

SEL LANGUAGE

Programming Manual



Intelligent Actuator Inc.

This publication was written to assist you in better understanding this part of your IA system. If you require further assistance, please contact IA Technical Support. For Central and East Coast Time Zones, please call our Itasca, IL office at 1-800-944-0333 or FAX 630-467-9912. For Mountain and Pacific Time Zones, please call our Torrance, CA office at 1-800-736-1712 or FAX 310-891-0815; Monday thru Friday from 8:30AM to 5:00PM.



Intelligent Actuator, Inc.

U.S. Headquarters
2690 W. 237th Street
Torrance, CA 90505
310-891-6015 / 310-891-0815 FAX

Intelligent Actuator, Inc.

Midwest Regional Office
1261 Hamilton Parkway
Itasca, IL 60143
630-467-9900 / 630-467-9912 FAX

www.intelligentactuator.com

©April, 1998 Intelligent Actuator, Inc. All rights reserved.

No portion of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written permission of Intelligent Actuator, Inc.

Disclaimer

The information and technical data contained herein are subject to change without notice. Intelligent Actuator, Inc. assumes no responsibility for any errors or omissions regarding the accuracy of the information contained in this publication.

Foreword

SEL Language is the simplest type of the numerous robot languages being used today. The perplexing problem of using simple language to effect high level control has been beautifully resolved with SEL Language.

The language most generally used for robot control is based on BASIC language which uses sentences formats and interpreting these sentences requires considerable time. As the level of expression gets higher, the interpreter becomes less capable of handling real time control ^{*1}. At this point, an extra step, compling ^{*2}, needs to be added. Furthermore, the process becomes increasingly complex, as one encounters problems with false commands, required knowledge of MS-DOS, etc.

It goes without saying that being able to accomplish the same task using a simple method is much better. So, we welcome you to step into the world of high level control using SEL Language which is a simple system, that operates at high speeds even as it is interpreting.

*1 An interpreter is executing a command as it translates that command into computer language.

*2 Compling refers to translating the command into computer language before the comand is executed,

Table of Contents

1.	Numerals and Symbols in SEL Language	
1.1	List of numerals handled by SEL Language	4
1.2	Symbols used in SEL Language	5
2.	I/O Ports	
2.1	Input ports	6
2.2	Output ports	6
2.3	List of I/O ports for the Super SEL type	7
2.4	List of I/O ports for the DS type	8
3.	Flags	10
4.	Variables	
4.1	What are variables?	11
4.2	Types of variables	11
5.	Tags	14
6.	Subroutines	15
7.	Axis Designation	
7.1	Axis No. and Display	16
7.2	Axis pattern.	17
8.	Structure of SEL Language	
8.1	Position program.	18
8.2	Commands	19
8.2-1	Structure of SEL Language	19
8.2-2	Expansion Condition	20
9.	List of Parameters	
9.1	Common parameters for multiple axes.	22
9.2	Common parameters for a single axis.	23
9.3	Parameters by axis	24
10.	List of SEL Language Command Codes by Function	26
11.	List of SEL Language Command Codes in Alphabetical Order	30

Table of Contents

12.	SEL Language	
12.1	Numeric calculation commands	32
12.2	Arithmetic calculation commands	34
12.3	Functional calculation commands	37
12.4	Logic operation commands	40
12.5	Comparison operation commands	43
12.6	Timer commands	44
12.7	I/O•Flag operation commands	46
12.8	Program control commands	52
12.9	Task management commands	54
12.10	Resource management commands	58
12.11	Position operation commands	59
12.12	Actuator control declarations	64
12.13	Actuator control commands	72
12.14	Structured IF commands	80
12.15	Structured DO commands	83
12.16	Branching commands	85
12.17	External I/O commands	89
12.18	String processing commands	92
13.	Error Codes	
13.1	List of Error Codes	98
13.2	What to do when an Error Code occurs	99

1. Numerals and Symbols in SEL Language

1.1 List of numerals handled by SEL Language

The various types of functions required in a program are expressed as numerals.

Function	Super SEL Type		DS Type		Remarks
	Global	Local	Global area	Local	
Input Port	000 ~ 287 (288)		001 ~ 015 (15)		Varies according to controller type
Output Port	300 ~ 587 (288)		300 ~ 307 (8)		Varies according to controller type
Flag	600 ~ 887 (288)	900 ~ 999 (100)	600 ~ 887 (288)	900 ~ 999 (100)	
Variable (Integers)	200 ~ 299 (100)	1 ~ 99 (99)	200 ~ 299 (100)	1 ~ 99 (99)	Use INB, OUTB for 99
Variable (Real numbers)	300 ~ 399 (100)	100 ~ 199 (100)	300 ~ 399 (100)	100 ~ 199 (100)	Use PPUT, PGET for 199
Column	300 ~ 999 (700 characters)		300 ~ 999 (700 characters)		
Tag No.		1 ~ 64 (64)		1 ~ 64 (64)	
Subroutine No.		1 ~ 64 (64)		1 ~ 64 (64)	
Axis No.	1 ~ 8		1		Varies according to controller type
Axis Pattern	8 7 6 5 4 3 2 1				Use 1 for the designated axis
Position No.	1 ~ 2000		1 ~ 500		
Program No.	1 ~ 64		1 ~ 32		
Step No.	1 ~ 3000		1 ~ 1000		
Task Level	1 ~ 5		1 ~ 5		
Resource No.	1 ~ 9		1 ~ 9		
Channel No.	1 ~ 2				Varies according to controller type
	Visible to all programs	Visible only within a program. (Local range clears at program startup)	Visible to all programs	Visible only within a program. (Local range clears at program startup)	Variables 99 and 199 are special variables used in calculation. Avoid using these for general use.

Note: Variables 99 and 199 are special registers that the system uses for calculations.

● Battery Back-up Range

When the power is turned back ON, everything will be cleared except the area backed up by the battery.
(Same as an emergency stop)

Program..... Stop
 Output Port Clear
 Local Flag Clear
 Local Variable Clear
 Home Position Clear
 Global Flag Maintained
 Global Variable Maintained

1. Numerals and Symbols in SEL Language

- Range of numerical values in SEL

SEL uses two types of numbers, integers and real numbers but are subject to the following limitations.

1. Inside the controller

The range of whole numbers that can be accommodated is $\pm 2,147,483,648$ and for real numbers the theoretical range is $\pm 3.4 \times 10^{38}$ as a single precision floating point.

2. Limitations in actual use

The programming tool developed initially was an LCD teaching pendant which resulted in certain constraints with respect to input and output from the program. The numerical values that can be handled from the program are -9,999,999~99,999,999 for integers and -999,999~9999,999 or -.999999~.999999, in other words an eight digit value including the decimal point sign for real numbers. Also, when doing floating point calculations, the significant figure can only be guaranteed up to 7 digits and it will include errors that are particular to floating points.

3. Position data

Internally, position data is handled as whole number data but during the calculation process, these are incorporated into real numbers and treated as real numbers. There are no problems when dealing with numbers ± 9999.999 but when these are internally calculated as general data and not position data (repeated multiplying and dividing), a problem arises with the accuracy of the last digit.

When using the Super SEL, please pay close attention to these points. In particular, if you use the CPEQ command in a comparative calculation using real numbers, you will see almost no correlation. In this case, you will need to use the CPLE/CPGE command which can view the large and small relations in parallel.

1.2 Symbols used in SEL Language

SYMBOL	MEANING	EXAPLANATION
ZR	Zero	When calculation results are 0, post turns ON
EQ	Equal	When operand 1 = operand 2, post turns ON
NE	Not equal	When operand 1 \neq operand 2, post turns ON
GT	Greater than	When operand 1 > operand 2, post turns ON
GE	Greater or equal to	When operand 1 \geq operand 2, post turns ON
LT	Less than	When operand 1 < operand 2, post turns ON
LE	Less than or equal to	When operand 1 \leq operand 2, post turns ON
PE	Position end	When movement is complete, post turns ON (turns ON 2 points before the end of path, circular and arc moves when used in successive, consecutive lines of code)
CP	Complete	When the command is completed, post turns ON
TU	Time up	After the time has elapsed, post turns ON
XX	No position data	When there is no valid value in the position, post turns ON
ON	ON	On
OF	OFF	Off
NT	Not	Invert
FN	Forward ON	Moves forward while the designated I/O · flag is ON
FF	Forward OFF	Moves forward while the designated I/O · flag is OFF
BN	Backward ON	Moves backward while the designated I/O · flag is ON
BF	Backward OFF	Moves backward while the designated I/O · flag is OFF

2. I/O Ports

2.1 Input Ports

Input ports are used for limit switches, sensor switches, etc.

Input No. Assignment	Type E-G
001~023	Standard
024~047	Option
048~071	Option
072~095	Option

Input No. Assignment	DS Type
001~015	Standard

2.2 Output Ports

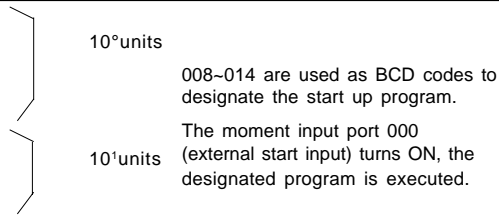
These are used as various output ports.

Output No. Assignment	Type E-G
300~323	Standard
324~347	Option
348~371	Option
372~395	Option

Output No. Assignment	DS Type
300~307	Standard

2. I/O Ports

2.3 List of ports for the Super SEL type

Port No.	Function	Explanation
000	External Start input	At the moment input turns ON, the designated program is executed.
001	User input	Users can use this port as they choose.
002	Emergency stop b-type contact input	When this input turns OFF, the controller goes into emergency stop.
003	System reserve	Cannot be used (this port will be used to add a new function in the future).
004	System reserve	Cannot be used (this port will be used to add a new function in the future).
005	User input	Users can use this port as they choose.
006	User input	Users can use this port as they choose.
007	User input	Users can use this port as they choose.
008	PRG No. 01 (user input)	 <p>10⁰units 008-014 are used as BCD codes to designate the start up program. The moment input port 000 (external start input) turns ON, the designated program is executed.</p>
009	PRG No. 02 (user input)	
010	PRG No. 04 (user input)	
011	PRG No. 08 (user input)	
012	PRG No. 10 (user input)	
013	PRG No. 20 (user input)	
014	PRG No. 40 (user input)	
015 ┆ 023	User input	Users can use this output as they choose.
024 ┆ 287	Expansion input	These are user input ports that can be used by adding on an expansion I/O card unit or high speed input port unit.
300	Emergency stop/Alarm output	This turns ON during an emergency stop or when an error occurs.
301	Ready output	This turns ON when the controller is ready.
302 ┆ 323	User output	Users can use this output as they choose.
324 ┆ 587	Expansion output	These are user output port that can be used by adding on an expansion I/O card unit.

2. I/O Ports

2.4 List of I/O ports for the DS type

Program Mode

Port No.	Function	Explanation
	PRG No. 01 (user input)	10°units Used as BCD codes to designate the start up program. The moment input port 000 (external start input) turns ON, the designated program is executed.
	PRG No. 02 (user input)	
	PRG No. 04 (user input)	
	PRG No. 08 (user input)	
	PRG No. 10 (user input)	
	PRG No. 20 (user input)	
	Reserve	
	CPU reset input	Restarts the controller.
000	External start input	At the moment input turns ON, the designated program is executed.
001 } 015	User input	Users can use this port as they choose.
300	Emergency stop/Alarm output	This turns ON during an emergency stop or when an error occurs.
301	Ready output	This turns ON when the controller is ready.
302 } 307	User output	Users can use this port as they choose.

2. I/O Ports

Positioning Mode

Port No.	Function	Explanation
	PRG No. 01 (user input)	When using the positioning mode, turn the Program No. input to [0] (OFF) status.
	PRG No. 02 (user input)	
	PRG No. 04 (user input)	
	PRG No. 08 (user input)	
	PRG No. 10 (user input)	
	PRG No. 20 (user input)	
	Reserve	
	CPU reset input	Restarts the controller.
000	External start input	At the moment input turns ON, the designated program is executed.
001	Hold input	When this turns ON, the actuator moves to the desinated position.
002 ┆ 003	NC	002~003 changes to NC during positioning mode.
004	Position No.1 input	<div style="display: flex; align-items: center;"> <div style="margin-right: 10px;"> 10^0units 10^1units 10^2units </div> <div> <p>004~014 are used as BCD codes to designate the Position No.</p> <p>The moment input port 000 (external start input) turns ON, the actuator moves to the designated position.</p> <p>Homing is performed when there is no Position No. designation.</p> </div> </div>
005	Position No. 2 input	
006	Position No. 4 input	
007	Position No. 8 input	
008	Position No. 10 input	
009	Position No. 20 input	
010	Position No. 40 input	
011	Position No. 80 input	
012	Position No. 100 input	
013	Positon No. 200 input	
014	Position No. 400 input	
015	NC	015 changes to NC during positioning mode.
300	Emergency stop/Alarm output	This turns ON during an emergency stop or when an error occurs.
301	Ready output	This turns ON when the controller is ready.
302	Positioning complete output	This turns ON when position move is complete.
303 ┆ 307	NC	303~307 changes to NC during mode.

3. Flags

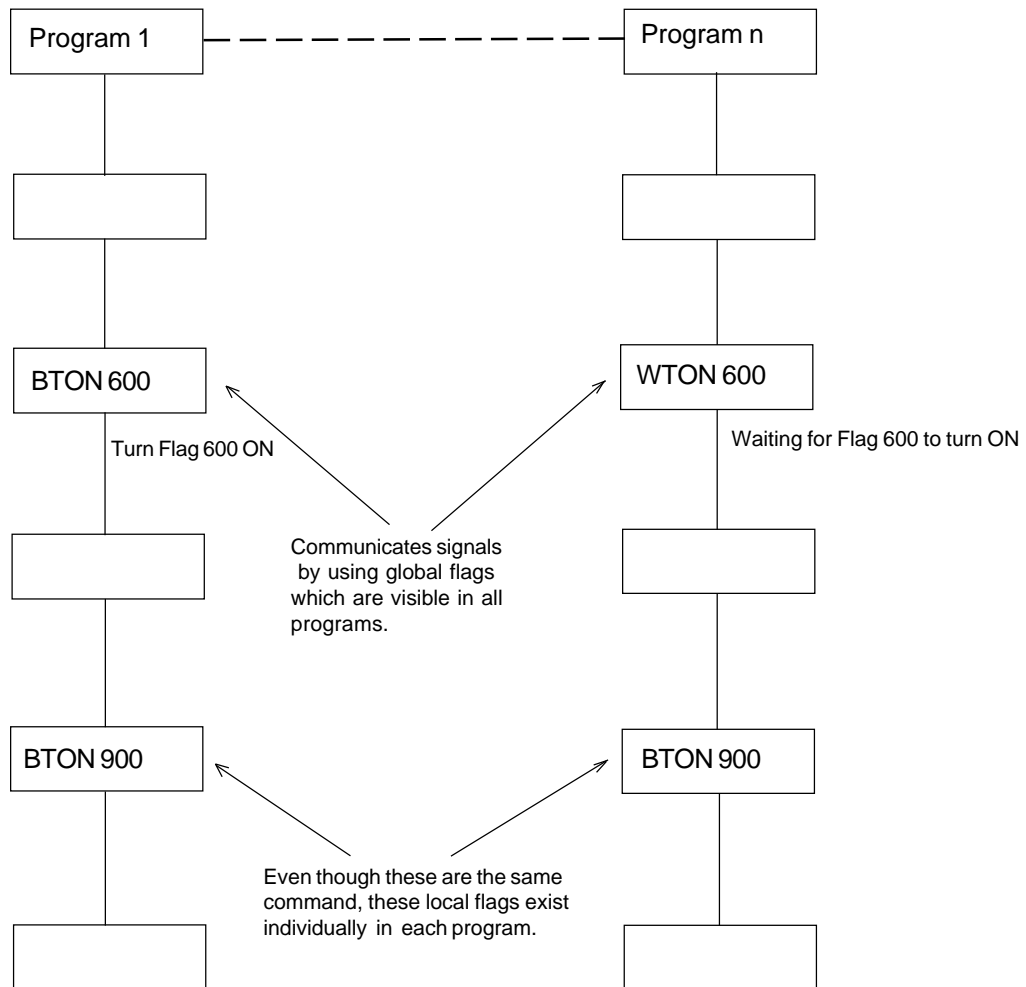
3. Flags

The function of flags is to set and reset data within "Memory." This is analogous to "internal relays" or "coils" in a PLC.

In general, there are two (2) types of flags: Global flags 600 ~ 887 which can be used in all programs and local flags 900 ~ 999 which can be used *only* in individual programs.

Global flags are saved when the power is turned OFF (battery backup). Local flags are erased when the power is turned OFF.

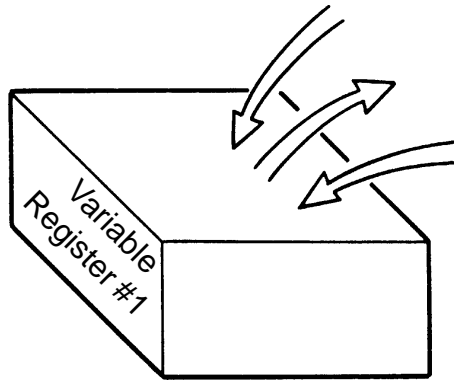
Flag Number	600~887	Global flag: Can be used in all programs
Flag Number	900~999	Local flag: Can be used only within an individual program



4. Variables

4.1 What are variables?

The term "variable register" is a software term. Imagine a box that holds numbers. Numbers can be put in and taken out, added, subtracted, and so on.



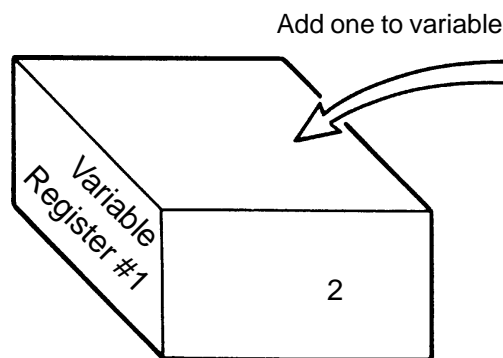
Put 1234 into variable register #1

Take 456 out of variable register #1

Add 1 to variable register #1

Command	Operand 1	Operand 2
Add	1	1

This command adds 1 to variable register #1. If the register contains 2, then the variable becomes 3.



Add one to variable register #1

(Already contains "2")

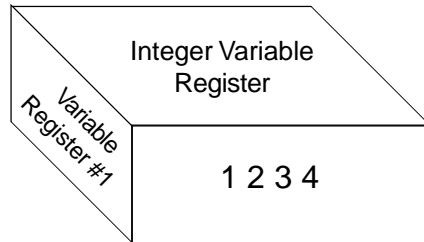
4. Variables

4.2 Types of variables

There are two types of variables.

① Integer variable

These are whole number variables which cannot take decimal points. For example: [-2, -1, 0, 1, 2, 3]

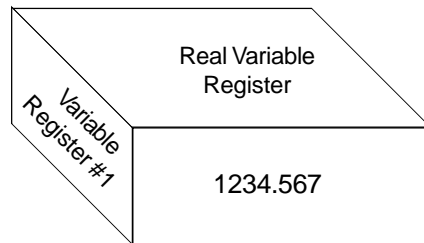


Integer Variable No.	200~299	Global integer variable: Can be used in all programs
Integer Variable No.	1~99	Local integer variable: Can be used only within an individual program

Note: The variable 99 is a special register for whole integer calculation.
The numbers that can be input in the program are -9,999,999 to 99,999,999.

② Real variable

These are variables that can accommodate the actual value exactly as it appears, including digits following the decimal point. [Example: 1234.567]



Integer Variable No.	300~399	Global integer variable: Can be used in all programs
Integer Variable No.	100~199	Local integer variable: Can be used only within an individual program

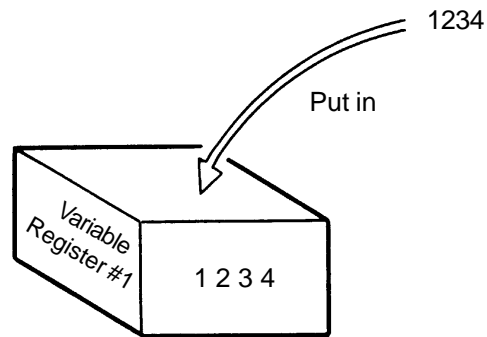
Note: The variable 199 is a special register for real number calculation.
The numbers that can be input in the program are -99,999.9 to 999,999.99 (8 digits which includes the decimal point sign).

4. Variables

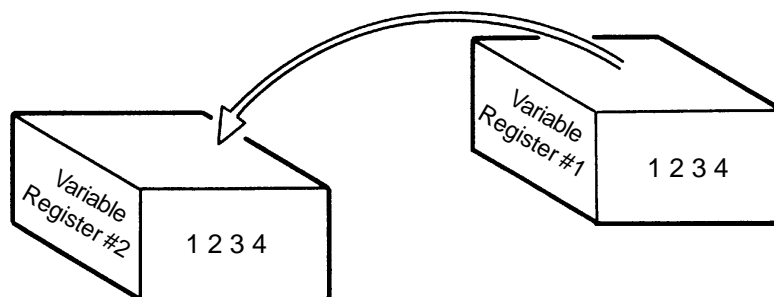
③ Variables with an asterisk (*)

The asterisk symbol (*) is used to designate contents of the variable register. In the example given below, the contents in variable register 1 are placed in variable register 2. If "1234" is in variable register 1, then "1234" is what goes in variable register 2.

Command	Operand 1	Operand 2
LET	1	1234

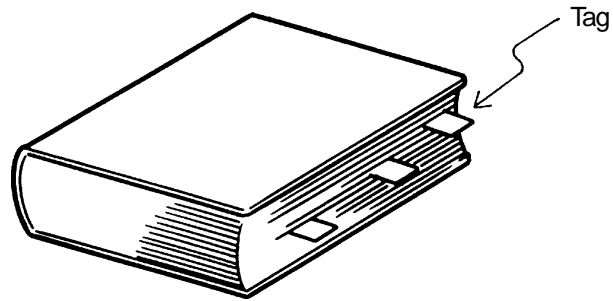


Command	Operand 1	Operand 2
LET	2	*1



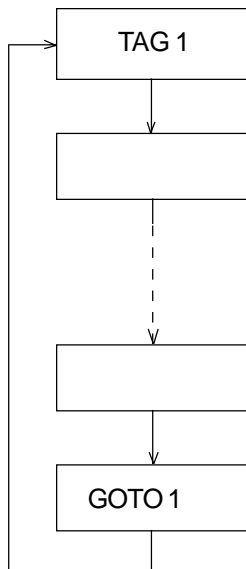
5. Tags

"Tag" means heading. A TAG can be thought of as the same as placing labels on important pages. The TAG as it is used in the SEL programming language is the "return to" area and is used in conjunction with the GOTO command to provide programming loops.



Command	Operand 1
TAG	Tag No. (Integers 1~64)

Can be used individually in each program.



6. Subroutines

Frequently repeated steps in a program can be expressed as subroutines in order to simplify the entire application program. These subroutines are individually usable in each program. (Up to a maximum of 15 subroutines can be nested)

Command	Operand 1
EXSR	Subroutine No. (1 ~ 64 Integers, or Variables)

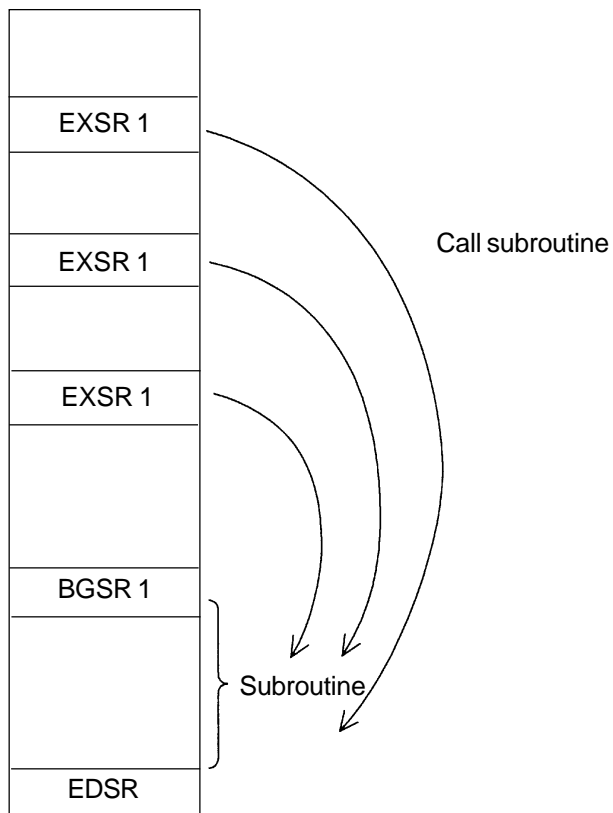
Execute subroutine command

Command	Operand 1
BGSR	Subroutine No. (1 ~ 64 Integers)

Begin subroutine command

Command	Operand 1
EDSR	-----

End subroutine command



7. Axis Designation

There are two ways to designate the axes to be used: axis number and axis pattern.

7.1 Axis number and notation

With the Super SEL controller, multiple axes are indicated as shown in the table, but it is possible to change the figures using the parameters.

Axis No.	Default Notation
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8

The axis no. is used when designating one axis out of many axes.

Commands to designate Axis No. are:
BASE, PPUT, PGET

Note: The DS type displays only one axis.

7. Axis Designation

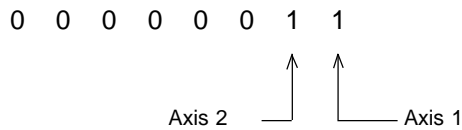
7.2 Axis Pattern

Selection of an axis is specified by either "1" or "0"

	(Upper)							(Lower)
Axis No.	8	7	6	5	4	3	2	1
Used	1	1	1	1	1	1	1	1
Not Used	0	0	0	0	0	0	0	0

Example

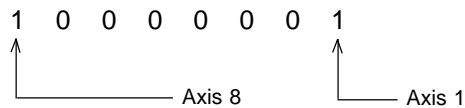
If Axis 1 and Axis 2 are in use, then this is signified by...



The zeroes before the 1 are unnecessary. The simplified form is 11, without leading zeroes.

Example

If Axis 1 and Axis 8 are in use, then this is signified by...



In this example, the zeroes *are* necessary in order to indicate the position of Axis 8.

Axis pattern is used when designating more than one axis at the same time.

Axis pattern designation command

OFST, GRP, SVON, SVOF, HOME, JFWN, JFWF, JBWN, JBWF, STOP, PTST, PRED

Note: The axis pattern for the DS type is preset since there is only one axis.

8. Structure of SEL Language

The SEL programming consists of a position and application program (command) section.

8.1 Position Program

In the position section, we have coordinates, velocity, acceleration, and variables.

Position No.	Velocity	Acceleration	Axis 1	Axis 2	Axis 3	Axis 4	Axis 5	Axis 6	Axis 7	Axis 8
1										
2										
3										
4										
...										
1997										
1998										
1999										
2000										

1-1500 mm/sec
*1,2

Standard
0.3G
*2

±9999.999mm

- *1 Varies according to the actuator model.
- *2 When velocity and acceleration are set in the position data, this has priority over the data set in the application program. To validate the application data, set x.xxx or 0 in the position data.

Note: The DS type is a single axis only. Also position numbers go up to 500.

8. Structure of SEL Language

The outstanding feature of Super SEL Language is the simplicity of its command structure which eliminates the need for a compiler and allows high speed operation with just an interpreter.

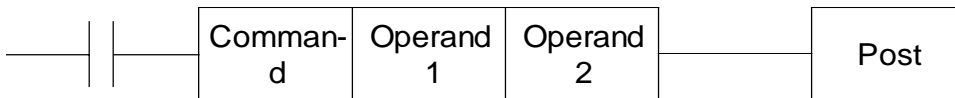
8.2 Commands

8. 2-1 Structure of Super SEL Language

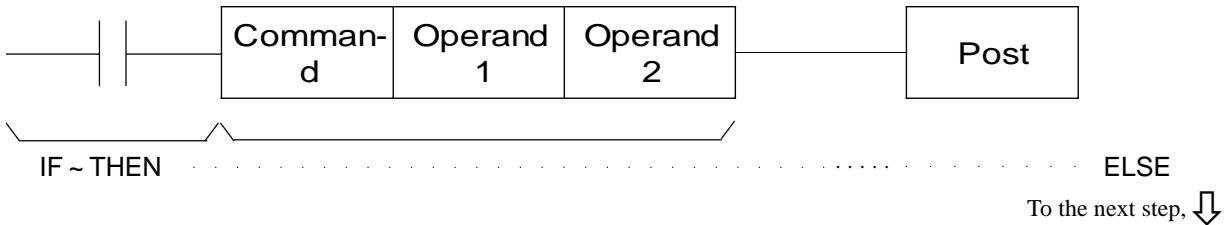
One step of the command has the following structure.

Expansion (AND · OR)	Input Condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	

Putting this in a ladder diagram,

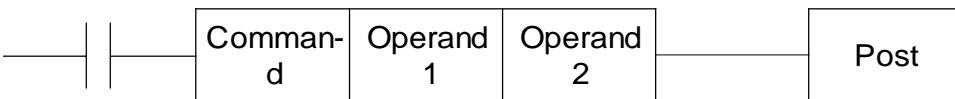


(1) The conditions before the commands are equivalent to "if ~ then" statements in BASIC language.



- ① Carry out a command when an input condition is established, and turn the post ON, if post is designated. When not established, go on to the next step regardless of the next command (ex. WTON, WTOF). The designated post remains the same, however it needs to be monitored carefully.
- ② If there is no conditioning set up, carry out command unconditionally.
- ③ If condition is used as "negative condition", then place an "N" (NOT).
- ④ Input/output port & flag can be used for condition.

(2) Post is set based on the result of the command execution.



- ① Actuator motion control commands: becomes OFF immediately after the command starts to be executed, and becomes ON when the command is completed. Computation commands: when the result becomes a certain value, it turns ON, and it stays OFF otherwise.
- ② Output ports and flags can be used for the post section.

8. Structure of SEL Language

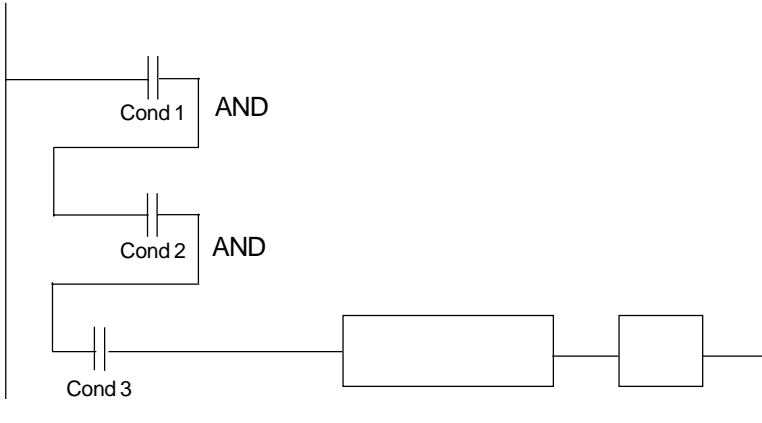
8. 2-2 Expansion Condition

It is possible to combine conditions to make more complicated conditions as follows:

AND Expansion

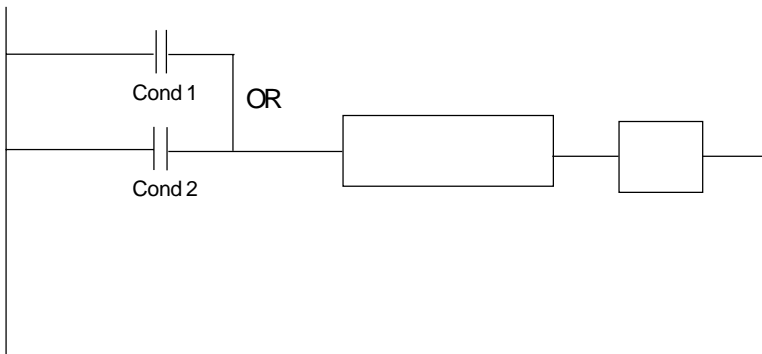
(Ladder Diagram display)

(Super SEL Language)



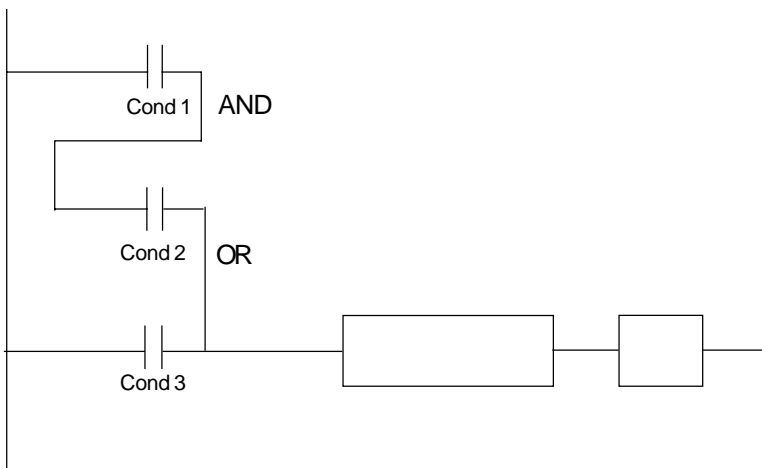
Expansion	Input	Commands			Output
		Command	Operand1	Operand2	
	Condition1				
AND	Condition 2				
AND	Condition 3	Command	Operand1	Operand2	

OR Expansion



Expansion	Input	Commands			Output
		Command	Operand1	Operand2	
	Condition 1				
OR	Condition 2	Command	Operand 1	Operand 2	

AND/OR Expansion



Expansion	Input	Commands			Output
		Command	Operand1	Operand2	
	Condition1				
AND	Condition 2				
OR	Condition 3	Command	Operand1	Operand2	

Note: By convention, all "AND" operations are performed before the "OR" operations when they are used in conjunction.

9. List of Parameters

Reference values are indicated below. Settings at the time of shipment vary according to the actuator model.

9.1 Common parameters for multiple axes

(1) Servo parameters

No.	Parameter Name	Default	Content
1	Axis Size	8	Number of axes
2	Numerator	1	Numerator
3	Denominator	1	Denominator
4	Override (%)	100	Override
5	Acceler (G)	0.30	Acceleration factor
6	Acceler Max (G)	1.00	Maximum acceleration factor (G)
7	Drive Vel (mm/s)	100	Drive velocity (mm/sec)
8	Drive Vel Max (mm/s)	1000	Maximum drive velocity (mm/sec)

(2) Program parameters

No.	Parameter Name	Default	Content
1	Auto Start Program	0	Auto start program number No.
2	Emergency Program	0	Emergency stop program number No.
3	Program Size	64	Number of programs
4	Task Size	16	Number of tasks
5	Step Size	3000	Number of program steps
6	Time Slice	0.01	Time slice check value

(3) Point parameters

No.	Parameter Name	Default	Content
1	Point Size	2000	Point data quantity

(4) Arc parameters

No.	Parameter Name	Default	Content
1	Circle Angle (°)	15.0	Slice angle (°)
2	Circle Delt (mm/s)	0	Velocity increment (mm/s)

9. List of Parameters

(5) Serial I/O parameters

No.	Parameter Name	Default	Content
1	Terminal ID	99	Multi-drop address code
2	Time Out (S)	0	Time out (S)
3	Baud Rate (bit/s)	3	Baud rate (bit/s)
4	Clear Length	0	Character length
5	Parity	0	Parity
6	Stop Bit	0	Stop bit

(6) Parallel input port parameters

No.	Parameter Name	Default	Content
1	Device No.	1~3	Device No.
2	Unit ID	1	Unit No.
3	Scan Time (ms)	1	Scan time (ms)
4	Unit ID	1	Unit No.
5	Scan Time (ms)	1	Scan time (ms)
6	Unit ID	1	Unit No.
7	Scan Time (ms)	1	Scan time (ms)
8	Unit ID	1	Unit No.
9	Scan Time (ms)	1	Scan time (ms)

9. List of Parameters

9.2 Common parameters for a single axis

(1) Servo parameters

No.	Parameter Name	Default	Content
1	Axis Size	1	Number of axes
2	Numerator	1	Numerator
3	Denominator	1	Denominator
4	Over Ride (%)	100	Over ride
5	Acceler (G)	0.30	Acceleration factor (G)
6	Acceler Max (G)	1.00	Maximum acceleration factor (G)
7	Drive Vel (mm/s)	100	Drive velocity (mm/sec)
8	Drive Vel Max (mm/s)	1000	Maximum drive velocity (mm/sec)

(2) Program parameters

No.	Parameter Name	Default	Content
1	Auto Start Program	0	Auto start program number No.
2	Emergency Program	0	Emergency stop program number No.
3	Program Size	32	Number of programs
4	Task Size	8	Number of tasks
5	Step Size	1000	Number of program steps
6	Time Slice	0.01	Time slice check value

(3) Point parameters

No.	Parameter Name	Default	Content
1	Point Size	500	Point data quantity

(4) Serial I/O parameters

No.	Parameter Name	Default	Content
1	Terminal ID	99	Multi-drop address code
2	Time Out (S)	0	Time out (S)
3	Baud Rate (bit/s)	3	Baud rate (bit/s)
4	Char Length	0	Character length
5	Parity	0	Parity
6	Stop Bit	0	Stop bit

9. List of Parameters

9.3 Parameters by axis

(1) Servo parameters by axis

No.	Parameter Name	Default	Content
1	Axis Name	1~8	Axis name
2	Servo Service	400	No. of times of servo service (times/s)
3	Numerator	1	Numerator
4	Denominator	1	Denominator
5	Over Ride (%)	100	Over ride (%)
6	Acceler	0.30	Acceleration (G)
7	Jog Vel (mm/s)	30	Jog velocity (mm/s)
8	Pend Band	10	Position end band (pulse)
9	Soft Limit Offset	2.0	Software limit offset
10	Soft Limit (+)	9999	Soft Limit (+)
11	Soft Limit (-)	0	Soft Limit (-)

(2) Homing parameters by axis

No.	Parameter Name	Default	Content
1	Home Dir	0	Home direction
2	Home Type	0	Homing method
3	Home Sequence	1	Sequence
4	Home Sw Pol	1	Limit input polarity
5	Home Z Edge	1	Z-phase detect edge
6	Home Creep Vel	100	Creep velocity
7	Home Back Vel	10	Run-in velocity
8	Home Z Vel	5	Z-phase search velocity
9	Home Offset	0	Offset move amount (length)
10	Home Deviation	667	Hard stop deviation (pulse)
11	Home Current	60	Current Limit

9. List of Parameters

(3) Motor parameters by axis

No.	Parameter Name	Default	Content
1	Motor RPM Max	4000	Motor RPM maximum
2	Encoder Pulse	400	Encoder pulse per revolution
3	Screw Lead	8	Screw lead (mm)
4	Multiple	4	Encoder pulse multiplier
5	Brake Time	0.1	Brake time
6	Position Gain	60	Position gain
7	Speed Gain	80	Speed gain
8	F/F Gain	0	Feed forward gain
9	Integral Gain	30	Integral gain
10	Total Gain	150	Total gain
11	Int. Volt. Lmt	60	Integral volatege limit
12	Over Speed	410	Over speed constant
13	Error Range	2666	Cumulative error
14	Motor Max Cur	90	Motor maximum current
15	Motor Over Load	16300	Motor overload lower limit

10. List of SEL Language Command Codes

Category	Condition	Command	Operand 1	Operand 2	Output	Function	Page	Multiple Axis	Single Axis
Numeric Calculations	Optional	LET	Assign variable	Assign Number	ZR	Assign	32	O	O
	Optional	TRAN	Variable copy destination	Variable to copy	ZR	Copy	32	O	O
	Optional	CLR	Begin clearing variable	End clear variable	ZR	Clear variables	33	O	O
Arithmetic Calculations	Optional	ADD	Add to variable	Number added	ZR	Add	34	O	O
	Optional	SUB	Subtract from variable	Number subtracted	ZR	Subtract	34	O	O
	Optional	MULT	Multiply variable	Multiplier Number	ZR	Multiply	35	O	O
	Optional	DIV	Divide variable	Dividing number	ZR	Divide	35	O	O
	Optional	MOD	Assign remainder variable	Calculation number (radian)	ZR	Figure remainder	36	O	O
Functional Calculations	Optional	SIN	Assign sin variable (radian)	Calculation number (radian)	ZR	Sine	37	O	O
	Optional	COS	Assign cosine variable	Calculation Number (radian)	ZR	Cosine	38	O	O
	Optional	TAN	Assign tangent variable	Calculation Number (radian)	ZR	Tangent	38	O	O
	Optional	ATN	Assign arctangent variable	Calculation Number	ZR	Arctangent	38	O	O
	Optional	SQR	Assign square root variable	Calculation Number	ZR	Square root	39	O	O
Logic Operations	Optional	AND	Variable to apply logic and to	Calculation Number	ZR	Logic and	40	O	O
	Optional	OR	Variable to apply logic or to	Calculation Number	ZR	Logic or	41	O	O
	Optional	EOR	Variable to apply exclusive or logic to	Calculation Number	ZR	Exclusive logic	42	O	O
Compare	Optional	CPXX	Comparison variable	Comparison Number	EQ NE GT GE LT LE	Comparison	43	O	O
Timer	Optional	TIMW	Wait time (sec)		TU	Time wait	44	O	O
	Optional	TIMC	Program No.			Time wait cancel	44	O	O
	Optional	GTTM	Time assign variable			Acquire time	45	O	O
IO • Flag Operations	Optional	BTXX	Start output • flag	End output • flag		Output • Flag [ON OF NT]	46	O	O
	Optional	WTXX	Wait time (sec)		TU	Input • Flag [ON OF NT] Wait	47	O	O
	Optional	IN	First IO • flag	End input • flag		Binary input (Max 31 bit)	48	O	O
	Optional	INB	First IO • flag	No. of digits to convert to		BCD input (Max 8 digits)	49	O	O
	Optional	OUT	First IO • flag	End output • flag		Binary input (Max 31 bit)	50	O	O
	Optional	OUTB	First IO • flag			BCD input (Max 8 digits)	51	O	O
Program Control	Optional	GOTO	Tag No. for jump			Jump	52	O	O
		TAG	Stated tag No.			Declare jump target	52	O	O
	Optional	EXSR	Execute subroutine No.			Execute subroutine	53	O	O
		BGSR	Stated subroutine			Start subroutine	53	O	O
		EDSR				End subroutine	53	O	O
Task Management	Optional	EXIT				Terminate program	54	O	O
	Optional	EXPG	Execute program No.		CP	Start program	54	O	O
	Optional	ABPG	Stop program No.		CP	Stop other programs	55	O	O
	Optional	*SLPG				Task pause	55	O	O
	Optional	*WUPG	Startup program No.		CP	Other task status	55	O	O
	Optional	*GTPG	Variable to store task status	Acquire program No.		Acquire task level	56	O	O
	Optional	*GTPR	Variable to store task level			Acquire task level	56	O	O
	Optional	*STPR	Task level			Change task level	57	O	O
	Optional	*SLIC	Task level	Check value (sec)		Time slice setting	57	O	O

* Commands not yet publicly available cannot be used.

Note: The circle (O) in the multiple axes, single axis columns indicates that the commands can be used for multiple axes or a single axis.

10. List of SEL Language Command Codes

Category	Condition	Command	Operand 1	Operand 2	Output	Function	Page	Multiple Axis	Single Axis
Resource	Optional	*GTRS	Obtain resource No.			Acquire resource	58	O	O
	Optional	*RLRS	Return resource No.			Return resource	58	O	O
Position Operation	Optional	PGET	Axis No.	Position No.		Assign position to variable 199	59	O	O
	Optional	PPUT	Axis No.	Position No.		Assign value of variable 199	59	O	O
	Optional	PCLR	Starting position No.	End position No.		Clear point data	60	O	O
	Optional	PCPY	Copy target position No.	Copy source position No.		Copy point data	60	O	O
	Optional	PRED	Axis pattern to read	Store target position No.		Read current position of axis	61	O	O
	Optional	PTST	Confirmed axis pattern	Confirmed position No.	XX	Confirm position data	61	O	O
	Optional	PVEL	Velocity (mm/sec)	Assign target position No.		Assign position velocity	62	O	O
	Optional	PACC	Acceleration (G)	Assign target position No.		Assign position acceleration	62	O	O
	Optional	PAXS	Axis pattern variable No.	Position No.		Read axis pattern	63	O	O
	Optional	PSIZ	Size variable No.			Check position size	63	O	O
Actuator Control Declarations	Optional	VEL	Velocity (mm/sec)			Set velocity	64	O	O
	Optional	OVRD	Velocity ratio (%)			Set velocity factor	64	O	O
	Optional	ACC	Acceleration (G)			Set acceleration	65	O	O
	Optional	SCRV	Ratio (%)			Set S-motion ratio	66	O	O
	Optional	OFST	Default axis pattern	Offset value (mm)		Set offset	67	O	O
	Optional	ATRG	Position ratio (%)			Set arch trigger	67	O	
	Optional	HOLD	Pause input port			Declare pause port	68	O	O
	Optional	BASE	Standard axis No.			Set standard axis	69	O	
	Optional	CANC	Stop complete input port			Declare stop complete port	69	O	O
	Optional	DEG	Angle of division			Set angle of division	70	O	
	Optional	GRP	Valid axis pattern			Set group axes	70	O	
Optional	AXST	Variable to store status			Acquire axis status	71	O	O	
Actuator Control Commands	Optional	SVXX	Operating axis pattern	Acquired axis No.		Servo [ON OF]	72	O	O
	Optional	HOME	Homing axis pattern		PE	Homing	72	O	O
	Optional	MOVD	Move position		PE	Directly designate move	73		DS
	Optional	MOVP	Move target position No.		PE	Move to designated position	73	O	O
	Optional	MVDI	Move amount		PE	Incremental movement	74		DS
	Optional	MOVL	Move target position No.		PE	Interpolated move to designated position	74	O	
	Optional	MVPI	Move target position No.		PE	Move to designated position	75	O	O
	Optional	MVLI	Move target position No.		PE	Interpolated move to designated position	75	O	
	Optional	ARCH	Start position No.	End position No.	PE	Arch movement	76	O	
	Optional	PATH	Start position No.	End position No.	PE	Path movement	76	O	O
	Optional	CIR	Passing position No.	Passing position No.	PE	Arc movement	77	O	
	Optional	ARC	Passing position No.	End position No.	PE	Circular movement	78	O	
	Optional	JXWX	Moving axis pattern	Start I/O • flag	PE	JOG [FN FF BN BF]	79	O	O
	Optional	STOP	Stopped axis pattern		PE	Axis slows to a halt	79	O	O

* Commands not yet publicly available cannot be used.

Note: The circle (O) in the multiple axes, single axis columns indicates that the commands can be used for used for multiple axes or a single axis. However, MOVD, MVDI are command languages specific to the DS type.

10. List of SEL Language Command Codes

Category	Condition	Command	Operand 1	Operand 2	Output	Function	Page	Multiple Axis	Single Axis
Structured IF	Optional	IFXX	Compare variable	Comparison variable		Compare [EQ NE GT GE LT LE]	80	○	○
	Optional	ISXX	Column or literal	Column or literal		Character string comparison [EQ NE]	81	○	○
		ELSE				Declare execution target when IF command condition is not met.	82	○	○
		EDIF				Declare IF complete	82	○	○
Structured DO	Optional	DWXX	Comparison variable	Comparison variable		Loop [EQ NE GT GE LT LE]	83	○	○
		EDDO				Declare DO complete	83	○	○
	Optional	LEAV				Exit from DO	84	○	○
	Optional	ITER				Repeat DO	84	○	○
Branching	Optional	SLCT				Declare start of branching	85	○	○
		WHXX	Compare variable	Comparison variable		Value branching [EQ NE GT GE LT LE]	86	○	○
		WSXX	Compare column	Column or literal		Character string branching [EQ NE]	87	○	○
		OTHE				Declare branching target when conditions are not met	88	○	○
		EDSL				Declare SLCT complete	88	○	○
Serial	Optional	OPEN	Channel No.			SIO Open	89	○	○
	Optional	CLOS	Channel No.			SIO Close	89	○	○
	Optional	READ	Read channel	Read Column No.		SIO input	90	○	○
	Optional	WRIT	Write channel	Write Column No.		SIO output	91	○	○
	Optional	SCHA	Character code			Set ending character	91	○	○
String Processing	Optional	SCPY	Copy target column	Column or literal		Copy character string	92	○	○
	Optional	SCMP	Compare column	Column or literal		Compare character string	92	○	○
	Optional	SGET	Send target variable	Send target column		Acquire character	93	○	○
	Optional	SPUT	Send target column	Write data		Arrange characters	93	○	○
	Optional	STR	Conversion target column	Conversion source variable		Convert to decimal character string	94	○	○
	Optional	STRH	Conversion target column	Conversion source variable		Convert to hexadecimal character string	95	○	○
	Optional	VAL	Conversion target variable	Conversion source column		Convert to decimal value	96	○	○
	Optional	VALH	Conversion target variable	Conversion source column		Convert to hexadecimal value	97	○	○
	Optional	SLEN	No. of characters to be operated on			Set no. of characters for operation	97	○	○

Note: The circle (○) in the multiple axes, single axis columns indicates that the command can be used for multiple axes or a single axis. However, the commands in the serial I/O category and string processing cannot be used with the DS type.

[Command languages specific to the DS type]

Category	Condition	Command	Operand 1	Operand 2	Output	Function	Page	Multiple Axis	Single Axis
Actuator Control Commands	Optional	MOVD	Move position		PE	Directly designate move	73		○
	Optional	MVDI	Move position		PE	Incremental movement	74		○

11. Alphabetical List of SEL Language Command Codes

Condition	Command	Operand 1	Operand 2	Output	Function	Page	Multiple Axis	Single Axis
Optional	ABPG	Stop program No.		CP	Stop other programs	55	O	O
Optional	ACC	Acceleration (G)			Set acceleration	65	O	O
Optional	ADD	Add to variable	Number added	ZR	Add	34	O	O
Optional	AND	Variable apply logic and to	Calculation number	ZR	Logic and	40	O	O
Optional	ARC	Passing position No.	End position No.	PE	Circular movement	78	O	
Optional	ARCH	Start position No.	End position No.	PE	Arch movement	76	O	
Optional	ATN	Assign arctangent variable	Calculation number	ZR	Arctangent	38	O	O
Optional	ATRG	Position ratio (%)			Set arch trigger	67	O	
Optional	AXST	Variable to store status	Acquired axis No.		Acquire axis status	71	O	O
Optional	BASE	Standard axis No.			Set standard axis	69	O	
	BGSR	Stated subroutine No.			Start subroutine	53	O	O
Optional	BTXX	Start output • flag	End output • flag		Output • flag [ON OF NT]	46	O	O
Optional	CANC	Stop complete input port			Declare stop complete port	69	O	O
Optional	CIR	Passing position No.	Passing position No.	PE	Arc movement	77	O	
Optional	CLOS	Channel No.			SIO close	89	O	O
Optional	CLR	Begin clearing variable	End clear variable	ZR	Clear variables	33	O	O
Optional	COS	Assign cosine variable	Calculation number (radian)	ZR	Cosine	37	O	O
Optional	CPXX	Comparison variable	Comparison number	EQ NE GT GE LT LE	Comparison	43	O	O
Optional	DEG	Angle of division (°)			Set angle of division	70	O	
Optional	DIV	Divide variable	Dividing number	ZR	Divide	35	O	O
Optional	DWX X	Compare variable	Comparison variable		Compare [EQ NE GT GE LT IE]	83	O	O
	EDDO				Declare DO complete	83	O	O
	EDIF				Declare IF complete	81	O	O
	EDSL				Declare SLCT complete	88	O	O
	EDSR				End subroutine	53	O	O
	ELSE				Declare execution target when IF command condition is not met	81	O	O
Optional	EOR	Variable to apply exclusive logic to	Calculation number	ZR	Exclusive or logic	42	O	O
Optional	EXIT				Terminate program	54	O	O
Optional	EXPG	Execute program No.		CP	Start program	54	O	O
Optional	EXSR	Execute subroutine number			Execute subroutine	53	O	O
Optional	GOTO	Tag No. Jump			Jump	52	O	O
Optional	GRP	Valid axis pattern			Set group axes	70	O	
Optional	*GTPG	Variable to store task status	Acquire program No.		Acquire task status	56	O	O
Optional	*GTPR	Variable to store task level			Acquire task level	56	O	O
Optional	*GTRS	Obtain resource No.			Acquire resource	58	O	O
Optional	GTTM	Time assign variable			Acquire time	45	O	O

* Commands not yet publicly available cannot be used.

Note: The circle (O) in the multiple axes, single axis columns indicates that the command can be used for multiple axes or a single axis.

11. Alphabetical List of SEL Language Command Codes

Condition	Command	Operand 1	Operand 2	Output	Function	Page	Multiple Axis	Single Axis
Optional	HOLD	Pause input port			Declare pause port	68	O	O
Optional	HOME	Homing axis pattern		PE	Homing	72	O	O
Optional	IFXX	Compare variable	Comparison variable		Compare [EQ NE GT GE LT LE]	80	O	O
Optional	IN	First I/O • flag	End input • flag		Binary input (Max 31 bit)	48	O	O
Optional	INB	First I/O • flag	No. of digits to convert to		BCD input (Max 8 digits)	49	O	O
Optional	ISXX	Compare column	Column or literal		Character string comparison [EQ NE]	81	O	O
Optional	ITER				Repeat DO	84	O	O
Optional	JXWX	Moving axis pattern	Start I/O • flag	PE	JOG [FN FF BN BF]	79	O	
Optional	LEAV				Exit from DO	84	O	O
Optional	LET	Assign variable	Assign number	ZR	Assign	32	O	
Optional	MOD	Assign remainder variable	Calculation number (radian)	ZR	Figure remainder	36	O	O
Optional	MOVD	Move position		PE	Directly designate move	73		DS
Optional	MOVL	Move target position No.		PE	Interpolated move to designated position	74	O	
Optional	MOVP	Move target position No.		PE	Move to designated position	73	O	O
Optional	MULT	Multiply variable	Multiplier number	ZR	Multiplier	35	O	O
Optional	MVDI	Move amount		PE	Incremental movement	74		DS
Optional	MVLI	Move target position No.		PE	Interpolated move to designated position	75	O	
Optional	MVPI	Move target position No.		PE	Move to designated position	75	O	O
Optional	OFST	Default axis pattern	Offset value (mm)		Set offset	67	O	O
Optional	OPEN	Channel No.			SIO open	89	O	O
Optional	OR	Variable to apply logic or to	Calculation number	ZR	Logic or	41	O	O
	OTHE				Declare branching target when conditions are not met	88	O	O
Optional	OUT	First I/O • flag	End input • flag		Binary input (Max 31 bit)	50	O	O
Optional	OUTB	First I/O • flag			BCD input (Max 8 digits)	51	O	O
Optional	OVRD	Velocity ratio (%)			Set velocity factor	64	O	O
Optional	PACC	Acceleration (G)	Assign target position No.		Assign position acceleration	62	O	O
Optional	PATH	Start position No.	End position No.	PE	Path movement	76	O	O
Optional	PAXS	Axis pattern variable No.	Position No.		Read axis pattern	63	O	O
Optional	PCLR	Starting position No.	End position No.		Clear point data	60	O	O
Optional	PCPY	Copy target position No.	Copy source position No.		Copy point data	60	O	O
Optional	PGET	Axis No.	Position No.		Assign position to variable 199	59	O	O
Optional	PPUT	Axis No.	Position No.		Assign position of variable 199	59	O	O
Optional	PRED	Axis pattern to read	Store target position No.		Read current position of axis	61	O	O
Optional	PSIZ	Size variable No.			Check position size	63	O	O
Optional	PTST	Confirmed axis pattern	Confirmed position No.	XX	Confirm position data	61	O	O
Optional	PVEL	Velocity (mm/sec)	Assign target position No.		Assign position velocity	62	O	O

* **Commands not yet publicly available cannot be used.**

Note: The circle (O) in the multiple axes, single axis columns indicates that the command can be used for multiple axes or a single axis. However, MOVD, MVDI are command languages specific to the DS type.

11. Alphabetical List of SEL Language Command Codes

Condition	Command	Operand 1	Operand 2	Output	Function	Page	Multiple Axis	Single Axis
Optional	READ	Read channel	Read column number		SIO output	90	O	O
Optional	*RLRS	Return resource			Acquire resource	58	O	O
Optional	SCHA	Character code			Set ending character	91	O	O
Optional	SCMP	Compare column	Column or literal		Compare character string	92	O	O
Optional	SCPY	Copy target column	Column or literal		Copy character string	92	O	O
Optional	SCRV	Ratio (%)			Set S-motion ratio	66	O	O
Optional	SGET	Send target variable	Send target column		Acquire character	93	O	O
Optional	SIN	Assign sin variable	Calculation number (radian)	ZR	Sine	37	O	O
Optional	SLCT				Declare start of branching	85	O	O
Optional	SLEN	Number of characters to be operated on			Set numbers of characters for operation	97	O	
Optional	*SLIC	Task level	Check value (sec)		Time slice setting	57	O	O
Optional	*SLPG				Task pause	55		DS
Optional	SPUT	Send target column	Write data		Arrange characters	93	O	O
Optional	SQR	Assign square root variable	Calculation number	ZR	Square root	39	O	O
Optional	STOP	Stopped axis pattern		PE	Axis slows to a halt	79	O	O
Optional	*STPR	Task level			Change task level	57		DS
Optional	STR	Conversion target column	Conversion source variable		Convert to decimal character string	94	O	
Optional	STRH	Conversion target column	Conversion source variable		Convert to hexadecimal character string	95	O	O
Optional	SUB	Subtract from variable	Number subtracted	ZR	Subtract	34	O	O
Optional	SVXX	Operating axis pattern			Servo [ON OF]	72	O	O
Optional	TAG	Stated tag No.			Declare jump target	52	O	O
Optional	TAN	Assign tangent variable	Calculation number (radian)	ZR	Tangent	38	O	O
Optional	TIMC	Program No.			Time wait cancel	44	O	O
Optional	TIMW	Wait time (sec)		TU	Time wait	44	O	O
Optional	TRAN	Variable copy destination	Variable to copy	ZR	Copy	32	O	O
Optional	VAL	Conversion target variable	Conversion source column		Convert to decimal value	96	O	O
Optional	VALH	Conversion target variable	Conversion source column		Convert to hexadecimal value	97	O	O
Optional	VEL	Velocity (mm/sec)			Set velocity	64	O	O
Optional	WHXX	Compare variable	Comparison variable		Loop EQ NE GT GE LT LE]	86	O	O
Optional	WRIT					91	O	O
Optional	WSXX	Compare column	Column or literal		Character string branching [EQ NE]	87	O	O
Optional	WTXX	I/O • flag	Wait time	TU	Input • flag [ON OF] wait	47	O	O
Optional	*WUPG	Start up Program No.		CP	Other task startup	55	O	O

* Commands not yet publicly available cannot be used.

Note: The circle (O) in the multiple axes, single axis columns indicates that the command can be used for multiple axes or a single axis. However, MOVD, MVDI are command languages specific to the DS type.

12. SEL Language

12.1 Numeric calculations commands

● LET (Assign)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	LET	Variable No.	Data	ZR

[Function] Assigns the value in operand 2 to the variable in operand 1.
When 0 is assigned to the variable in Operand 1, the output turns ON.

[Example 1] LET 1 10 Assign a value of 10 to variable register 1.

[Example 2] LET 1 2 Assign 2 to variable 1.
LET 3 10 Assign 10 to variable 3.
LET *1 *3 Assign 10 (content of variable 3) to variable 2 (content of variable 1).

● TRAN (Transfer)

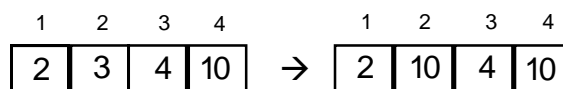
Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	TRAN	Variable No.	Variable No.	ZR

[Function] Assigns the contents of the variable in Operand 2 to the variable in Operand 1.
This function is also known as "indirect addressing" or "pointing to a pointer."

[Example 1] TRAN 1 2 Assign the content of variable 2 to variable 1.

[Example 2] LET 1 2 Assign 2 to variable 1.
LET 2 3 Assign 3 to variable 2.
LET 3 4 Assign 4 to variable 3.
LET 4 10 Assign 10 to variable 4.
TRAN 1 *3 Assign 10 (variable 4 which is the content of variable 3) to the variable for 2.

The variables change in the following manner.



12. SEL Language

● CLEAR (Clear Variables)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	CLR	Variable No.	Variable No.	ZR

[Function] Clears the variables from the variable in operand 1 to the variable in operand 2.
 The contents of the cleared variables becomes 0.
 When 0 is assigned to the variable in operand 1, the output turns ON.

[Example 1] CLR 1 5 Clear variables from 1 to 5.

[Example 2] LET 1 10 Assign 10 to variable 1.
 LET 2 20 Assign 20 to variable 2.
 CLR *1 *2 Clear variables from 10 (content of variable 1) to 20 (content of variable 2).

12. SEL Language

12.2 Arithmetic calculation commands

● ADD (Add)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	ADD	Variable No.	Data	ZR

[Function] Adds the value in operand 2 to the contents of the variable in operand 1, then stores this in the variable in operand 1. The output turns ON when the result of the operation is 0.

[Example 1] LET 1 3 Assign 3 to variable 1.
 ADD 1 2 Add 2 to 3 (content of variable 1).
 3+2 is 5 which is entered in variable 1.

[Example 2] LET 1 2 Assign 2 to variable 1.
 LET 2 3 Assign 3 to variable 2.
 LET 3 2 Assign 2 to variable 3.
 ADD *1 *3 Add 2 (content of variable 3) to variable 2 (content of variable 1).
 3+2 is 5 which is entered in variable 2.

● SUB (Subtract)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	SUB	Variable No.	Data	ZR

[Function] Subtracts the value in operand 2 from the contents of the variable in operand 1, then stores this in the variable in operand 1. The output turns ON when the result of the operation is 0.

[Example 1] LET 1 3 Assign 3 to variable 1.
 ADD 1 2 Subtract 2 from 3 (content of variable 1).

[Example 2] LET 1 2 Assign 2 to variable 1.
 LET 2 3 Assign 3 to variable 2.
 LET 3 2 Assign 2 to variable 3.
 ADD *1 *3 Subtract 2 (content of variable 3) from variable 2
 (content of variable 1).
 3-2 is 1 which is entered in variable 2.

12. SEL Language

● MULT (Multiply)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	MULT	Variable No.	Data	ZR

[Function] Multiplies the contents of the variable in operand 1 by the value in operand 2, then stores this in the variable in operand 1. The output turns ON when the result of the operation is 0.

[Example 1] LET 1 3 Assign 3 to variable 1.
 MULT 1 2 Multiply 3 (content of variable 1) by 2.

[Example 2] LET 1 2 Assign 2 to variable 1.
 LET 2 3 Assign 3 to variable 2.
 LET 3 2 Assign 2 to variable 3.
 MULT *1 *2 Multiply variable 2 (content of variable 1) by 2
 (content of variable 1).
 3x2 is 6 which is entered in variable 2.

● DIV (Divide)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	DIV	Variable No.	Data	ZR

[Function] Divides the contents of the variable in operand 1 by the value in operand 2, then stores this in the variable in operand 1. The output turns ON when the result of the operation is 0.

Note: When operand 1 is an integer type variable, anything beyond the decimal point is disregarded.

[Example 1] LET 1 3 Assign 6 to variable 1.
 DIV 1 2 Divide 6 (content of variable 1) by 2.

[Example 2] LET 1 2 Assign 2 to variable 1.
 LET 2 6 Assign 3 to variable 2.
 LET 3 2 Assign 2 to variable 3.
 MULT *1 *3 Divide variable 2 (content of variable 1) by 2
 (content of variable 1).
 6÷2 is 3 which is entered in variable 2.

12. SEL Language

● MOD (Remainder)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	MOD	Variable No.	Data	ZR

[Function] Divides the contents of the variable in operand 1 by the value in operand 2, then stores the remainder in the variable in operand 1. The output turns ON when the result of the operation is 0.

Note: The MOD command is used with respect to an integer type variable.

[Example1] LET 1 7 Assign 7 to variable 1.
 MOD 1 3 Figure the remainder when 7 (content of variable 1) is divided by 3.
 7 ÷ 3 is 2 with a remainder of 1 which is entered in variable 1.

[Example 2] LET 1 2 Assign 2 to variable 1.
 LET 2 3 Assign 7 to variable 2.
 LET 3 3 Assign 3 to variable 3.
 MOD *1 *2 Figure the remainder when variable 2 (content of variable 1).
 7 ÷ 3 is 2 with a remainder of 1 which is entered in variable 1.

12. SEL Language

12.3 Functional calculation commands



● SIN (Sine)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	SIN	Variable No.	Data	ZR

[Function] Assigns the sine of operand 2 to the contents of the variable in operand 1. The output turns ON when the result of the operation is 0. For the setting in operand 1, designate a real number variable in the operand 100~199, 300~399.

Note 1: $Radian = Angle \times \pi \div 180$

[Example 1] SIN 100 0.523599 Assign 0.5, sine of 0.523599, to variable 100.

[Example 2] LET 1 100  Assign 100 to variable 1.
 LET 101 30  $30 \times \pi \div 180$ (radian) (convert 30° to radian and assign this to variable 101).
 MULT 101 3.141592
 DIV 101 180 Assign 0.5, sine of contents in variable 101 to
 SIN *1 *101 variable 100 (contents of variable 1).

● COS(Cosine)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	COS	Variable No.	Data	ZR

[Function] Assigns the cosine of operand 2 to the contents of the variable in operand 1. The output turns ON when the result of the operation is 0. For the setting in operand 1, designate a real number variable in the range 100~199, 300~399.

Note: $Radian = Angle \times \pi \div 180$.

[Example 1] SIN 100 1.047197 Assign 0.5, sine of 1.047191 to variable 100.

[Example 2] LET 1 100 Assign 100 to variable 1.
 LET 101 30 $60 \times \pi \div 180$ (radian) (convert 60° to radian and
 MULT 101 3.141592 assign this to variable 101).
 DIV 101 180 Assign 0.5, cosine of contents in variable 101 to
 SIN *1 *101 variable 100 (contents of variable 1).

12. SEL Language

● TAN (Tangent)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	TAN	Variable No.	Data	ZR

[Function] Assigns the tangent of operand 2 to the contents of the variable in operand 1. The output turns ON when the result of the operation is 0. For the setting in operand 1, designate a real number variable in the range 100~199, 300~399. The unit in operand 2 is radians.

Note 1: $Radian = Angle \times \pi \div 180$

[Example 1] TAN 100 0.785398 Assign 1, tangent of 0.785398, to variable in 100.

[Example 2] LET 1 100 Assign 100 to variable 1.
 LET 101 45 $45 \times \pi \div 180$ (radian) (convert 45° to radian and
 MULT 101 3.141592 assign this to variable 101).
 DIV 101 180 Assign 1, tangent of contents in variable 101 to
 TAN *1 *101 variable 100 (contents of variable 1).

● ATN (Arctangent)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	ATN	Variable No.	Data	ZR

[Function] Assigns the arctangent of operand 2 to the contents of the variable in operand 1. The output turns ON when the result of the operation is 0. For the settings in operand 1, designate a real number variable in the range 100~199, 300~199. The unit in operand 2 is radians.

Note 1: $Radian = Angle \times \pi \div 180$

[Example 1] ATN 100 1 Assign 0.785398, arctangent of 1, to variable 100.
 Assign 100 to variable 1.

[Example 2] LET 1 100 Assign 1 to variable 101.
 LET 101 1 Assign 0.785398, arctangent of contents in variable 101
 TAN *1 *101 to variable 100 (contents of variable 1).

12. SEL Language

12.4 Logic and Operation commands

● AND (Logic And)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	AND	Variable No.	Data	ZR

[Function] Stores the results of the logic AND operation on the contents of the variable in operand 1 and the in operand 2, in the variable in operand 1. The output turns ON when the result of the operation is 0.

[Example 1] LET 1 204 Assign 204 to variable 1.
 AND 1 170 Assign 136 the result of logic AND on 204 (contents of variable 1)
 AND 170 (data in operand 2), to vaiable 1.

[Example 2] LET 1 2 Assign 2 to variable 1.
 LET 2 204 Assign 204 to variable 2.
 LET 3 170 Assign 170 to variable 3.
 AND *1 *3 Assign 136, the result of logic AND on 204(content of variable 2 which is
 the conteng in variable 1) and 170 (content of variable 3), to 2(the content
 of variable 1).

Decimal number	Binary number
204	11001100
<u>AND 170</u>	<u>AND 10101010</u>
136	10001000

12. SEL Language

● OR (Logic Or)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	OR	Variable No.	Data	ZR

[Function] Stores the results of the logic OR operation on the contents of the variable in operand 1 and the data in operand 2, in the variable in operand 1. The output turns ON when the result of the operation is 0.

[Example 1]

LET	1	204	Assign 204 to variable 1.
OR	1	170	Assign in 238, the result of logic AND on 204 (contents of variable 1) and 170 (data in operand 2), to variable 1.

[Example 2]

LET	1	2	Assign 2 to variable 1.
LET	2	204	Assign 2 to variable 1.
LET	3	170	Assign 170 to variable 3.
OR	*1	*3	Assign 238, the result of logic OR on 204 (content of variable 2 which is the content in variable 1) and 170 (content of variable 3), to 2(the content of variable 1).

Decimal number	Binary number
204	11001100
<u>OR 170</u>	<u>OR 10101010</u>
238	10001000

12. SEL Language

● EOR (Exclusive Or Logic)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	EOR	Variable No.	Data	ZR

[Function] Stores the results of the exclusive logic operation on the contents of the variable in operand 1 and the data in operand 2, in the variable in operand 1. The output turns ON when the result of the operation is 0.

[Example 1]

LET	1	204	Assign 204 to variable 1.
EOR	1	170	Assign 102 in, the result of exclusive logic of 204 (contents of variable 1) and 170 (data in operand 2), to variable 1.

[Example 2]

LET	1	2	Assign 2 to variable 1.
LET	2	204	Assign 2 to variable 1.
LET	3	170	Assign 170 to variable 3.
EOR	*1	*3	Assign 102, the result of logic OR on 204 (content of variable 2 which is the content in variable 1) and 170 (content of variable 3), to 2 (the content of variable 1).

Decimal number	Binary number
204	11001100
<u>EOR 170</u>	<u>EOR10101010</u>
102	0110110

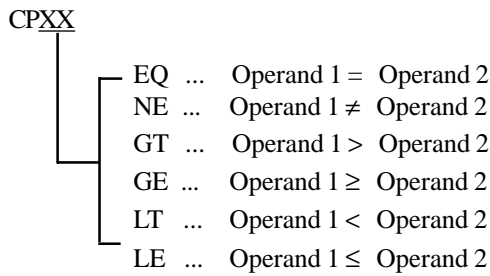
12. SEL Language

12.5 Comparison operation commands

● CPXX (Compare)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	CPXX	Variable No.	Data	EQ NE GT GE LT LE

[Function] Compares the contents of the variable in operand 1 and the value in operand 2 and if the condition is satisfied, the output turns ON. When the condition is not satisfied, the output turns OFF.



- [Example 1]
- | | | | | |
|----------|---|----|-----|--|
| LET | 1 | 10 | | Assign 10 to variable 1. |
| 600 CPEQ | 1 | 10 | 600 | If the content of variable 1 is 10, flag 600 turns ON, |
| ADD | 2 | 1 | | If flag 600 is ON, 1 is added to variable 2. |
- [Example 2]
- | | | | | |
|------|----|----|-----|---|
| LET | 1 | 2 | | Assign 2 to variable 1. |
| LET | 2 | 10 | | Assign 10 to variable 2. |
| LET | 3 | 10 | | Assign 10 to variable 3. |
| CPNE | *1 | *3 | 310 | If the variable in 2 (the content of variable 1) does not equal the content of variable 3, then output 310 turns ON. Therefore, in this example, output 310 is OFF. |

12. SEL Language

12.6 Timer Commands

● TIMW (Timer)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	TIMW	TIME		TU

[Function] The program stops and waits for the time set in operand 1.
 Setting range is 0.01 ~ 99 and units are seconds.
 When the designated time has elapsed and the program moves to the next step, the output turns ON.

[Example 1] TIMW 1.5 Wait for 1.5 seconds.

[Example 2] LET 1 10 Assign 10 to variable 1.
 TIMW *1 Contents of variable 1 waits 10 seconds.

● TIMC (Timer Cancel)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	TIMC	Program No.		

[Function] Cancels the time of the other programs designated in operand 1 that are running in parallel.

[Example 1] TIMC 10 Cancel time wait for program 10.

[Example 2] LET 1 10 Assign 10 to variable 1.
 TIMC *1 Cancel time wait for program 10 (content of variable 1).

12. SEL Language

● GTTM (Time Acquisition)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	GTTM	Variable No.		

[Function] Writes the system time to the variable in operand 1. The time unit is 10msec.
The time obtained with this command is a value that has no base. Therefore, call this command twice, and the difference gives the time that has elapsed.

[Example 1]

GTTM	1		Read the reference time to variable 1.
ADD	1	500	Set the ending time for 5 seconds later.
GTTM	2		Read the current system time to variable 2.
DWGE	1	*2	After 5 seconds, proceeds to the next step after EDDO.
:			The processing during this time will repeat for 5 seconds.
:			
GTTM	2		Read the current system time to variable 2.
EDDO			

[Example 2]

LET	1	5	Assign 5 to variable 1.
GTTM	*1		Store the current system time in the variable for 5 (content of variable 1).

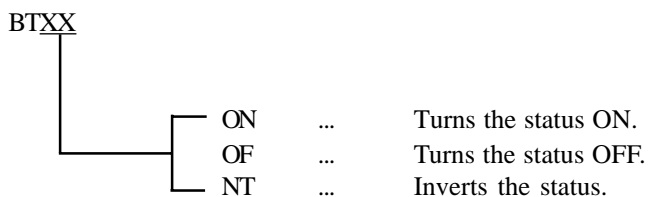
12. SEL Language

12.7 I/O · Flag operation commands

● BTXX (Output Port · Flag Operation)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	BTXX	Output · Flag	Optional	

[Function] Turns ON, OFF, or inverts from the output · flag designated in operand 1 to the output · flag designated in operand 2.



[Example 1] BTON 300 Output port 300 turns ON.

[Example 2] BTOF 300 307 Output port 300~307 turns OFF.

[Example 3] LET 1 600 Assign 600 to variable 1.
 BTNT *1 Invert flag 600 (content of variable 1).

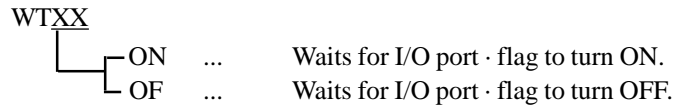
[Example 4] LET 1 600 Assign 600 to variable 1.
 LET 2 607 Assign 607 to variable 2.
 BTON *1 *2 Turns flags from 600 (content of variable 1) to 607 (content of variable 2) ON.

12. SEL Language

● WTXX (I/O Port • Flag Wait)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	WTXX	I/O · Flag	Optional Time out	TU

[Function] Program waits until designated in operand 2 turns ON/OFF.
 Can abort the wait after a set time by designating a time in operand 2.
 Setting range is 0.01~99 seconds. After a set time has elapsed, the output turns ON (Only where is an operand 2).



[Example 1] WTON 15 Waits for I/O port 15 to turn ON.

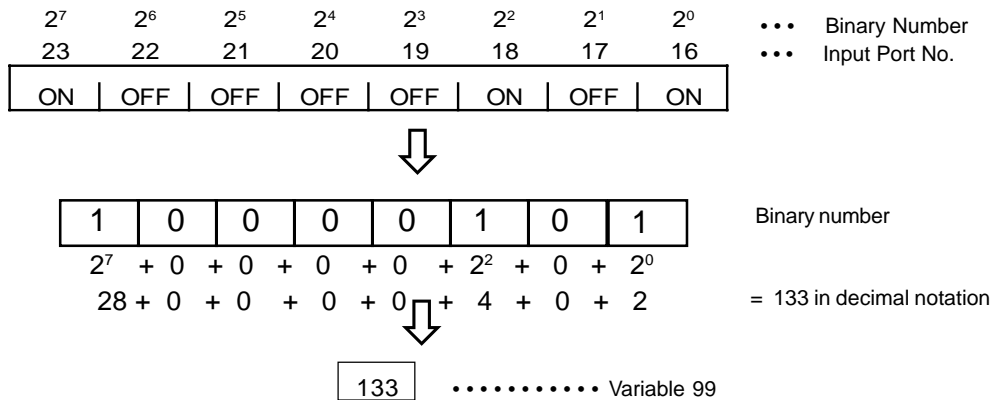
[Example 2] WTOF 25 5 900 Wait for input 25 to turn ON. If not ON within 5 seconds, turn flag 900 ON and proceed to the next step.

12. SEL Language

● IN (Binary Number Read I/O • Flag)

Expansion condition (AND • OR)	Input condition (I/O • Flag)	Command			Post (Output • Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	IN	I/O port-Flag	I/O port-Flag	

[Function] Reads the value from the designated I/O port or flag as a binary number, then stores this value in variable register 99.



Note: The maximum input limit for the port is 31 consecutive bits.

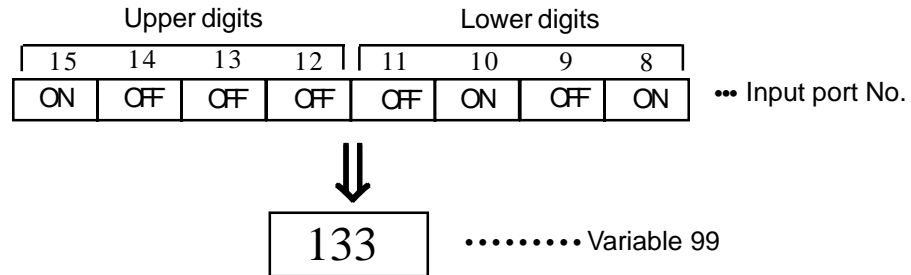
- [Example 1] IN 8 15 Read input ports 8 ~15 as a binary number in variable 99.
- [Example 2] LET 1 8 Assign 8 to variable 1.
 LET 2 15 Assign 15 to variable 2.
 IN *1 *2 Read port 8(content of variable 1) to port 15
 (content of variable 2) as a binary number to variable 99.

12. SEL Language

● INB (BCD Read I/O • Flag)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	INB	Input port	No. of BCD digits	

[Function] Reads the BCD value from the designated input port, then stores this value in variable register 99.



Note 1: The maximum number of digits that can be input is 8 (32 bits).

Note 2: The I/O Port • Flag used is 4 x n (number of digits).

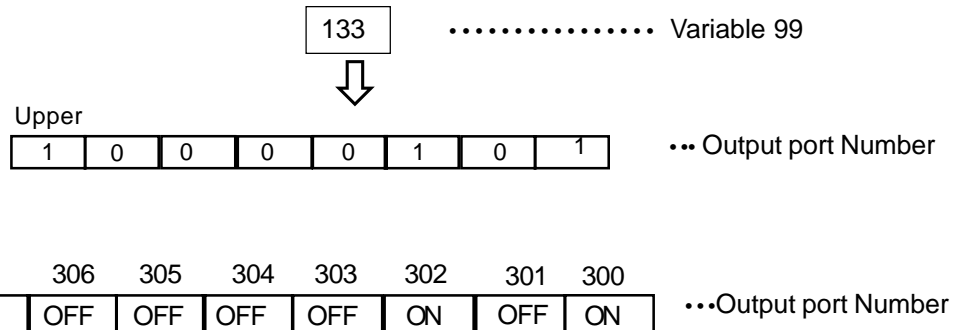
- | | | | | |
|-------------|-----|----|----|--|
| [Example 1] | INB | 8 | 2 | Read input port from 8 for 2 digits (up to 15) as a binary number to variable 99. |
| [Example 2] | LET | 1 | 8 | Assign 8 to variable 1. |
| | LET | 2 | 2 | Assign 2 to variable 2. |
| | IN | *1 | *2 | Read from input port 8 (content of variable 1) for 2 digits (content of variable 2)(up to 15) as a BCD value to variable 99. |

12. SEL Language

● OUT (Binary Number Output)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	OUT	I/O · Flag	I/O · Flag	

[Function] Output the value of variable 99 to output ports or flags from operand 1 to operand 2.



Note 1: The maximum number of digits that can be output is 32 bits.

[Example 1] OUT 300 307 Write the value of variable 99 as a binary value to output ports from 300~307.

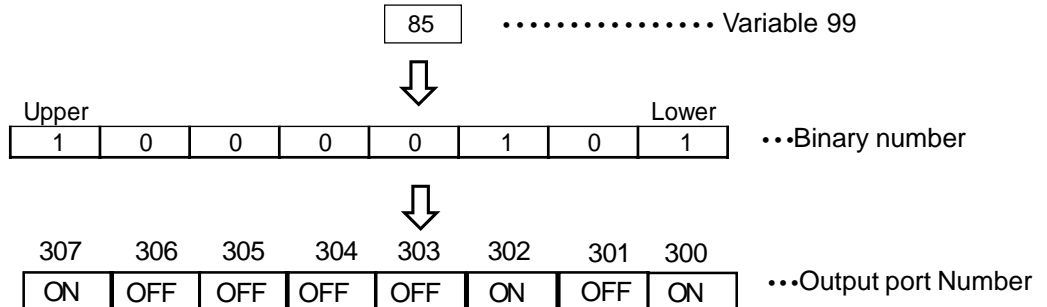
[Example 2] LET 1 300 Assign 300 to variable 1.
 LET 2 307 Assign 307 to variable 2.
 OUT *1 *2 Write the value of variable 99 as a binary number to output ports 300 (content of variable 1) through 307 (content of variable 2).

12. SEL Language

● OUTB (BCD Output)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	OUTB	I/O · Flag	No. of BCD digits	

[Function] Output the value of variable 99 to output ports or flags from operand 1 to operand 2.



Note 1: The maximum number of digits that can be output is 8 (32 bits).

Note 2: The output port · flag used is 4 x n (number of digits).

[Example 1] OUT 300 2 Write the value of variable 99 as a BCD value to output ports from 300 for 2 digits (up to 307).

[Example 2] LET 1 300 Assign 300 to variable 1.
 LET 2 2 Assign 2 to variable 2.
 OUT *1 *2 Write the value of variable 99 as a BCD value to output ports from 300 (content of variable 1) for 2 digits (content of variable 2) (up to 15).

12. SEL Language

● GOTO (Jump)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	GOTO	Tag No.		

[Function] Jumps to the position of the tag number designated in operand 1.

Note: The GOTO command is valid only within the same program.

[Example 1] TAG 1 Set the tag.
 :
 :
 :
 GOTO 1 Jump to tag 1.

[Example 2] LET 1 10 Assign 10 to 1.
 GOTO *1 Jump to tag 10 (content of variable 1).

● TAG (Tag Declaration)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	TAG	Tag No.		

[Function] Sets the tag number designated in operand 1.

[Example] Refer to the GOTO command.

12. SEL Language

● EXSR (Execute Subroutine)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output · Flag)
		Command	Operand 1	Operand 2	
		EXSR	Subroutine No.		

[Function] Executes the subroutine number designated in operand 1.

Note: Only a subroutine number within the same program is enabled.

```
[Example1]      EXSR 1           Execute subroutine 1.
                  :
                  :
                  EXIT
                  BGSR 1         Begin subroutine 1
                  :
                  :
                  :
                  EDSR           End subroutine 1.
```

```
[Example 2]      LET    1    10   Assign 10 to 1
                  EXSR   *1     Execute subroutine 10 (content of variable 1).
```

● BGSR (Begin Subroutine)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output · Flag)
		Command	Operand 1	Operand 2	
		BGSR	Subroutine No.		

[Function] Commands the start of the subroutine number designated in operand 1.

[Example] Refer to the EXSR command.

● EDSR (End Subroutine)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
		EDSR			

[Function] Commands end of subroutine. This is always required at the end of a subroutine. After this, the program moves to the step after the EXSR called out.

[Example 1] Refer to the EXSR command.

12. SEL Language

12.9 Task management commands

● EXIT (Exit Program)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	EXIT			

[Function] Finishes the program. When final step is reached without the EXIT command, process returns to the front.

* The status when the program is complete

- Output Port Valid
- Local Flag Invalid
- Local Variable Invalid
- Current Value Valid
- Global Flag Valid
- Global Variable Valid

[Example] :
 :
 EXIT Ends the program.

● EXPG (Start Another Program)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	EXPG	Program No.		CP

[Function] Starts another program and processes it in parallel.
 When that program (task) has been started, the port and flag in the post section is output.

[Example 1] EXPG 10 Start program number 10.

[Example 2] LET 1 10 Assign 10 to variable 1.
 EXPG *1 Start variable 1 (content 10) program.

12. SEL Language

● ABPG (Stop Other Program)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	ABPG	Program No.		CP

[Function] Forces the other program being executed in operand 1 to end.
When that program (task) is forced to end, the port and flag in the post section is output.

Note: The ABPG command stops the program once the command being executed is completed.

[Example 1] ABPG 10 Stop program No. 10.

[Example 2] LET 1 10 Assign 10 to variable 1.
ABPG *1 Stop variable 1 (content 10) program.

● SLPG (Task Pause) * *Commands not yet publicly available can not be used.*

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	SLPG			

[Function] Pauses self task, and begins execution again by WUPG command of another program.

[Example 1] SLPG Pauses self task.

● WUPG (Startup Other Task) * *Commands not yet publicly available can not be used.*

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	WUPG	Program No.		CP

[Function] Executes programs paused by the SLPG command assigned in operand 1.
Once the startup is successful, output turns ON.

[Example 1] WUPG 1 Execute program No. 1.

[Example 2] LET 1 10 Assign 10 to variable 1.
WUPG *1 Execute variable1 (content 10) program.

12. SEL Language

● GTPG (Acquire Task Level) * Commands not yet publicly available can not be used.

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	GTPG	Variable No.	Program No.	

[Function] Reads the task condition of operand 2 program into operand 1 variable.
The value being read is as follows:

Execute condition	...	TTS__RUN	01
Execute able condition	...	TTS__RDY	02
Wait condition	...	TTS__WAI	04
Forced wait condition	...	TTS__SUS	08
Double wait condition	...	TTS__RDY	0C (12) WAI + SUS (4+8)
Pause conditon	...	TTS__RDY	10 (16)

[Example 1] GTPG 10 5 Read task condition of program 5 into variable 10.

[Example 2] LET 1 10 Assign 10 to variable 1.
LET 2 5 Assign 5 to variable 1.
GTPG*1 *2 Read task condition of variable 2 (content 5) program into variable 1 (content 10 variable).

● GTPR (Acquire Task Level) * Commands not yet publicly available can not be used.

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	GTPR	Variable No.		CP

[Function] Stores task level of self task to operand 1 variable.

[Example 1] GTPR 1 Store task level into variable 1.

[Example 2] LET 1 10 Assign 10 to variable 1.
GTPR *1 Store task level into variable 1 (content 10).

12. SEL Language

● STPR (Task Level Change) * Commands not yet publicly available can not be used.

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	STPR	Task level		

[Function] Change the task level of the self task to the value of operand 1. The task level range is 1~5, smaller number receiving priority.

[Example 1] STPR 1 Change task level to 1.

[Example 2] LET 1 Assign variable 1 to 3.
STPR *1 Change the task level of variable 1 (content 3).

● SLIC (Time Slice Value Change) * Commands not yet publicly available can not be used.

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	SLIC	Task level	Time slice	

[Function] Changes task level of operand 1 to the check value of operand 2. Assigns task level range from 1~5 and check value at 10mm/sec per unit.

[Example 1] SLIC 2 0.02 Change the check value of task level 2 to 0.02 sec.

[Example 2] LET 100 Assign 0.5 to variable 100.
SLIC 3 *100 Changes the task data of task level 3 to variable 100 (content 0.5 sec).

12. SEL Language

12.10 Resource management commands

● **GTRS (Obtain Resource)** *Commands not yet publicly available can not be used.

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	GTRS	Resource No.		

[Function] Obtains resource. If resource is not available, program pauses until the resource is released.
Assigns resource no. 1~9.

[Example 1] Define X axis as resource 1.

Program 1	Program 2	
GTRS 1	:	Program 1 obtain resource 1.
MOVL 1	GTRS 1	Program 2 can not obtain resource 1.
RLRS 1	(Obtain wiat)	As resource 1 returns, Program 2 obtains it.
:	MOVP 2	Program 2 uses the X axis.
:	RLRS 1	Program 2 returns the resource.

By doing the above, even when the same axis is used for multiple programs, an error will not be encountered. .

[Example 2] LET 1 5 Assign 5 to variable 1.
GTRS *1 Obtain resource of variable 1 (content 5).

● **RLRS (Return Resource)** * Commands not yet publicly available can not be used.

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	RLRS	Resource No.		

[Function] Returns the obtained resources.

[Example 1] Please refer to GTRS.

12. SEL Language

● PCLR (Position Data Clear)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	PCLR	Position No.	Position No.	

[Function] Clears the data in the range of positions designated by operand 1 and operand 2 (becomes XX.XXX, not 0.00).

[Example] PCLR 10 20 Clears data from position 10 in operand 1 through position 20 in operand 2.

● PCPY (Position Data Copy)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	PCPY	Position No.	Position No.	

[Function] Copies data in the designated position No. (copy data in operand 2 to operand 1).

[Example] PCPY 20 10 Copy data from position 10 in operand 2 to position 20 in operand 1.

12. SEL Language

● PRED (Read Coordinates)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	PRED	Axis pattern	Position No.	

[Function] Reads the current coordinates of the axis designated in operand 1 and writes it to the position designated in Operand 2.

[Example] PRED 11 10 Read the current coordinates of axis 1 and axis 2 designated in operand 1 into position 10.

● PTST (Check Position Data)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	PTST	Axis pattern	Position No.	Required

[Function] Checks to see whether there is valid data in the designated axis pattern and position number. If there is no data, the post flag or output port turns ON. Post section turns ON only when all the axes specified by the axis pattern are XX.XXX. ("0" is considered as data.)

[Example] PTST 11 11 600 If there is no data in position 11 of axis 1 and 2, flag 600 turns ON.

12. SEL Language

● PVEL (Assign Velocity Data)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	PVEL	Velocity	Position No.	

[Function] Assigns the value in operand 1 as the velocity for the designated position data. Variables can also be used. This command is used to change the actual velocity.

* When a value is assigned that will result in a negative number after calculation, there is no warning at the time you execute this command but an alarm will occur when you try to use the data.

[Example] PVEL 100 3 Assign a value of 100mm/sec to the velocity data for position number 3.

● PACC (Assign Acceleration Data)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	PACC	Acceleration	Position No.	

[Function] Assigns the acceleration data in Operand 1 for the acceleration of the position data. As with the PVEL command, you can assign a value using the variable but there is no function that checks the value range when executing this command. Therefore, please be careful not to assign a value that exceeds the actuator's limits.

[Example] PACC 0.3 3 Assign a value of 0.3 to the acceleration speed data for position number 3.

12. SEL Language

● PSIZ (Check Position Data Size)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	PSIZ	Variable No.		

[Function] Checks the maximum size of the position data that can be used.

[Example] PSIZ 1 The maximum value of the position data goes into variable 1 (variable to be assigned) in operand 1.

● PAXS (Read Axis Pattern)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	PAXS	Variable No.	Position No.	

[Function] Stores the axis pattern of the position in operand 2 into the variable in operand 1.

[Example 1] PAXS 100 200 Store the axis pattern of position 200 into variable 100. When the points are set as in the position table below, 2 (10 in binary notation) is stored in variable 100.

[Example 2] When the points are set as below, 2 (10 in binary notation) is stored in variable 100.

```
LET 1 3 Assign 3 to variable 1.
PAXS *1 *2 Assign 101 to variable 2.
LET 2 301 Store the axis pattern of the position for 101 which is the value contained in
variable 2, to variable 3 which is contained in variable 1. When the points are
set as below, 3 (11 in binary notation) is stored in variable 3.
```

12. SEL Language

● VEL(Velocity)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	VEL	Velocity		

[Function] Sets the velocity of an actuator movement in mm/sec. The maximum velocity varies according to the model of the actuator so please set below that value.

*Decimal places cannot be used. Entering a decimal value will cause an error.

*The minimum velocity setting is 1mm/sec.

[Example] VEL 1000
1000mm/sec (Velocity Setting)

```

VEL1000
MOVP 1  ] The velocity between these two points is 1000mm/sec.
MOVP 2  ]
VEL 500
MOVP 3  ] The velocity between these two points is 500mm/sec.
MOVP 4  ]
    
```

● OVRD (Override)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	OVRD	Velocity ratio value		

[Function] This command decreases the velocity according to the designated ratio. (Velocity coefficient setting). The range of the ratio settings is from 1 ~100%.

*When you use the override function, any value below 1 will be clamped at 1. Any decimal value in the speed setting will be rounded off.

[Example] VEL 100 100mm/sec setting.
OVRD 50 100mm/sec is reduced by 50% and the actual velocity becomes 50mm/sec.

12. SEL Language

● ACC (Acceleration)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	ACC	Acceleration		

[Function] Sets the acceleration of the actuator movement which is expressed in G (Gravity). The maximum acceleration varies and depends on the actuator model and payload. The rated acceleration is 0.3 G. The actuator moves at the rated acceleration 0.3 G when acceleration is not set by the ACC command.

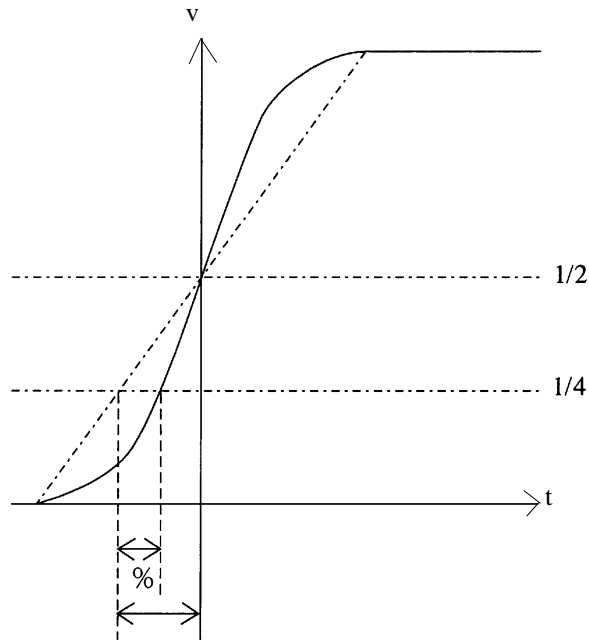
[Example] ACC 0.3
 0.3G (Acceleration setting at 0.3G)

12. SEL Language

● SCR (S Motion Ratio Setting)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	SCRV	Ratio		

[Function] Sets the ratio to control the S motion of the actuator.
 The setting range is integers from 0 ~ 50 (%).
 If this command is not used to set the ratio or when it is set to 0 (%), the actuator makes a trapezoid motion.



[Example 1] SCR 30 S motion ratio is set to 30%.

[Example 2] LET 1 50 Assign 50 to variable 1.
 SCR *1 S motion ratio is set to 50 (%) which is the content of variable 1.

12. SEL Language

● ATRG (Arch Motion Trigger)

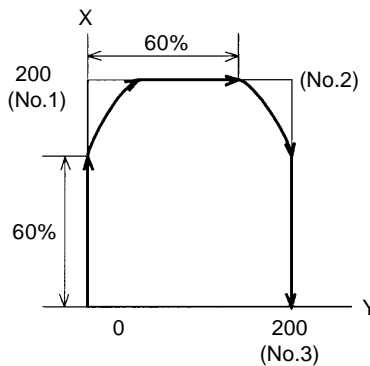
Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	ATRG	Position ratio (%)		

[Function] Sets the axis movement position ratio to execute the ARCH command.

*The position ratio depends on the distance of the movement but it should be set at 50~60% or higher. When the ratio is set too low, a "C2" alarm may occur.

[Example] Application program

```
HOME 11
VEL 100
ATRG 60
ACC 0.3
ARCH 1 3
```



Position Data	X Axis	Y Axis
No. 1	200.000	xxx.xxx
No. 2	xxx.xxx	200.000
No. 3	0.00	xxx.xxx

● OFST (Offset)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	OFST	Axis pattern	Offset value	

[Function] This command adds an offset value to the target value when the actuator moves. The offset amount is given in mm and the resolution is 0.001mm. Offset values can be negative numbers within the range of movement.

* The OFST command can only be used for the axes in that program. To set an offset value for axes in multiple programs, the OFST command must be executed for each program.

[Example] OFST 1000011 50.000 50mm is added to the movement amount of Axis 1, Axis 2, and Axis 8.

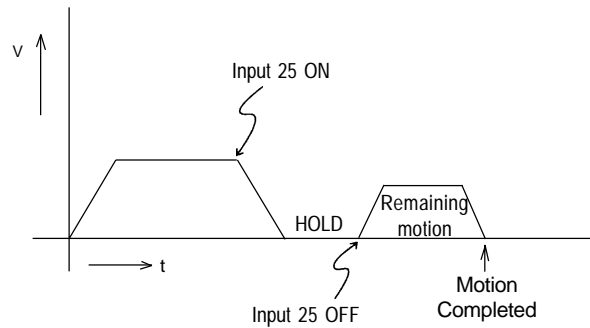
12. SEL Language

● HOLD(Hold: Axis Temporary Stop)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	HOLD	I/O · Flag		

[Function] Designates an input port for sending a command to decelerate and stop while a move command is being executed. If the designated input port turns ON, then velocity decreases until all motion stops. When the input port turns OFF, then motion begins again. The HOLD command applies only to the axes in the designated task (program), and does not affect axes running in other programs.

[Example] HOLD 25 When input port 25 turns ON, velocity decreases until all motion stops.



- * When the HOLD function is used during PATH, CIR, ARC motion commands, the actuator stops at the next position. During the execution of straight line motion commands such as MOVL, MOVP, it stops immediately.
- * The IA system uses a unique homing sequence which locks the servo and detects the stroke edge during homing. If the HOLD is activated at the end of homing, this might cause a "servo run-away = alarm" after the HOLD is released. Therefore, HOLD should be designated after the HOME command. If you need to designate HOLD from the beginning, a home area detection switch (an area limit switch) must be installed so that the HOLD designation will not be carried out in this area.
- * HOLD and CANCEL cannot be used in the same program. (If both are written in the same program, the command that is designated later is the one that becomes effective).

12. SEL Language

● CANCEL (Cancel: Cancels Remainder of Move)

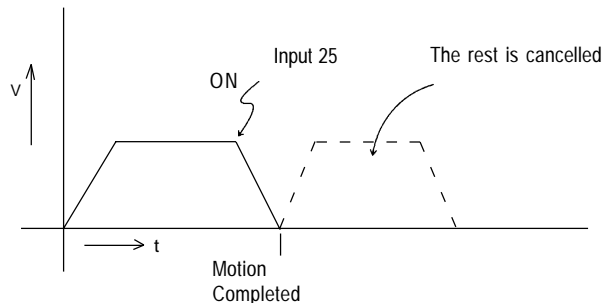
Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	CANC	I/O · Flag		

[Function] Designates an input or flag for sending a command to decelerate and stop while a move command is being executed. If the actuator is moving and the designated input port turns ON, then velocity decreases until all motion stops. Any other programmed motion thereafter is cancelled and not executed.

* HOLD and CANCEL cannot be used in the same program. (If both are written in the same program, the command that is designated later is the one that becomes effective).

[Example] CANCEL 25

When input port 25 turns ON, velocity decreases until all motion stops. All motion after this is cancelled.



* During PATH, CIR, ARC motion designation, the actuator moves to the next position. Any other programmed motion thereafter is cancelled and not executed.

● BASE (Axis Base Designation)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	BASE	Axis No.		

[Function] Count axes starting with the designated axis as the first axis.

[Example] HOME 11 Axis No.1 and Axis No.2 perform homing.
 BASE 3 Axis No.3 is counted as the first axis.
 HOME 11 Axis No.3 and Axis No.4 perform homing.
 After homing, Axis No.3~8 move by designating Axis No.1~6
 (axis pattern and position data).

12. SEL Language

●GRP (Grouping of Axes)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	GRP	Axis pattern		

[Function] This command moves the actuator through the position data of the designated axis pattern.
(Even if there is data in axes other than those designated, the actuator will not move to these positions).

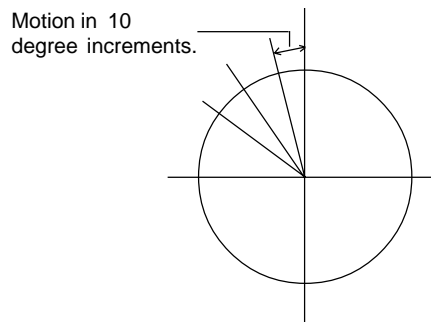
[Example] GRP 0000011 From position data for 8 axes, data from axis 1 and 2 is taken out and executed.

●DEG (Degree Setting)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	DEG	Angle		

[Function] This command sets up the motion increments for use with CIR (Circular Movement) and ARC (Arc Movement) commands. When performing CIR and ARC commands, passing points will be calculated by dividing a circle into the degrees as set. When increments are set small, the circular movement is accurate, however, when they are too small, the speed becomes too slow. When CIR and ARC commands are performed without setting increments, the motion increments of the actuator will be 15 degrees.

[Example] DEG 10



12. SEL Language

●AXST (Axis Status Acquisition)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	AXST	Variable No.	Axis No.	

[Function] Stores the status (error code) of the axis in Operand 2 in the variable in Operand 1. Only error codes that begin with the letter "A" will be stored in the register in Operand 1. These error codes are the same ones that are displayed on the front panel of the controller. (The error codes in the table are written in hexadecimal numbers. The hexadecimal value in the variable in Operand 1 must be converted to a decimal number to identify its error code.)

[Example] AXST 1 2 Read the status for Axis 2 to variable 1.
 If 161 was in variable 1, $161 \div 16 = 10 (= A)$ with a remainder of 1, then this means that error code A1 (External Interrupt Error) occurred on Axis 2.

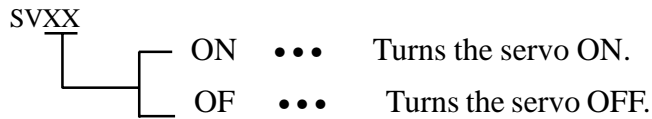
12. SEL Language

12.13 Actuator Control Commands

● SVXX (Servo ON/OFF)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	SVXX	Axis pattern		

[Function] This commands turns the servo of the designated axes ON/OFF .



[Example] SVON SVON 11001100

Turn on the servo for axis 4, 7, 8. This does not affect axes that are already ON.

● HOME (Return Home)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	HOME	Axis pattern		Optional

[Function] This command executes homing of the designated axes. Servos turn ON automatically.

[Example] HOME 10000011 Axis 1, 2, and 8 axes execute homing.

12. SEL Language

● MOVD (Direct Designate Move)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	MOVD	Move position		Optional

[Function] Moves the actuator to a position designated in operand 1. Output turns OFF when axis move starts, and upon completion, turns ON.

[Example 1] MOVD 100 Move the axis to position 100.

[Example 2] LET 1 100 Assign 100 to variable 1.
MOVD *1 Move the axis to variable 1 (content 100) position.

* Note: This command is applicable to the DS type only.

● MOVP (Point to-Point Position Data)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	MOVP	Position no.		Optional

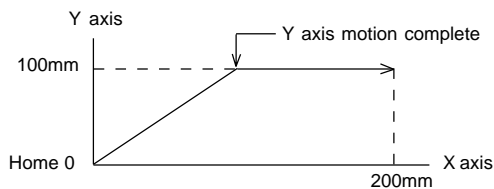
[Function] This command moves the actuator to the designated position number from point to point without interpolation.

[Example] MOVP 100
Moves to Position No. 100 (point to point).

MOVP *1
If variable 1 is 150, then the actuator moves to position number 150 (point to point).

Position No.	1	2	3
1			
2			
3			
⋮			
100	100.0- 0	100.0- 0	xxx.x- x
⋮			
150	200.0- 0	200.0- 0	xxx.x- x
⋮			

Move path when X-axis moves to 200mm point and Y-axis moves to 100mm point from home.



Each axis moves at its own designated speed.

12. SEL Language

● MOVL(Position Data with Interpolation)

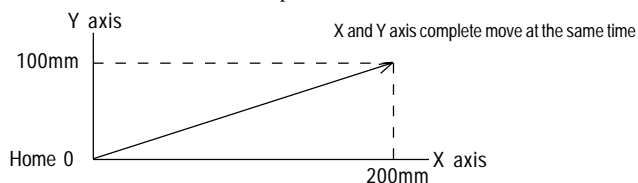
Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	MOVL	Position No. Variable		Optional

[Function] Moves the actuator to the designated point while using interpolation (not point to point).

[Example] MOVL 100
Move to position No. 100 using interpolation.

MOVL *1
If variable 1 is 150, then the actuator moves to position 150 using interpolation.

Move path when X-axis moves to 200mm point and
Y-axis moves to 100mm point from home.



The tip of the combined motion for each of the axes moves at the designated speed. The path from the starting point to the end point makes a straight line.

● MVDI (Incremental Move)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	MVDI	Position Increment		PE

[Function] Moves the as a move load of the value designated in operand 1.
The output becomes OFF when axis move starts, and upon completion, turns ON.

[Example] MVDI 30 Move 30mm from the current position to the +direction.

* Note: This command is applicable to the DS type only.

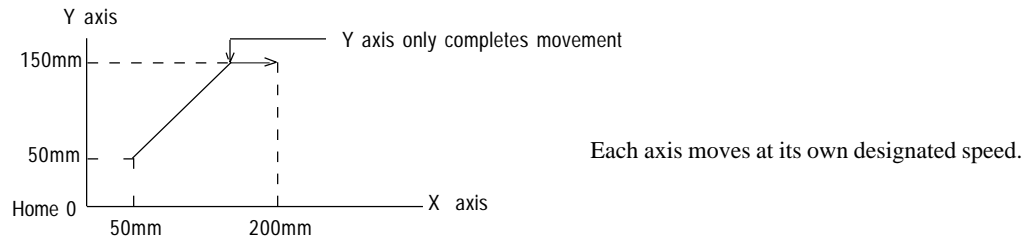
12. SEL Language

● MVPI (Incremental PTP Move)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	MVPI	Position No.		PE

[Function] Moves the actuator to the designated position number in reference to the current position from point to point without interpolation.

[Example] MVPI 1
 When the current position is (50, 50) and position 1 data is (150, 100), the actuators move 150 in the X direction and 100 in the Y direction to the position (200, 150).

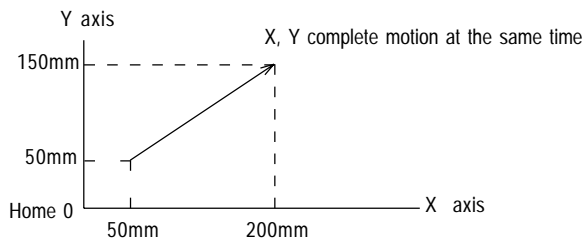


● MVLI (Incremental Interpolation Movement)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	MVLI	Position no.		Optional

[Function] Moves the actuator to the designated point in operand 1 from the current position while interpolating (not point to point).

[Example] MVLI 1
 When the current position is (50, 50) and the position 1 data is (150, 100), the actuators move to the position (200, 150) which is 150 in X direction and 100 in Y direction from the current position.



The tip of the combined motion for each of the axes moves at the designated speed. The path from the start to the finish point makes a straight line.

12. SEL Language

● ARCH (Arch Motion)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	ARCH	Starting position No.	Ending position No.	

[Function] In order to accomplish faster pick & place motion in the air and the tact time, the actuator performs an arch motion movement by changing the position ratio of the axis movement before it reached the intermediate target position.

[Example] LET 1 1 Assign 1 to variable 1.
 LET 2 3 Assign 3 to operand 2.
 ATRG 70 Set position ratio to 70%.
 ARCH *1 Execute arch move from variable 1 (content 1) to variable 2 (content 3) position.

*Note: See ATRG command

● PATH (Path Movement)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	PATH	Starting position No.	Ending position No.	PE

[Function] Actuator moves continuously between the designated starting point and the finishing point. The locus is a B-spline-type, free-form curve which passes through the inside of the designated coordinate. It is possible for the actuator to move close to the designated coordinate by increasing the acceleration. However, when it exceeds the maximum acceleration, an error will occur.

* Three and four axis motion can be performed by this command.

[Example 1] PATH 100 120 Moves continuously to position 100 ~ 120.

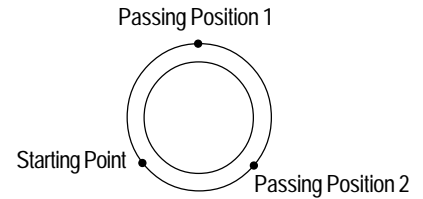
[Example 2] LET 1 50 Assign 50 to variable 1.
 LET 2 100 Assign 100 to variable 2.
 PATH *1 *2 Continuously move positions from variable 1 (content 50) to variable 2 (content 100).

12. SEL Language

● CIR (Circular Movement)

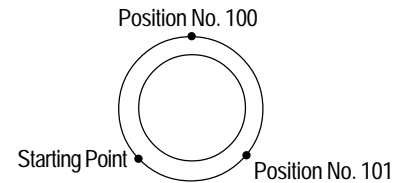
Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	CIR	Passing position 1	Passing position 2	Optional

[Function] Executes circular motion using the current position as the starting point and passing points 1 and 2. The rotation direction is determined by the position data. The following diagram shows CW (clockwise) motion but this can be changed to CCW (counterclockwise) by exchanging positions 1 and 2.



*This command is available for the specified orthogonal plane (Automatically selected by position data. Generally the XY plane is selected.) Also, care is needed if using this with the OFST command (check movement).

[Example] CIR 100 101
 Executes a circular motion passing through position numbers 100 and 101.



12. SEL Language

● ARC (Arc Movement)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	ARC	Passing position 1	Passing position 2	PE

[Function] Executes an arc motion from the current position to operand 2 position, passing through operand 1. The output turns OFF during an arc move and turns ON upon completion. However, when other commands as ARC, PATH and CIR follow in a consecutive order, turns ON at the 2nd point prior to the last point.

* This command is available for the specified orthogonal plane (automatically selected by position data). Generally the XY plane is selected.

[Example 1] ARC 100 101 Executes an arc motion from the current position to position 101, passing through 100.

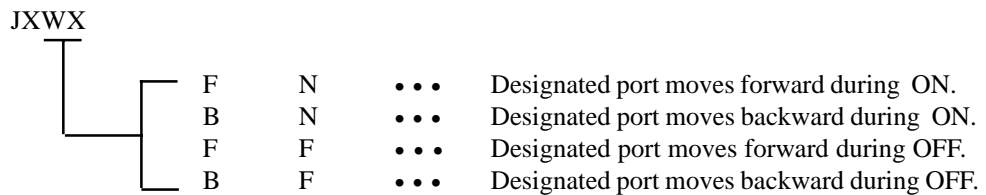
[Example 2] LET 1 5 Assign 5 to variable 1.
 LET 2 6 Assign 6 to variable 1.
 ARC *1 *2 Executes an arc motion from the current position (passing through variable 1 content 5) to variable 2 (content 6) position.

12. SEL Language

●JXWX (Jog move)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	JXWX	Axis pattern	I/O Flag	PE

[Function] Continues to move while the axis designated in operand 1 meets the condition I/O port flag designated in operand 2. Stops when the axis reached soft limit.



Note: HOLD is not available for this command.

[Example 1] JBWF 11001100 10 While input 10 is OFF, Axis 3, 4, 7, and 8 moves backwards.

[Example 2] LET 5 20 Assign 20 to variable 5.
 JFWN 10101010 *5 While input of variable 5 (content 20) is ON, move forward axis 2, 4, 6, and 8.

●STOP (Slows to a Stop)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	STOP	Axis pattern		PE

[Function] Axis designated in operand 1 axis pattern slows to a stop.

[Example] STOP 11001100 Axis 3, 4, 7 and 8 Slows to a stop.

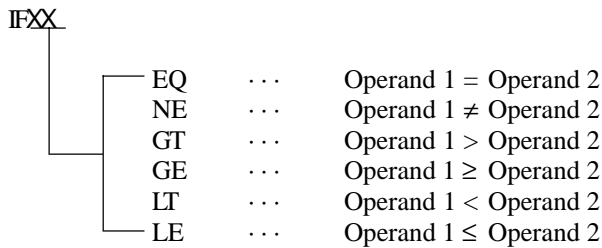
12. SEL Language

12.14 Structured IF Commands

● IFXX (Structured IF)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	IFXX	Variable No.	Data	

[Function] Compares the contents of the variable in operand 1 and the value in operand 2. When the condition is established, the program proceeds to the next step. When the condition is not established, if there is a corresponding ELSE command, the program proceeds to the next step after that. If not, it proceeds to the next step after the corresponding EDIF command. When the input condition is not established and there is no IFXX command executed, the program proceeds to the step following the corresponding EDIF. Up to 15 levels of nesting are available when ISXX and DWXX are combined.



[Example]

```

600 IFEQ 1 1 Select axis
    IFGE 2 0 Select moving direction
    JFWN 01 5 Move Axis 1 forward
    ELSE
    JBWN 01 5 Move Axis 1 backward
    EDIF
    ELSE
    IFNE 2 1 Select moving direction
    JFWN 10 5 Move Axis 2 forward
    ELSE
    JBWN 10 5 Move Axis 2 backward
    EDIF
    EDIF
  
```

Variable 1 selects Axis 1 or Axis 2. Variable 2 selects forward or backward to jog.
 When flag 600 is OFF, nothing is done and the program proceeds to the step after the last EDIF.

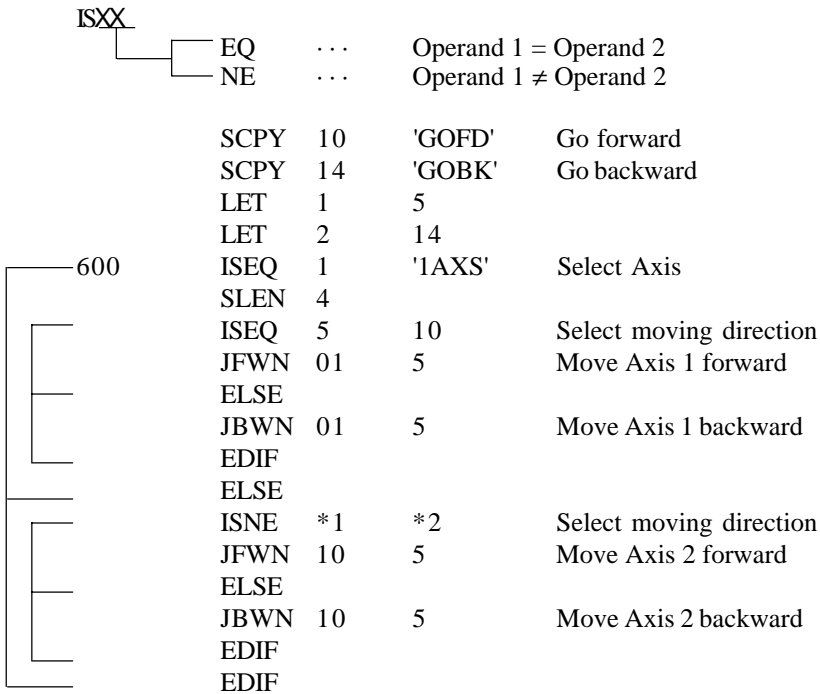
*Do not use GOTO (TAG) in between IFXX and EDIF.

12. SEL Language

● ISXX (String Comparison)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	ISXX	Column No.	Column No. Literal character	

[Function] Compares the character string in the column numbers in Operand 1 and Operand 2. When the condition is established, the program proceeds to the next step. When the condition is not established, if there is a corresponding ELSE command, the program proceeds to the next step after that. If not, it proceeds to the next step after the corresponding EDIF command. The length of the string to be compared is set by the SLEN command. If there is a literal character in either Operand 1 or Operand 2, the length to be compared is that of the literal character. When the input condition is not established and there is no IFXX command executed, the program proceeds to the step following the corresponding EDIF. Up to 15 levels of nesting are available when ISXX and DWXX are combined.



Column 1 ~ 4 is to select Axis 1, Axis 2 and column 5 ~ 8 is to select the jog direction.
 When flag 600 is OFF, nothing is done and the program proceeds to the step after the last EDIF.
 When column 1 ~ 8 contains the data shown below, Axis 1 moves forward.

	1	2	3	4	5	6	7	8
	1	A	X	S	G	O	F	D

12. SEL Language

● ELSE

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
		ELSE			

[Function] The ELSE command is used in conjunction with the IFXX command and ISXX command. When the condition is not established, the command following the ELSE statement will be executed.

[Example] Refer to IFXX and ISXX.

● EDIF (IFXX End)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
		EDIF			

[Function] Declares the end of an IFXX command.

[Example] Refer to IFXX.

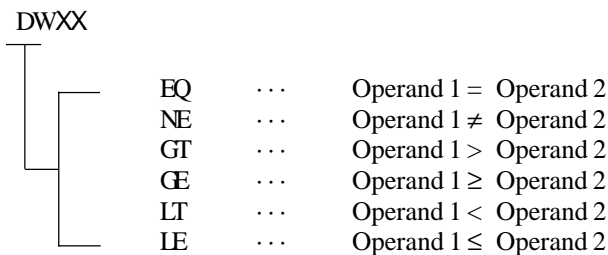
12. SEL Language

12.15 Structured DO Command

● DWXX (DO WHILE)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	DWXX	Variable No.	Data	

[Function] Compares the contents of the variable in Operand 1 and the value in Operand 2. While the condition is established, the commands are executed up to EDDO. When the condition is not established, the program proceeds to the step after the corresponding EDDO command. The LEAV command can be used to force the end of the loop. When the input condition is not established, the DWXX command is not executed and the program proceeds to the next step after the corresponding EDDO. Up to 15 levels of nesting are available when ISXX and DWXX are combined.



```

[Example]           DWEQ 1 0
                   .
                   .
                   600 LEAV
                   .
                   .
                   EDDO
  
```

While variable 1 is 0, the commands up to the EDDO command are repeated. If flag 600 turns ON during this time, the loop is forced to end and the program proceeds to the next step after the EDDO command.

● EDDO (End DO WHILE)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	EDDO			

[Function] Declares the end of the loop which started with DWXX. When a DWXX condition is not established, the program proceeds to next step after this command.

[Example] Refer to DWXX.

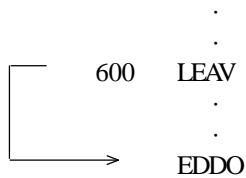
12. SEL Language

●LEAV (Escape From DO WHILE)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	LEAV			

[Function] Escapes the DOXX loop, then the program proceeds to the next step after EDDO.

[Example] DWEQ 1 0



While variable 1 is 0, the commands up to the EDDO command are repeated.

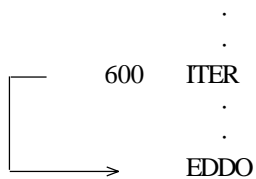
If flag 600 turns ON during this time, the loop is forced to end and the program proceeds to the next step after the EDDO command. If variable 1 is still 0, then the loop is repeated.

●ITER (Repeat)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	ITER			

[Function] Forces the control to move to EDDO during the DOXX loop.

[Example] DWEQ 1 0



While variable 1 is 0, the commands up to the EDDO command are repeated.

If flag 600 turns ON during this time, the loop is forced to end and control is forced to move to the EDDO command. If variable 1 is still 0, then the loop is repeated.

12. SEL Language

12.16 Branching commands

●SLCT (Start Selection Group)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	SLCT			

[Function] Branches to the next step after the OTHE command if none of the conditions set up by the WHXX, WSXX or any commands up to the EDSL command are met.

(Example) SCPY 1 'Right' Assign 'Right' to column 1 and 5.
 :
 600 SLCT Since the string in columns 1 through 5 are equal to 'Right', the
 commands that follow this WSEQ will be executed.
 WSEQ 1 'Right' Since the string in columns 1 through 5 are not equal to 'left',
 the commands that follow this WSEQ will not be executed. If it
 is neither, then the commands that follow OTHE are executed.
 WSEQ 1 'Left' When flag 600 is OFF, or if any one of the conditions is executed,
 : then end the select.
 :
 OTHE
 :
 EDSL

12. SEL Language

● WHXX (Selected When True Variable)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
		WHXX	Variable No.	Data	

[Function] This is used between the SLCT~ EDSL commands. Compares the contents of the variable in operand 1 to the value in operand 2. If the conditions are met, then the code following the WHXX will be executed up to the next WHXX. If the conditions are not met, the program will go to the next WHXX command or OTHE command or EDSL.

```

WHXX
├── EQ ..... Operand 1 = Operand 2
├── NE ..... Operand 1 ≠Operand 2
├── GT ..... Operand 1 >Operand 2
├── GE ..... Operand 1 ≥Operand 2
├── LT ..... Operand 1 <Operand 2
└── LE ..... Operand 1 ≤Operand 2
  
```

[Example]

```

LET 1 20 Assign 20 to variable 1.
LET 2 10 Assign 10 to variable 2.
:
SLCT Branches.
WHEQ 1 10 If the content of variable is 10, (1) is executed but since the
: content is 20, program refers to the next condition.
(1)
:
WHGT 1 *2 Executed if the content of variable 1 is greater than the
:- content of variable 2.
(2) Variable 1 (=20) > variable 2 (=10), so (2) is executed.
:
OTHE If no conditions are fulfilled, this is executed. Since (2) was
: executed, (3) will not be executed.
(3)
:
EDSL When one of the conditions is met and that command is
: performed, processing moves to EDSL. In this example,
(4) (2) and (4) are executed.
:
  
```

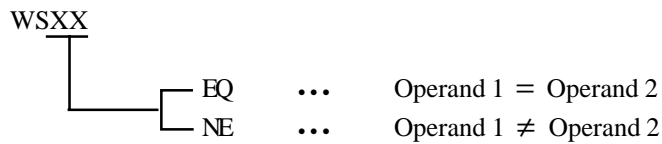
* When there is a possibility of several conditions being met, the WXXX command that appears first goes into effect and the commands that follow are not executed. When conditions are demanding, list the ones with the highest priority first.

12. SEL Language

● WSXX (Selected When True Character)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
		WSXX	Colum No.	Column No. · Literal character	

[Function] This is used during SLCT~EDSL. Compares the character string in the columns in operand 1 and operand 2. If the conditions are met, then the code following the WSXX will be executed up to the next WSXX. If the conditions are not met, the program will go up to the next WSXX command or OTHE command or EDSL. Comparison is made based on the length designated in the SLEN command. When operand 2 is a literal character, that is the length that is executed.



[Example]

SLEN	3		Sets the number of characters to be compared to 3.
SCPY	1	'ABC'	Assign 'ABC' to column 1.
LET	1	3	Assign 3 to variable 1.
:			
SLCT			Branches.
WSEQ	1	'XYZ'	If columns 1~3 are 'XYZ', (1) is executed but since columns 1~3 are 'ABC', this is not executed.
:			
(1)			
:			
WSEQ	1	'ABC'	If columns 1~3 are 'ABC', (2) is executed. Therefore, this code is executed.
:			
(2)			
:			
OTHE			If no conditions are fulfilled, this is executed. Since (2) was executed, (3) will not be executed.
:			
(3)			
:			
EDSL			When one of the conditions is met and that command is performed, processing moves to EDSL. In this example, (2) and (4) are executed.
:			
(4)			

* When there is a possibility of several conditions being met, the WXXX command that appears first goes into effect and the commands that follow are not executed. When conditions are demanding, list the ones with the highest priority first.

12. SEL Language

● OTHE (Selected in Case of Other)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
		OTHE			

[Function] This is used between SLCT~EDSL commands. This declares the command to be executed when no other conditions are met.

[Example] Please refer to SLCT, WHXX and WSXX.

● EDSL (End of Selected Group)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
		EDSL			

[Function] Declares the end of SLCT command.

[Example] Please refer to SLCT, WHXX and WSXX.

12. SEL Language

12.17 External input output command

● OPEN (Open Channel)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	OPEN	Channel No.		

[Function] Opens the channel specified in operand 1. Channels specified after this will be able to transmit and receive signals. An ending character must be set by the SCHA command before executing this command.

[Example] SCHA 10 Designate 10 (=LF) as the ending character.
OPEN 1 Open channel 1.

SCHA 13 Designate 13 (=CR) as the ending character.
LET 1 2 Assign 2 to variable 1. Open channel 2, the value contained in variable 1.
OPEN *1

● CLOS (Close Channel)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	CLOS	Channel No.		

[Function] Closes the channel specified in operand 1. Channels specified after this will be unable to transmit and receive signals.

[Example] CLOS 1 Close the channel.

[Example] LET 1 2 Assign 2 to variable 1.
CLOS *1 Close channel 2, the value contained in variable 1.

12. SEL Language

● READ

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	READ	Channel No.	Column No.	

[Function] Reads the character string from the channel in operand 1 to the column in operand 2.
 Stops reading when the character designated in the SCHA command appears.
 The column can be either local or global.

[Example]

SCHA	10			Set LF (= 10) for the ending character.
OPEN	1			Open channel 1.
READ	1	2		Read the character string from channel 1 to column 2 until LF appears.
CLOS	1			Close the channel.
LET	1	2		Assign 2 to variable 1.
LET	2	3		Assign 3 to variable 2.
SCHA	13			Set CR (= 13) for the ending character.
READ	*1	*2		Read the character string from channel 2 (content of variable 1) to column 3 (content of variable 2) until CR appears.

12. SEL Language

● WRIT (Write)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	WRIT	Channel No.	Column No.	

[Function] Writes the character string from the channel in operand 1 to the column in operand 2.
Stops writing after the character designated in the SCHA command is written.
The column can be either local or global.

[Example]

SCHA	10			Set LF (= 10) for the ending character.
OPEN	1			Open channel 1.
WRIT	1	2		Write the character string from channel 1 to column 2 until LF appears.
CLOS	1			Close the channel.
LET	1	2		Assign 2 to variable 1.
LET	2	3		Assign 3 to variable 2.
SCHA	13			Set CR (= 13) for the ending character.
WRIT	*1	*2		Read the character string from channel 2 (content of variable 1) to column 3 (content of variable 2) until CR appears.

● SCHA (Set Ending Letter)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	SCHA	Character code	Column No.	

[Function] Sets the ending letter to be used in the READ command and WRIT command. A value from 0 ~ 255 (character code used in BASIC) can be designated for the character.

[Example] Refer to the READ command and WRIT command.

12. SEL Language

12.18 String processing commands

● SCPY

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	SCPY	Column No.	Column No. Literal Character	

[Function] Copies the character string from the column in operand 2 to the column in operand 1. Copies only the length set by the SLEN command. When operand 2 is a literal character, that is the length copied.

[Example]

SCPY	1		'ABC'	Copy 'ABC' to column 1.
SLEN	10			Set the length of the operation to 10 bytes.
SCPY	100	200		Copy the length of the operation to 10 bytes. Copy 10 bytes from the column 200 to column 100.
LET	1	300		Assign 300 to variable 1.
LET	2	400		Assign 400 to variable 2.
SLEN	5			Set the length of the operation to 5 bytes.
SCPY	*1	*2		Copy 5 bytes from column 400 (the content of variable 2) to column 300 (the content of variable 1).

● SCMP (Compare Character String)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	SCMP	Column No.	Column No. Literal Character	EQ

[Function] Compares the column in operand 1 and the column in operand 2. Compares only the length set by the SLEN command. When operand 2 is a literal character, that is the length compared.

[Example]

SCMP	1		'ABC'	600	When column 1~3 are 'ABC' flag 600 turns ON.
SLEN	5				Set the length to be compared to 5 bytes.
SCPY	10	30		999	When the 5 bytes from column 10 and column 30 are equal, flag 999 turns ON.
LET	1	10			Assign 300 to variable 1.
LET	2	20			Assign 400 to variable 2.
SLEN	3				Set the length of the operation to 5 bytes.
SCMP	*1	*2		310	When the 3 bytes in column 10 (the content of variable 1) and the 3 bytes in column 20 (the content of variable 2) are equal, then, the output turns ON.

12. SEL Language

● SGET (Acquire Character String)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	SGET	Variable No.	Column No.	

[Function] Assigns 1 character from the column in operand 2 to the variable in operand 1.

[Example]

SGET	100		Assign 1 byte of column 100 to variable 1.
LET	1	3	Assign 3 to variable 1.
LET	2	1	Assign 1 to variable 2.
SCPY	1	'A'	Copy 'A' to column 1.
SGET	*1	*2	Assign 'A' in column 1 (content of variable 2) to variable 3(content of variable 1).

● SPUT (Set Character)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	SPUT	Column No.	Data	

[Function] Sets the data in operand 2 to the column in operand 1.

[Example]

SPUT	5	10	Set 10 (LF) to column 5.
LET	1	100	Assign 100 to variable 1.
LET	2	50	Assign 50 to variable 2.
SPUT	*1	*2	Set 50 ('2') which is the content of variable 2 to column 100 (content of variable 1).

12. SEL Language

● STR (Change Character String Decimal)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	STR	Column No.	Data	

[Function] Copies the data in operand 2 which has been converted to a decimal character string to the column in operand 1. Uses zero-suppress to match this to the length set by the SLEN command. Even if the data is longer than the length, the length set by the SLEN command takes precedence.

[Example] SLEN 5.3
 STR 1 123
 Set the length to a 5 digit integer with 3 decimals.
 The following will be set in column 1~9,

1	2	3	4	5	6	7	8	9
		1	2	3	.	0	0	0

LET 1 10
 LET 2 987.6543
 SLEN 2.3
 STR *1 *2
 Assign 10 to variable 1.
 Assign 987.6543 to variable 2.
 Set the length to a 2 digit integer with 3 decimals.
 The following will be set in column 10~15,

10	11	12	13	14	15
8	7	.	6	5	4

Since the data was longer than the set length, 9 in the 100s place and 3 in the 4th decimal place are cut off.

12. SEL Language

● STRH (Change Character String Hexadecimal)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	STRH	Column No.	Data	

[Function] Copies the data in operand 2 which has been converted to a hexadecimal character string to the column in operand 1. Uses zero-suppress to match only the integers to the length set by the SLEN command. Even if the data is longer than the set length, the setting by the SLEN command will take precedence.

[Example] SLEN 5
 STRH 1 255
 Set format for a 5 digit integer.
 The following will be set in column 1~5,

1	2	3	4	5
			F	F

LET 1 10
 LET 2 987.6543
 SLEN 2.3
 STRH *1 *2
 Assign 10 to variable 1.
 Assign 987.6543 to variable 2.
 Set format for a 2 digit integer with 3 decimals.
 The following will be set in column 10~11,

10	11
D	B

.3, the decimal segment of the SLEN command, and .6543 in variable 2 will be ignored. The integer expressed in hexadecimal notation is '3DB'. However, 3 in the third digit will be cut off since the length is set to 2 digits.

12. SEL Language

● VAL (Character String Change Data Decimal)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	VAL	Variable No.	Column No.	

[Function] Converts the data in the column in operand 2 to a number and assigns this to the variable in operand 1. The length set by the SLEN command will be converted.

[Example]

SCPY	10	'1234'	Copy the string '1234' into columns 10~13.
SLEN	4		Set the length to 4 bytes.
VAL	1	10	'1234' in column 10 is converted to the number 1234 and assigned to variable 1.
LET	1	100	Assign 100 to variable 1.
LET	2	20	Assign 20 to variable 2
SCPY	20	'1234'	Copy '1234' to column 20~23.
SCPY	24	'.567'	Copy '.567' to columns 24~27.
SLEN	8		Set the length to 8 bytes.
VALH	*1	*2	'1234.567' in column 20 (content of variable 2) will be converted to the binary number 1234.567 and assigned to variable 100 (content of variable 1).

12. SEL Language

● VALH (Character String Data Hexadecimal)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	VALH	Variable No.	Column No.	

[Function] Converts the hexadecimal data in the column in operand 2 to a decimal number and assigns this to the variable in operand 1. The length set by the SLEN command will be converted. Only the integers will be converted and the decimal places will be disregarded.

[Example]

SCPY	10	'1234'	Set '1234' in column 10.
SLEN	4		Set the length to 4 bytes.
VAL	1	10	The hexadecimal number '1234' in column 10 is converted to the number 4660 and assigned to variable 1.
LET	1	100	Assign 100 to variable 1.
LET	2	20	Assign 20 to variable 2
SCPY	20	'ABCD'	Copy 'ABCD' to column 20.
SLEN	4		Set the length to 4 bytes.
VALH	*1	*2	The hexadecimal 'ABCD' in column 20 (content of variable 2) will be converted to the binary number 43982 and assigned to variable 100 (content of variable 1).

● SLEN (Set Length)

Expansion condition (AND · OR)	Input condition (I/O · Flag)	Command			Post (Output port · Flag)
		Command	Operand 1	Operand 2	
Optional	Optional	SLEN	Length		

[Function] Sets the length for the string command.

SCMP	...	Decimals Invalid
SCPY	...	Decimals Invalid
ISXX	...	Decimals Invalid
STRH	...	Decimals Invalid
VAL,VALH	...	Decimals Invalid
STR	...	Decimals Valid

[Example] Refer to each of the commands above.

13. Error Codes

13.1 List of error codes

Error Code	Error Name	Explanation
A1	External Interrupt Error	1. Motor over current 2. Over regenerative current (over negative load) 3. Driver overheat
A2	Motor Overload Error	Mechanical overload of motor
A3	Deviation Error	Motor is unable to perform properly due to mechanical overload
A4	Software Limit Error	Exceeded software limit
A5	Pole Sense Error	Unable to sense pole
B0	No Program Error	Program does not exist
B1	Program Execution Error	Execution of a currently executing program
B2	Program Over Error	Number of tasks exceeds those set as parameters
B3	Double Subroutine Number Error	Two or more of the same subroutine number are used
B4	Double Tag Number Error	Two or more of the same tag number are used
B5	Undefined Subroutine Number	Subroutine number is not defined
B6	Undefined Tag Number	Tag number is not defined
B7	Subroutine Pair Error	BGSR and EDSR are not the same quantity
B8	Step 1 BGSR Error	Step 1 is a BGSR Error
B9	DO, EDDO Pair Error	DO and EDDO are not the same quantity
BA	DO Nest Over Error	DO was used more than 15 times
BB	IF Pair Error	IF and ELSE are not the same quantity
BC	ELSE Error	ELSE was used in a place which was not between IF and EDIF
C0	No Homing Error	Homing was not performed before running actuators
C1	Point Data Error	Attempt has been made to executed unregistered point data
C2	Axis Double Execution Error	Move command given to axis currently moving
C3	Software Limit Error	Software limit exceeded in program
CA	Column Error	Column number was set outside the range of 1 ~ 999
CB	Channel No. Error	Device was set outside the range of 1 ~ 2
CC	Terminator Error	Ending letter was not set
CD	Source No. Error	Source number was set outside the range of 1 ~ 9
CE	S Motion Percent Error	S motion percent ws set outside the range of 0 ~ 50%
CF	Arch Trigger Error	Trigger was set outside the range of 50 ~ 100%
D0	Acceleration Error	Acceleration exceeds limits
D1	No Velocity Error	Velocity has not been set
D2	Override Error	Override was set outside the range of 1 ~ 100%
D3	Angle Error	Angle was set outside the range of 0.1 ~ 120 degrees
D4	Axis Pattern Error	Axis pattern was not set correctly. Displays D4 also for C1 (point data error)
D5	Axis Number Error	Axis number was set outside the range of 1 ~ 8
D6	Axis Error	More than 3 axes are designated in circular/arc motion
D7	Program Number Error	Program number exceeds the limit
D8	Position Number Error	Position number exceeds the limit
D9	Point Number Error	Negative number was input in the point number
DA	Flag Number Error	Flag is not assigned correctly
DB	Variable Error	Variable is not assigned correctly
DC	Digits Over Error	Assigned number exceeds 8 digits (binary 32 bits)
DD	Division (0) Error	Result of the division is "0"
DE	Circular Motion Computation Error	Position data that cannot perform circular motion was input
DF	Task Level Error	Task level was set outside of the range of 1 ~ 5
E0	Undefined Command Error	Attempted to execute undefined command
E1	Subroutine Over Nesting Error	Nesting of more than 15 subroutines
E2	Subroutine Under Nesting Error	EXSR and EDSR are not making a pair
E3	Controlling Column Error	Use of condition is not correct
EG	EMG Error	Emergency (Emergency Stop) was asserted
F0	Interrupt Error	Motor CPU and Interrupt management do not match

* For the DS Type, E is attached in the front of the error code, displayed on the 1st column.

13. Error Codes

13.2 What to do When an Error Code Occurs

Below we indicate what to do in case any of the error codes described on the preceding page appear in the 7-segment display on the face of the controller.

(1) A1 ~ A5 alarms related to the servo

When one of these alarms related to the axis appears, determining which axis is the cause of the error makes it easier to solve the problem. One of the ways to do this is to judge by the axis status or movement at the time the error was generated. After the error is generated, you can try moving the axis manually if it is a small system. If the axis (when there is no brake) moves without resistance, there is a good possibility that this is the axis causing the problem. When these errors occur, you should ascertain the status of the actuator such as whether it was in the middle of homing.

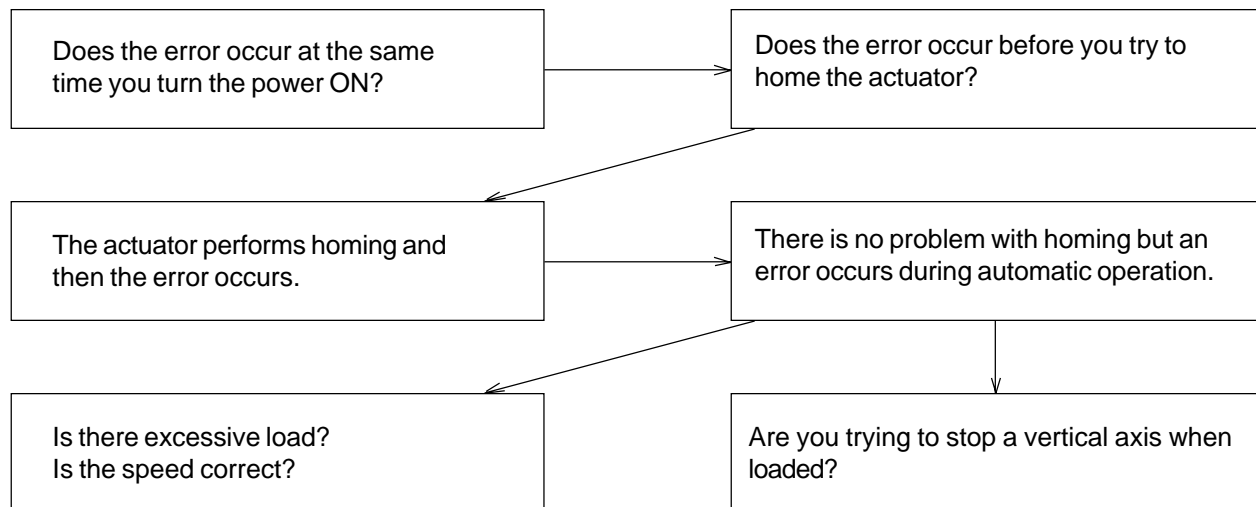
In the case of an A2 alarm where there is excessive load, the cause of the error must be corrected. If you are unsure of the cause, turn on the emergency stop or turn off the power and then after about 10 seconds, turn the power back on to see how the actuator runs. If you cannot find what the trouble is, please contact IAI or one of its agents.

When there is an A3 deviation error, it is possible that something is wrong with the connector cable.

When there is an A4 error, it is almost always caused by a programming error. Recheck the program to make sure that you are not trying to move the actuator beyond the stroke length.

When there is an A5 error, check to see how the axis is moving and then contact IAI. A5 errors can be caused by encoder breakdown, cable problems or driver problems.

Checklist



13. Error Codes

(2) B0 - BC Programming Errors

Group 1

An error will be displayed when there is a problem with the written program itself or the program that was started up. In this case, alarm output 300 will not be asserted.

Code	Error	What To Do
B0	No program	The program that was run from external startup has no defined content. Run the program with the proper number.
B1	Program run error	The program that was running restarted. This is not necessarily a problem - the code display is just a warning to the operator.
B2	Program over error	This occurs if you try to run 17 or more programs. Multi-tasking handles only up to 16 programs.
B3	Subroutine no. multiple definitions	A duplicate subroutine number was used. Revise the number.
B4	Tag no. multiple definitions	A duplicate tag number was used. Assign a different number.
B5	Subroutine no. undefined	The subroutine number being called up is undefined. Created the designated subroutine or check the number being input.
B6	Tag no. undefined	The GOTO destination tag is undefined. Check for tag number error or create tag definition.
B7	Subroutine pair error	BGSR is not paired with EDSR. Another BGSR was started before the EDSR was executed which is not allowed.
B8	Step 1 has a BGSR error	Defining a BGSR at the head of a program is not allowed. Define the subroutine at the end of the program.
B9	Too many DO, EDDO nested layers	DO is not paired with EDDO. The number of EDDO is greater or lesser than DO and needs correction. To perform homing after this error occurs, you must turn on the emergency stop once. Currently, this display may appear in the case of a [BB] error.
BA	Too many DO nested layers	This occurs when more than 15 DO nesting layers are set or there are more than a total of 15 nesting layers for the expansion commands. Take note of these nesting layers when using the expansion commands.
BB	IF pair error	IF is not paired with EDIF. The number of EDIF is greater or lesser than IF. Make sure to pair these correctly.
BC	ELSE error	ELSE was used at some place other than between IF and EDIF. Correct the syntax.

13. Error Codes

(3) C0 - CF programming errors Group 2/Command Error - 1

This group of errors is also related to programming, but primarily arises from the way the commands are used.

C0	Homing incomplete error	Tried to execute move command without performing homing. After the power is turned ON, or after an emergency stop, homing must always be performed.
C1	Position designation error	Tried to move to a position not specified by the position data. Set position data.
C2	Axis-in-motion error	Commanded axis to move again while already in motion. Be careful when doing multi-tasking.
C3	Soft limit error	Commanded axis to move beyond soft limit during the program. Or, the soft limit went into effect when mistakenly changing the parameter settings. Check conditions and make necessary corrections.
CA	Column error	Specified a column number outside the 1 ~ 999 range with respect to communication. Column number should be within 1 ~ 999.
CB	Channel number error	A channel device other than 1 - 2 was specified. Currently, only 1 - 2 can be used.
CC	Terminator error	The terminating character was not specified. Set the terminating character using the SCHA command.
CD	Resource No. error (Reserve error - currently not used)	Resource no. outside of 1 ~ 9 was specified. (Currently, commands that would generate this error are not supported).
CE	S motion percent error	An S motion percent other than 0 ~ 50 was specified. Reset using the range 0 ~ 50.
CF	Arch trigger error	Trigger setting outside of 50 ~ 100 was specified. Reset using the range 50 ~ 100.

(4) D0 - DF programming errors Group 3/Command Error - 2

Like Group 2 above, this group of errors primarily arises from the way the commands are used.

D0	Acceleration error	Commanded axis to accelerate beyond the parameter upper limit. Although acceleration speed can be set at a fairly high value, 0.3G is the basic speed that can be guaranteed. If this error occurs, there is a problem with the speed.
D1	No speed error	There is no speed setting written in this program. It is necessary to specify a speed in the program using the VEL command or using the position data.

13. Error Codes

D2	Override error	The override was specified outside the range of 1 ~ 100%. Specify value within this range.
D3	Angle error	The angle parameter for the circular move command was specified outside the range of 0.1 ~ 120°. Specify angle within this range.
D4	Axis pattern error	The axis pattern designation is incorrect. Or, the problem is the same as for a C1 position setting error. Correct the data setting.
D5	Axis number error	An axis number outside the range 1 ~ 8 was specified or an axis not supported by the controller was specified. Set the correct axis number.
D6	Circular axis designation error	There are data settings for more than 2 axes. The ARC/CIR commands can only be executed in two dimensions. Correct the data setting.
D7	Program number error	Operator attempted to run a program number higher than 64. Only program numbers 1 ~ 64 can be run.
D8	Point number error	A point number higher than 2000 was specified. For point numbers, only the numbers 1 ~ 2000 can be used.
D9	Point data error	Point data was specified as a negative number. Position data must be a positive number although other data can be stored as a negative number.
DA	Flag number error	The flag number assigned was incorrect. Flags can only use numbers 600 ~ 999.
DB	Variable error	The variable number assigned was incorrect. Variables and variables with * indicator must be within 1 ~ 399.
DC	Digit over error	The value input in operand 1, 2 exceeds 8 digits. Or, a value exceeding the range of 32 bits was assigned in the IN command. Assign a value within 8 digits and 32 bits.
DD	Divide by 0 error	Result of the division calculation is 0. To perform a division calculation, the denominator must be a number other than 0. Recheck the algorithm.
DE	Circular move calculation error	Position data was assigned that does not allow a circular move. Set position data that allows circular move.
DF	Task level error (reserve error)	A task level number other than 1 ~ 5 was assigned. (Currently, the command that would cause this error is not supported).

13. Error Codes

(5) E0 - E3 programming errors Group 4/Command error - 3

These errors, like those in sections 3 and 4 above, primarily arise from the way the commands are used.

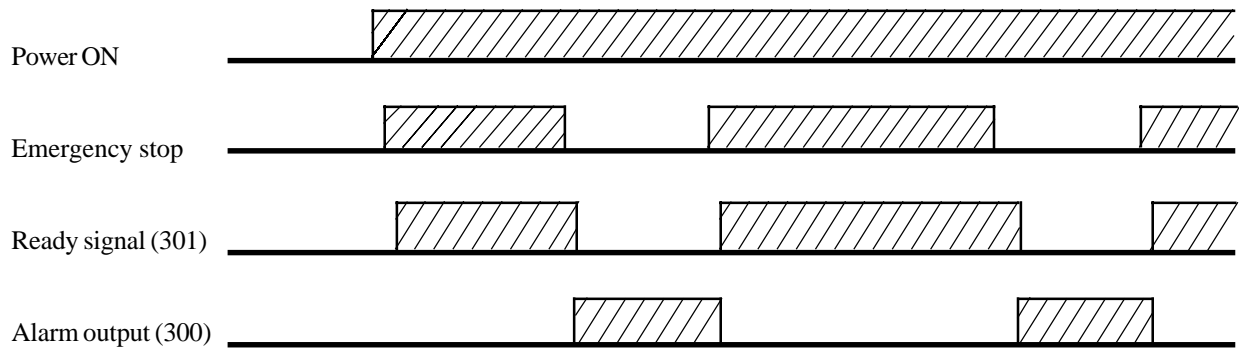
E0	Undefined command error	Attempted to execute an undefined command. If you use the PC interface software, the check function will prevent this.
E1	Subroutine overnesting error	There are more than 15 subroutines nested. This alarm occurs after trying to run the program. Write the program so that the number of nested subroutines is less than 15. Also note that when there is complex use of the IF command, this is likely to cause a [BA] error.
E2	Subroutine under nesting error	BGSR is not paired with EDSR. Another BGSR is found before the EDSR. Correct the syntax.
E3	Controlling column error	The expansion condition is used incorrectly. If you are using the PC software, it will sometimes indicate a syntax error at the time of input.

(6) EG error - Emergency Stop

If an EG alarm occurs, consider the following.

① Emergency stop signal asserted

Determine what triggered the emergency stop and then release the emergency stop (press the button). During an emergency stop, the ready signal or output 301 is OFF and the alarm or output 300 is ON. However, if the emergency stop is asserted when the power is turned ON, alarm output 300 will not go ON. The 300 output functions after the ready signal is given.



13. Error Codes

② Another thing to consider with an emergency stop

Usually, the emergency stop input is tied to a ground. In the case where you are using an external power supply, the power supply voltage can drop, causing an emergency stop to occur. The way the circuitry is designed, the 24V DC power supply must be turned on before the controller, and the power supply must not be turned OFF while the controller is in operation.

In addition, if there is a malfunction in the controller causing a part of the unit to break down, the EG condition will remain in effect and homing cannot be performed. If this happens, please contact IAI.

7 Other Errors

The following errors occur only rarely under normal operating conditions.

F0	Interrupt error	Motor CPU and the interrupt number do not match. This error could occur when noise interference causes faulty controller operation or there is a breakdown in the hardware. Homing is possible after turning the power OFF and then ON again. If this error occurs several times, contact IAI.
FF	CPU fault error	This indicates a fatal error occurred in the main CPU processing. In this case, the controller will stop working. You must perform homing after turning the power OFF and then ON. This alarm will occur if too many digits were used in a floating point calculation. Make sure that calculations using a real variable will produce a value within $\pm 3.4x$

Precautions when handling errors/alarms

When you need to turn the power OFF and then ON again, please make sure to wait approximately 15 seconds after turning the controller power OFF before turning it back ON.

If an error occurs in which you cannot perform homing, please contact a service representative after you have checked out the condition at the time the error occurred as thoroughly as possible. In some cases, the problem might be in the program itself and the representative may request a program list from you.

Intelligent Actuator Inc.

2690 W. 237th Street

Torrance, CA 90505

310-891-6015 / 310-891-0815 (Fax)

www.intelligentactuator.com