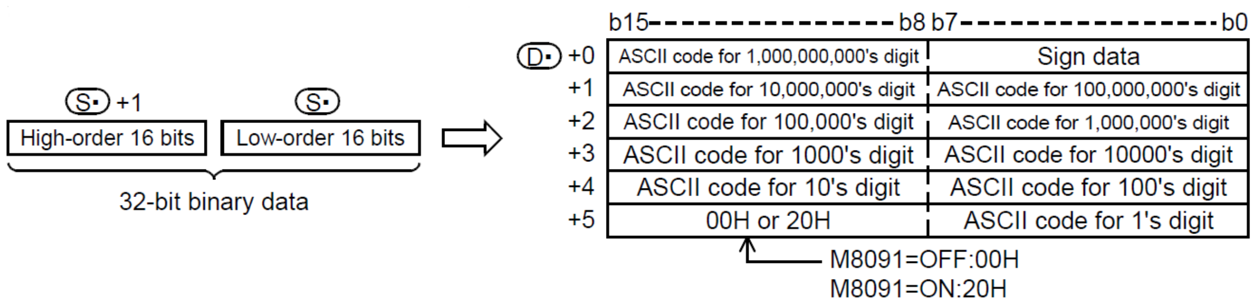




2. 32-bit operation(DBINDA,DBINDAP)

Each digit of 32-bit binary data stored in **[S.+1, S.]** is converted into an ASCII code (30H to 39H),and stored in **D.** and later.





3. Data explanation

1) The numeric range of 16-bit binary data stored in is from -32768 to +32767;

The numeric range of 32-bit binary data stored in **[S.+1, S.]** is from -2,147,483,648 to +2,147,483,647. **S.+5** high-order byte is ignored.

2) The conversion result stored in D. is as follows:

a) "sign data" (low-order byte of **D.**) "20H (space)" is set when the 16-bit binary data stored is positive, and "2DH (-)" is set when 16-bit binary data is negative.

b) "20H (space)" is stored for "0" on the left side of the effective digits (zero suppression).

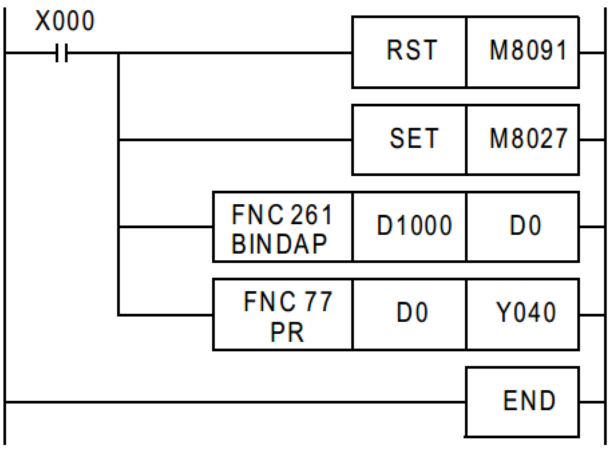
c) The high-order byte of **D.+3/D.+5** is set as follows depending on the ON/OFF status of M8091.

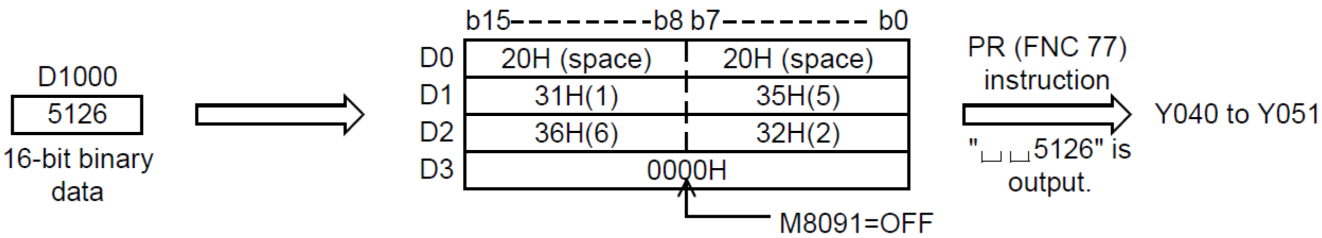
ON/OFF status	Contents of processing
M8091=OFF	The high-order byte of **D.+3/D.+5** is set to "00H (NULL)."
M8091=ON	The high-order byte of **D.+3/D.+5** is set to "20H (space)."

● An operation error is caused in the following case; The error flag M8067 turns ON, and the error code is stored in D8067

• When the occupied device point of storing the ASCII code character string exceeds the corresponding device rang (error code: K6706).

Program Example

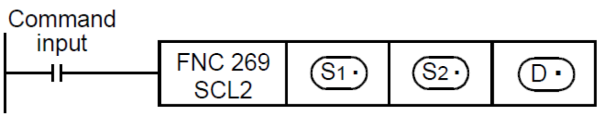


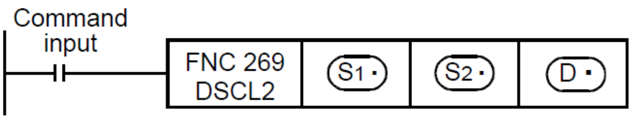


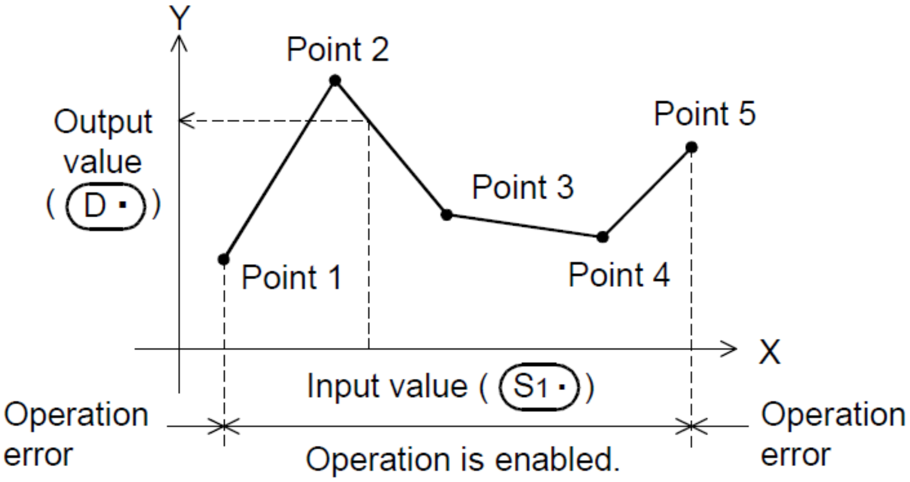
◆ X000=ON, 16-bit binary data stored in D1000 is converted into decimal ASCII codes.and the ASCII codes converted by PR (FNC 77) instruction are output one by one in the time division method to Y040 to Y051. By setting to OFF the output character selector signal M8091 and setting PR mode flag M8027 ON, ASCII codes up to "00H" are output.

26.7 SCL2/Scaling 2 (Coordinate by X/Y Data)

This instruction executes scaling of the input value using a specified data table, and outputs the result.
SCL (FNC259) is also available with a different data table configuration for scaling.

<table>
<tr><th colspan="3">Instruction</th><th>Operand Type</th><th colspan="7">Function</th></tr>
<tr><td rowspan="3">D</td><td rowspan="3">FNC269
SCL2</td><td rowspan="3">P</td><td rowspan="3">S1.
S2.
D.</td><td>16-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td><td rowspan="3"></td><td>32-bit Instruction</td><td>Mnemonic</td><td>Operation Condition</td></tr>
<tr><td>7 steps</td><td>SCL2</td><td>Continuous Operation</td><td>13 steps</td><td>DSCL2</td><td>Continuous Operation</td></tr>
<tr><td></td><td>SCL2P</td><td>Pulse (Single) Operation</td><td></td><td>DSCL2P</td><td>Pulse (Single) Operation</td></tr>
<tr><td colspan="3" rowspan="3">Operand number</td><td>S1.</td><td colspan="5">Input value used in scaling or device number storing the input value
Applicable devices: KnX, KnY, KnM, KnS, T, C, D, R, K, H, Modify</td><td colspan="2" rowspan="3">BIN16/32 bit</td></tr>
<tr><td>S2.</td><td colspan="5">Head device number storing the conversion table used in scaling
Applicable devices: D, R, Modify</td></tr>
<tr><td>D.</td><td colspan="5">Device number storing the output value controlled by scaling
Applicable devices: KnY, KnM, KnS, T, C, D, R, Modify</td></tr>
<tr><td colspan="3">Instruction Explanation</td><td colspan="8">1. 16-bit operation(SCL2,SCL2P)
The input value specified in S1. is processed by scaling for the specified conversion characteristics, and stored to a device number specified in D.. Conversion for scaling is executed based on the data table stored in a device specified in S2. and later.
If the output data is not an integer, however, the number in the first decimal place is rounded.


2. 32-bit operation(DSCL2,DSCL2P)
The input value specified in [S1.+1, S1.]is processed by scaling for the specified conversion characteristics, and stored to a device number specified in [D.+1, D.]. Conversion for scaling is executed based on the data table stored in a device specified in [S2.+1, S2.]and later.
If the output data is not an integer, however, the number in the first decimal place is rounded.

</td></tr>
</table>



3. Setting the conversion table for scaling

The conversion table for scaling is set based on the data table stored in a device specified in **[S2.+1, S2.]** and later.

The data table has the following configuration:

Set item		Device assignment in setting data table	
		16-bit operation	32-bit operation
Number of coordinate points (as above picture, it is“5”)		**S2.**	**[S2.+1, S2.]**
X coordinate	Point 1	**S2.+1**	**[S2.+3, S2.+2]**
	Point 2	**S2.+2**	**[S2.+5, S2.+4]**
	...	**...**	**...**
	Point n (last)	**S2.+n**	**[S2.+2n+1, S2.+2n]**
Y coordinate	Point 1	**S2.+n+1**	**[S2.+2n+3, S2.+2n+2]**
	Point 2	**S2.+n+2**	**[S2.+2n+5, S2.+2n+4]**
	...	**...**	**...**
	Point n (last)	**S2.+2n**	**[S2.+4n+1、S2.+4n]**

- An operation error is caused in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067.

  • When the Xn data is not set in the ascending order in the data table (error code: K6706)

  The data table is searched from the low-order side of the device numbers in the data table in the operation. Accordingly, even if only some Xn data is set in the ascending order in the data table, the instruction is executed without operation error up to the area of the data table in which the Xn data is set in the ascending order.

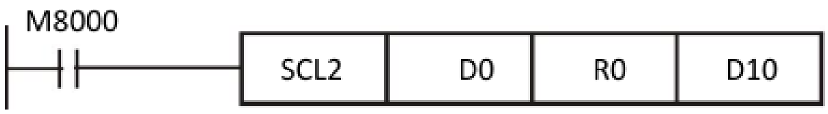
  • When **S1.** is outside the data table (error code: K6706)

  • When the value exceeds the 32-bit data range in the middle of operation (error code: K6706)

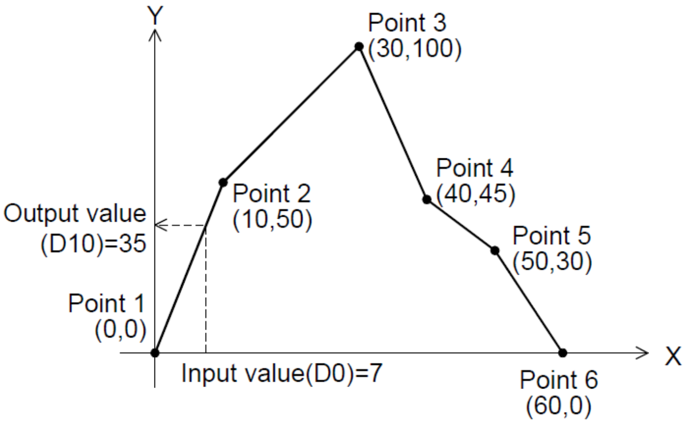
  In this case, check whether the distance between points is not “65535” or more.

  If the distance is “65535” or more, reduce the distance between points.

Program Example



◆ The value input to D0 is processed by scaling based on the conversion table for scaling set in R0 and later, and output to D10.



Set item		Device	Setting contents
Number of coordinate points		R0	K6
X coordinate	Point 1	R1	K0
	Point 2	R2	K10
	Point 3	R3	K30
	Point 4	R4	K40
	Point 5	R5	K50
	Point 6	R6	K60
Y coordinate	Point 1	R7	K0
	Point 2	R8	K50
	Point 3	R9	K100
	Point 4	R10	K45
	Point 5	R11	K30
	Point 6	R12	K0

27 High Speed Processing 2

FNC NO.	Instruction	Function	Supported PLC series		
			3G PLC	2N PLC	MX2N PLC
280	HSCT	High Speed Counter Compare With Data Table	★		
281	—				
282	—				
283	—				
284	—				
228	—				
286	—				
287	—				
288	—				
289	—				

27.1 HSCT/High Speed Counter Compare With Data Table

This instruction compares the current value of a high speed counter with a data table of comparison points,and then sets or resets up to 16 output devices.

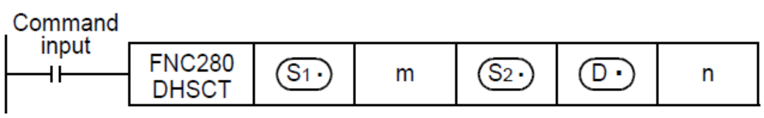
Instruction	Operand Type	Function						
FNC280 HSCT D	S1. m S2. D. n	16-bit Instruction	Mnemonic	Operation Condition		32-bit Instruction	Mnemonic	Operation Condition
		9 steps	HSCT	Continuous Operation		21 steps	DHSCT	Continuous Operation

Operand number	Operand	Description	Type
Operand number	S1.	Head device number storing the data table **Applicable devices:** D, R, Modify	BIN16 bit /32 bit
	m	Number of comparison points in data table [1≦m≦128] **Applicable devices:** K, H	BIN16 bit
	S2.	High speed counter number (C235 to C255) **Applicable devices:** C, Modify	BIN32 bit
	D.	Head device number to which the operation status is output **Applicable devices:** Y, M, S, Modify	bit
	n	Number of devices to which the operation status is output[1≦n≦16] **Applicable devices:** K, H	BIN16 bit

Instruction Explanation

32-bit operation(DHSCT)

The current value of a high speed counter specified in **S2.** is compared with the data table shown below which has (**3×m**) points stored in **S1.** and later, and the operation output set value (ON or OFF) specified in the data table is output to **D.** ~ **D.+n-1**.

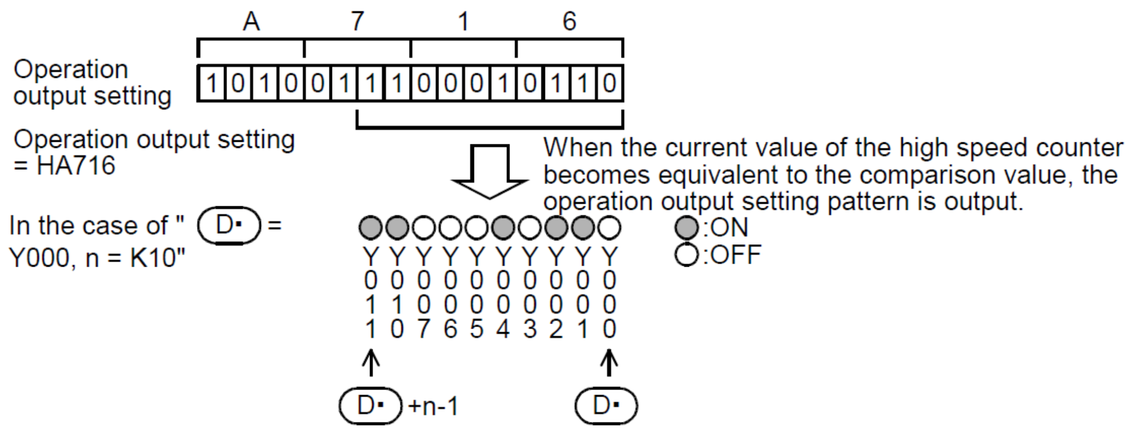


Data table used for comparison

Comparison point number	Comparison value	Operation output set value (SET [1] or RESET [0])	Operation output destination
0	**S1.+1, S1.**	**S1.+2**	**D.~D.+n-1**
1	**S1.+4, S1.+3**	**S1.+5**	
2	**S1.+7, S1.+6**	**S1.+8**	
...	**...**	**...**	
m-2	**S1.+3m-5, S1.+3m-6**	**S1.+3m-4**	

	m-1	**S1.+3m-2, S1.+3m-3**	**S1.+3m-1**	

Operation output setting (SET [1] or RESET [0]) [Up to 16 points]



1) When this instruction is executed, the data table is set as the comparison target.

2) When the current value of the high speed counter, specified in **S2.**, becomes equivalent to the comparison value in the data table, the corresponding operation output specified in the data table is output to **D.~D.+n-1**.

If an output (Y) is specified in **D.**, the output processing is executed immediately without waiting for the output refresh executed by the END instruction.

When specifying an output (Y), make sure that the least significant digit of the device number is “0”.

Examples: Y000, Y010 and Y020

3) Immediately after step 2), “1” is added to the current table counter value D8138.

4) The next comparison point is set as the comparison target data.

5) Steps 2) and 3) are repeated until the current value of the table counter D8138 becomes "m". When the current value becomes "m", the instruction execution complete flag M8138 turns ON, and the execution returns to step 1). At this time, the table counter D8138 is reset to "0".

6) When the command contact is set to OFF, execution of the instruction is stopped and the table counter D8138 is reset to “0”.

- **Related device**

Device	Name	Description
M8138	HSCT(FNC280) instruction execution complete flag	Turns ON when the operation for the final table No. "m-1" is completed
D8138	HSCT(FNC280) table counter	Stores the comparison point number handled as the comparison target.

- **Cautions**

• This instruction can be executed only once in a program.

If this instruction is programmed two or more times, an operation error is caused by the second instruction and later, and the instruction will not be executed. (error code: K6765)

• This instruction constructs the data table at the END instruction of the first execution of the instruction.

Accordingly, the operation output works after the second scan and later.

• DHSCT Instruction(FNC280), DHSCS Instruction(FNC53),DHSCR Instruction(FNC54) and DHSZ Instruction(FNC55) up to 32 instructions can be executed in one operation cycle. An operation error is caused by the 33rd instruction and later, and the instruction will not be executed. (error code: K6705)

• If an output (Y) is specified in **D.**, the output processing is executed immediately without waiting for the output refresh executed by END instruction.
When specifying an output (Y), make sure that the least significant digit of the device number is "0". Examples: Y000, Y010 and Y020

• When a high speed counter specified in **S2.** is indexed with index, all high speed counters are handled as software counters.

• For this instruction, only one comparison point (one line) is handled as the comparison target at one time. Processing will not move to the next comparison point until the current counter value becomes equivalent to the comparison point currently selected as the comparison target.
If the current value of a high speed counter executes up counting using the comparison data table shown in the operation example on the previous page, make sure to execute the instruction while the current value of the high speed counter is smaller than the comparison value in comparison point No. 1.

● An operation error occurs in the following cases; The error flag M8067 turns ON, and the error code is stored in D8067.

• When any devices other than high speed counters C235 to C255 are specified in **S2.** (error code: K6706)

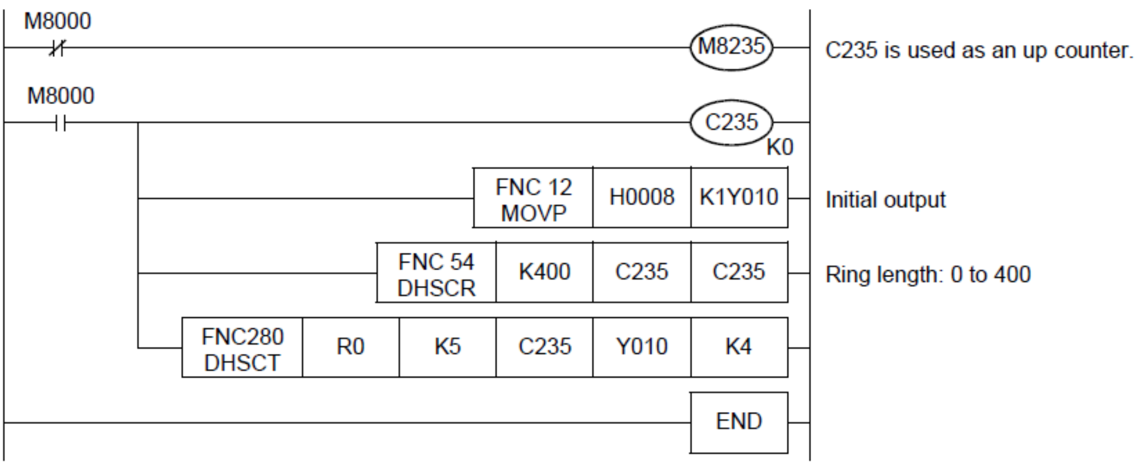
• When the "3m-1"th device from a device specified in **S1.** exceeds the last number of the device (error code: K6706)

• When the "n"th device from a device specified in **D.** exceeds the last number of the device (error code: K6706)

• When this instruction is used two or more times in a program (error code: K6765)

• DHSCT Instruction(FNC280), DHSCS Instruction(FNC53), DHSCR Instruction(FNC54) and DHSZ Instruction (FNC55) up to 32 instructions can be executed in one operation cycle. An operation error is caused by the 33rd instruction and later, and the instruction will not be executed. (error code: K6705)

Program Example



◆ The current value of C235 (counting X000) is compared with the comparison data table set in R0 and later, and a specified pattern is output to Y010 to Y013.

◆ **Operation**

Comparison point	Comparison data		SET/RESET pattern		Table counter D8138
	Device	Comparison value	Device	Operation output set value	
0	R1,R0	K100	R2	H0007	0↓
1	R4,R3	K150	R5	H0004	1↓
2	R7,R6	K200	R8	H0003	2↓
3	R10,R9	K250	R11	H0006	3↓
4	R13,R12	K300	R14	H0008	4↓ (Repeated from "0↓")

◆ **Current value of C235**